

# DeepKeyStego: Protecting Communication by Key-dependent Steganography with Deep Networks

Zheng Li

School of Computer Science and Technology  
Shandong University  
Qingdao, China  
zheng.li@mail.sdu.edu.cn

Ge Han

School of Computer Science and Technology  
Shandong University  
Qingdao, China  
hangehg@126.com

Shanqing Guo\*

School of Cyber Science and Technology  
Shandong University  
Qingdao, China  
guoshanqing@sdu.edu.cn

Chengyu Hu

School of Cyber Science and Technology  
Shandong University  
Qingdao, China  
hcy@sdu.edu.cn

**Abstract**—Hiding both the presence and the content of secret information against eavesdropping over public communication channels is crucial for protecting privacy sensitive communication. Steganography is the art of hiding confidential information into normal carriers. Images are the most widely used containers for steganography. However, most of the current popular image steganographic schemes are designed with prescribed human-based rules and can be effectively detected by existing steganalysis tools. Even though the steganography implemented by deep networks has a better performance against steganalysis to some extent, it is still exposed to a threat that an attacker with access to the decoding model can recover the embedded information from steganographic images.

Hence, we introduce the keys to the steganography based on deep networks and design symmetric and asymmetric steganographic schemes where the embedded information will never be recovered without secret keys. We further propose DeepKeyStego, a framework training key-dependent steganographic models with novel network structures. Compared to previous methods, DeepKeyStego based on novel network structures achieves excellent invisibility and produces a steganographic algorithm without any prescribed rules or hand-crafted features, which can perform competitively remarkable undetectability. Moreover, DeepKeyStego is the first to successfully design and implement symmetric (secret key) and asymmetric (public key) steganography, which considers the usage of keys to enhance security. Finally, we simulated our trained steganography in a practical situation and proved that the decoder can successfully recover 99.8% of embedded information, showing rather effectiveness and integrity.

**Index Terms**—Steganography, adversarial training, symmetric, asymmetric

## I. INTRODUCTION

The security of communication is of skyrocketing concern because the current development of computers and network provide an inexpensive and quick means for information transmission. Steganography, as an approach to protect communication, plays an important role in hiding the presence of sensitive information within transmitted messages, which has much

practical significance in military, e-commerce, government affairs, copyright protection, etc.

The challenge of steganography is that embedding a message by altering the appearance and underlying statistics of the carrier, and meanwhile a steganalyzer tries to detect the existence of embedded message and even recover it viciously. In addition to the carrier itself and the amount of payload, the extent of alteration depends on the concrete steganographic encoding algorithm as well.

Researchers have introduced techniques facilitating steganography to protect privacy-sensitive communication. Several studies [1]–[3] proposed multiple promising steganographic algorithms and proved that the secret message can be concealed into cover images by hand-crafted rules. However, they'll consistently alter the statistics of the image, resulting in reliable detection [9]. In addition, with the rise of deep learning in recent years, deep learning has been applied to steganography. One of the current studies [4] proposed a GAN-based method using a sample network architecture with linear layers to hide message resulting in weak invisibility. Furthermore, their work did not take the usage of keys into consideration. Nevertheless, existing works in steganography [5], [6] have formally defined and discussed the great importance of keys for security. Once the model is stolen or leaked, attackers can easily use the model to recover the secret message.

Therefore, we propose DeepKeyStego, a novel key-dependent steganographic framework, for hiding secret data into the ordinary image with the aid of keys. Moreover, symmetric (secret-key) and asymmetric (public-key) steganographic schemes are separately proposed and each scheme is successfully designed and implemented. In brief, our work includes three main contributions:

- taking advantage of keys to enhance the security of steganography against any adversary having full knowledge of the decoding model;

- proposing a novel steganographic framework, DeepKeyStego and introducing the adversarial training technique into the training of deep convolutional networks.
- conducting an empirical validation and a real-world simulation to demonstrate the effectiveness and integrity of DeepKeyStego.

## II. RELATED WORK

In this section, two types of previous studies directly associated with ours, including various classical steganographic methods and current promising GAN-based methods, are briefly reviewed.

### A. Classical methods

A wide variety of classical steganographic settings and methods have been proposed in the literature. One of the most popular and easy-to-implement steganographic encoding algorithms is the Least Significant Bit (LSB) algorithm [7]. Its main idea is to store the secret message in the least significant bit of some color channel of each pixel in a given image container. There are several popularly used and sophisticated steganographic algorithms and tools developed from LSB, like WOW [1], HUGO [2], S-UNIWARD [3]. Though often not visually observable, they may alter the statistics of containers. Statistical analysis of container files will reveal whether the resultant file is different from the unchanged file, leading to effective detection [8], [9]. Unlike artificially designed classical methods, DeepKeyStego is based on deep neural networks, to be trained for completing steganographic goals without any prescribed human-based rule or hand-crafted feature, showing excellent performance in undetectability.

### B. GAN-based methods

This subsection reviews two crucial works related to our work. Volkhonskiy et al. introduced a steganographic generative adversarial network (SGAN [11]) to generate images as steganographic containers. Hiding information into containers produced by SGAN is safer than in normal containers, which is in the different goal and way to DeepKeyStego. Ste-GAN-ography [4] also applied adversarial training to steganography, but it generates stego images, which are embedded with messages, rather than container images. Furthermore, it does not take the usage of keys into consideration. Once the decoding model is leaked or stolen, the attackers can easily use the model to recover the secret message from the steganographic image, causing serious security problems. Therefore, we have introduced keys to the steganography, and we apply one fresh key for each cover image or secret message. Even if the decoding model is in the hands of an attacker, the attacker cannot recover the original message without the corresponding key. Furthermore, relying on the design of novel network structures, we show stronger invisibility quantificationally.

## III. STEGANOGRAPHIC ADVERSARIAL TRAINING

DeepKeyStego is proposed to hide the secret message into a cover image. In this section, we first provide an overview

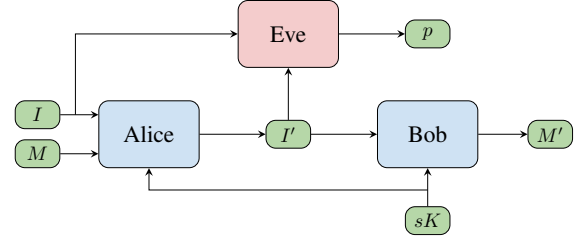


Fig. 1. Symmetric steganography

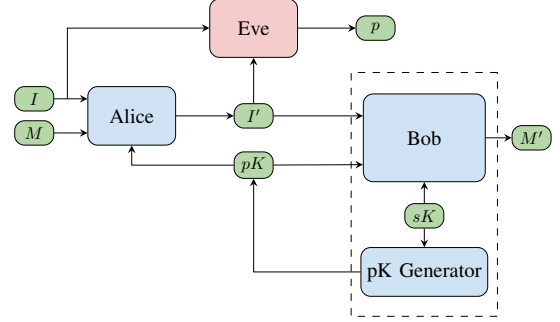


Fig. 2. Asymmetric steganography

of DeepKeyStego. Then we introduce the elaborate processing and organization of DeepKeyStego.

### A. Framework overview

In this paper, we discuss key-dependent steganography with deep networks in two scenarios: symmetric and asymmetric. Two communication parties Alice and Bob, as depicted in Fig.1, leverage the same key (secret key) for both encoding and decoding in symmetric steganography. The shared key is therefore presumed to be transmitted over a secret channel that is not accessible for attackers. For most common cases, this scheme is convenient and safe enough. However, under some higher security requirements, such as commercial transactions or government communications, we exhibit a more practical scheme of key-dependent steganography in Fig. 2, asymmetric steganography. There is additionally a public key generator, *pKey Generator*, on the side of Bob. It generates the public key from the given secret key, and the public key is transmitted over a public channel to Alice for encoding so that the secret key does not require any transmission. The definitions of four basic models widely applied in our work are introduced as follows:

**Definition 1 Alice.** The model *Alice* with trainable parameters  $\theta_A$  is an encoder that can hide the secret message into a cover image. Given the cover image  $I$ , secret message  $M$  and key (secret key  $sK$  or public key  $pK$ ), Alice outputs steganographic image  $I'$  (stego image).

**Definition 2 Bob.** The model *Bob* with trainable parameters  $\theta_B$  is a decoder that can recover the secret message from stego image with the aid of key ( $sK$  or  $sK + pK$ ), outputting the decoded message  $M'$ .

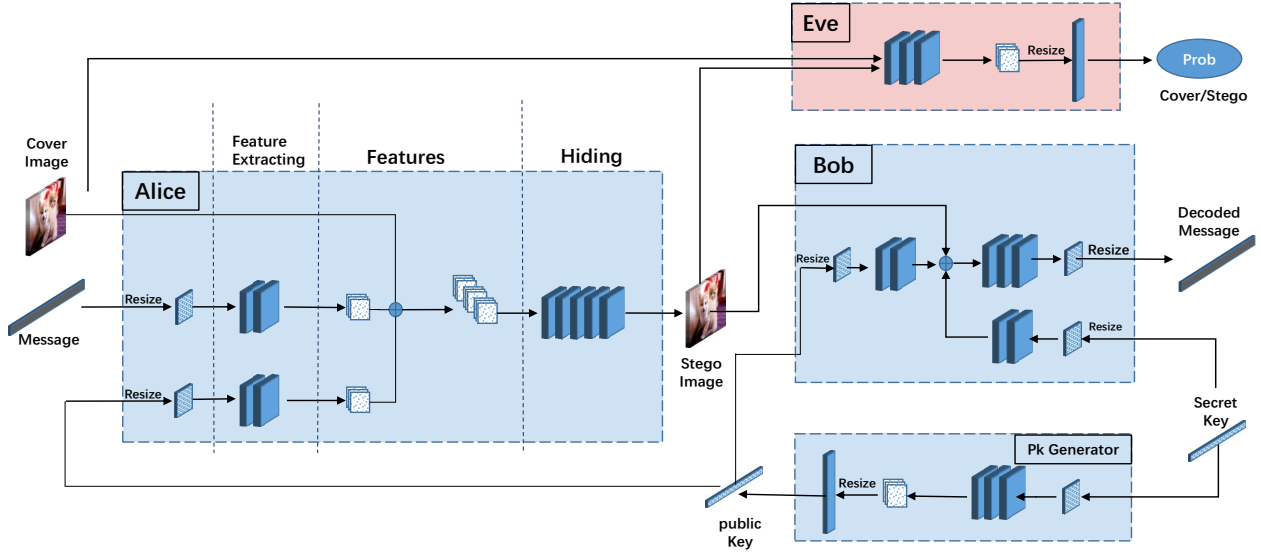


Fig. 3. The asymmetric workflow of DeepKeyStego

**Definition 3 Eve.** The model *Eve* with trainable parameters  $\theta_E$  is a discriminative adversary that attempts to distinguish stego images from real ones by estimating the probability  $p$  that the input is an innocent image sampled from training data rather than a loaded image generated by *Alice*.

**Definition 4 pK Generator.** In the asymmetric scheme, the lightweight model *pK Generator* with trainable parameters  $\theta_K$  is a public-key generator that accepts the secret key  $sK$  as input, outputs the public key  $pK$  which can be transmitted over public communication tunnels.

Each model is designed as deep convolutional networks with trainable parameters  $\theta$  and our training method is exactly the process to adjust these parameters. The detailed adversarial training schemes of symmetric and asymmetric steganography are separately described in the rest of this section.

### B. Model Architecture

**Symmetric.** We denote the abbreviation of Conv2d, ConvTranspose2d, Batch Normalization, and LeakyReLU as Conv, ConvT, BN, and LReLU. In our experiments, all blocks mentioned below, unless otherwise specified, have  $3 \times 3$  kernels, stride 1, and padding 1.

The encoder *Alice* receives a cover image  $I$  of shape  $C \times H \times W$ , a binary secret message  $M \in \{0, 1\}^{L_M}$  of length  $L_M$  and a binary random secret key  $sk \in \{0, 1\}^{L_{sk}}$  of length  $L_{sk}$ . The encoder applies 2 ConvT-BN-LReLU blocks with  $4 \times 4$  kernel, stride 2, padding 1, and  $C$  output filters each for preprocessing key, and 2 ConvT-BN-LReLU blocks<sup>1</sup> with  $4 \times 4$  kernel, stride 2, padding 1 and  $C$  output filters each for processing message. The message  $M$  and key  $sk$  are separately resized to a tensor of size  $1 \times h \times w$  where  $h \times w = (L_M \text{ or } L_{sk})$ , then fed into preprocessing block generating a feature

<sup>1</sup>The setting is different for the different length of a message.

map of size  $C \times H \times W$ , the same size of the cover image  $I$ . The two feature maps and  $I$  are concatenated to a tensor of size  $3C \times H \times W$  where the two feature maps are dispersed throughout all bits in the image, then fed to 4 Conv-BN-LReLU blocks with 64 output filters each and a Conv-Sigmoid blocks with  $C$  output filters to reduce the number of channels from  $3C$  to  $C$ . Finally, we get the output of *Alice*, the stego image  $I'$  of size  $C \times H \times W$ .

The decoder *Bob* accepts the stego image and the secret key as input. The secret key  $sk$  firstly is resized and fed to 2 ConvT-BN-LReLU blocks with  $4 \times 4$  kernel, stride 2, padding 1, and 1 output filters each to generate a feature map. It is concatenated with the stego image  $I'$ , which is then fed into 2 Conv-BN-LReLU blocks with 64 output filters each, then a Conv-BN-LReLU block and a Conv-Sigmoid block<sup>1</sup> with  $4 \times 4$  kernel, stride 2, padding 1 and 1 output filters each to output the decoded message  $M'$ .

The adversary *Eve* has a simpler structure than the decoder. *Eve* applies 3 Conv-BN-LReLU blocks with 64 output filters each, and particularly a fully connected layer followed by the sigmoid activation, and outputs a binary classification  $p(I)$  for a given image  $\tilde{I} \in \{I, I'\}$ , i.e. either a cover image or stego image.

**Asymmetric.** In asymmetric steganography, only *Bob* has access to the underlying secret key. The secret key  $sk$  is firstly resized to a tensor of size  $1 \times h \times w$  where  $h \times w = L_{sk}$ , and fed to a Conv-BN-LReLU block and a Conv-BN-Sigmoid with 1 output filter generating a random public key  $pk$ . Then the  $pk$  is resized to a tensor of length  $L_{pk}$ ,  $L_{pk} = L_{sk}$ . Finally, the public key  $pk$ , cover image, and secret message participate in the training process which is the same as that of the symmetric scheme. Particularly, the decoder applies another 2 ConvT-BN-LReLU blocks with 1 output filters each

to the public key, generating a feature map. The  $pk$  feature map is concatenated with  $sk$  feature map and stego image  $I'$ , fed into next several layers. Fig. 3 illustrates the workflow of the asymmetric scheme.

### C. Loss Function

**Symmetric.** Unlike previous works [4], they just take the mean square error (MSE) or the Euclidean distance, which only punishes the large errors of the corresponding pixels between two images, ignoring the underlying structure in the images. So we introduce the structural similarity index (SSIM) [13], which consists of multiple comparisons: brightness, contrast, and structure.

$$MSE(I, I') = 1/n \sum (x_i - y_i)^2 \quad (1)$$

$$SSIM(I, I') = \frac{(2\mu_I\mu_{I'} + c_1)(2\sigma_{II'} + c_2)}{(\mu_I^2 + \mu_{I'}^2 + c_1)(\sigma_I^2 + \sigma_{I'}^2 + c_2)} \quad (2)$$

Considering pixel value difference and structure differences simultaneously, we use MSE (1) and SSIM (2) together. The value range of SSIM is  $[0, 1]$ , and the higher the index is, the more similar the two images are. So the loss (metric) measuring the difference between the cover and stego images is as below:

$$Loss_I(I, I') = \alpha MSE(I, I') + \beta(1 - SSIM(I, I')) \quad (3)$$

where  $\alpha, \beta > 0$  are the hyper parameters to trade off the importance of each individual metric term.

In fact, to measure the reality of images does not only depend on MSE or SSIM. Classic steganographic algorithms, such as WOW [1], HUGO [2], S-UNIWARD [3], conceal the secret information into the cover image by hand-crafted rules, which also shows excellent performance in MSE and SSIM. However, they may cause changes in the statistical properties of the image, and many current steganalysis techniques can effectively detect the stego images [8], [9]. To evade current detection, we apply the adversarial training [10] to our scheme. We introduce an adversarial loss to present the ability to defend against discriminator detecting a stego image  $I'$ :

$$Loss_p(I') = \log(1 - p(I')) \quad (4)$$

We apply the L1 distance to measure the message reconstruction loss for Bob.

$$Loss_M(M, M') = \sum_i^{L_M} |M_i - M'_i| \quad (5)$$

Since Alice embeds message  $M$  into a cover image  $I$  with the aid of secret key  $sk$  and Bob attempts to reconstruct message with the same  $sk$ , the secret key  $sk$  plays an important role in the design of symmetric steganography as a bridge strengthening the link between Alice and Bob. Besides  $sk$ , the reconstruction performance of Bob also depends on the other input, stego image  $I'$  generated by Alice. That means, the design of decoder shall be in accordance with the encoder and the parameters of Bob shall be tuned simultaneously with Alice's. We perform stochastic gradient descent (SGD) [14] on updating  $\theta_A$  and  $\theta_B$ , then obtain the optimal Alice and

Bob by minimizing the following loss over the distribution of input images, message, and key:

$$L_{AB}(\theta_A, \theta_B) = Loss_I(I, I') + \gamma Loss_M(M, M') + \delta Loss_p(I') \quad (6)$$

$$O_{AB}(\theta_A, \theta_B) = \operatorname{argmin}_{(\theta_A, \theta_B)} (L_{AB}(\theta_A, \theta_B))$$

where  $\gamma, \delta > 0$  are the hyperparameters to trade off the quality of stego images, revealed secret message and the ability to defend against Eve's detecting.

The discriminator Eve incurs a classification loss function from its predictions:

$$Loss(\tilde{I}) = (1 - \log(p(I))) + \log(p(I')) \quad (7)$$

Minimizing the above loss means improving the ability to detect whether a given image contains an encoded message. Meanwhile, this provides an adversarial property against Alice and through the iterative training of Alice and Eve, the quality of generated stego images will be improved. We also perform SGD on updating  $\theta_E$ , and obtain the optimal Eve by minimizing the following loss over the same distribution:

$$O_E(\theta_E) = \operatorname{argmin}_{\theta_E} Loss(\tilde{I})$$

**Asymmetric.** In the asymmetric scheme, public-key generator serves for Alice and Bob and its objective is consistent with Alice and Bob's. Therefore, during the updating of parameters  $\theta_A, \theta_B$  for Alice and Bob,  $\theta_K$  is updated at the same time. The loss function and optimal value of each agent in this asymmetric scheme are defined in a similar manner as those in the symmetric steganography:

$$O_{ABK}(\theta_A, \theta_B, \theta_K) = \operatorname{argmin}_{\theta_A, \theta_B, \theta_K} (L_{AB}(\theta_A, \theta_B))$$

$$O_E(\theta_E) = \operatorname{argmin}_{\theta_E} Loss(\tilde{I})$$

## IV. EXPERIMENT

As a proof-of-concept, we implement our steganographic training framework for symmetric steganography as well as asymmetric steganography. In this section, we first explain the dataset and primary hyperparameters, then we display the training results.

### A. Implementation

**Datasets and Hyper parameters.** We started our experiment with ImageNet [15] Dataset. It is a large-scale, accurate and diverse image database built upon the hierarchical structure provided by WordNet, consisting of a total of 3.2 million cleanly annotated images. In the experiment, all models are trained on 80,000 cover images from the ImageNet [15] training set, resized to experiment-specific dimensions  $3 \times 128 \times 128$ . Evaluation is performed on a 10,000 image test set unseen during training. Message and key are generated with each bit drawn uniformly at random. Moreover, we introduce a parameter bitrate referring to the number of bits to be concealed within per pixel of containers (bpp). We embed binary messages of length  $L = 1024$  or  $L = 8192$  into  $128 \times$

128 pixels images, that is to say, the bitrate is 0.06 bpp or 0.5 bpp. The keys, not only secret key in each scheme but also public key generated in asymmetric steganography, are fixed to be 1024 bits in length. We adopted the learning rate = 0.001 for gradient descent, and performed grid search to find the optimum hyperparameters  $\alpha = 0.5, \beta = 0.5, \gamma = 0.03, \delta = 0.1$ . We trained our model for 100 epochs with batch size 64.

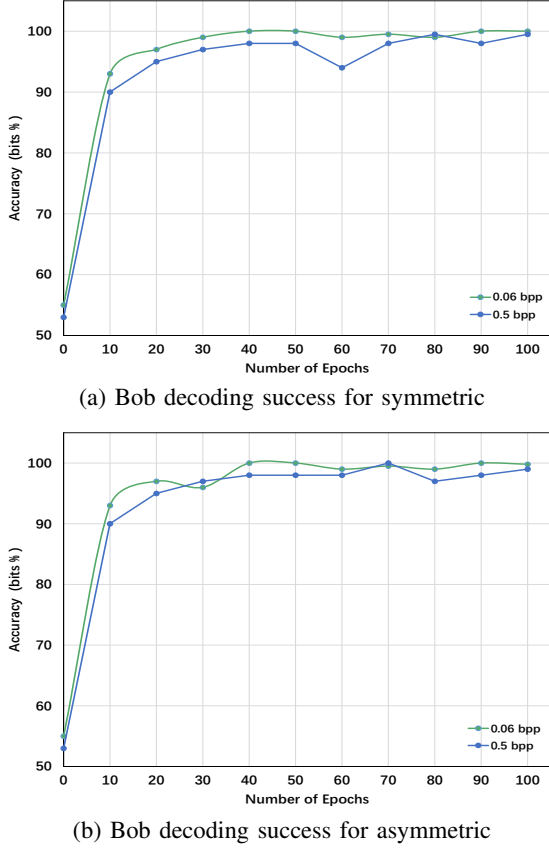


Fig. 4. Results of message reconstruction accuracy on ImageNet dataset

## B. Results

The training of DeepKeyStego has been successfully implemented in our experiments for both symmetric steganography and asymmetric steganography

In Fig. 4, it can be seen that the accuracy of correctly decoded bits from either the symmetric or the asymmetric decoder Bob with the bitrate of 0.06 bpp, climbs faster than that from decoders with that of 0.5 bpp. At the end of our training process, both of the symmetric and the asymmetric decoders with the bitrate of 0.06 bpp achieved 100% accuracy, while the accuracy of the decoders with a bitrate of 0.5 bpp was more than 99%.

## V. EVALUATION

Confidentiality and integrity are two of the most crucial properties of secure communication. Confidentiality is a set



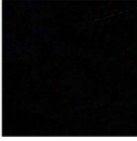
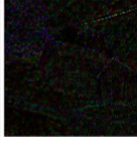



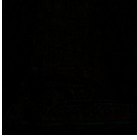
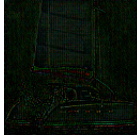

of rules to prevent sensitive information from being accessed by unauthorized entities, and integrity is the guarantee of reliable and accurate transmission of data. In this section, we evaluate the symmetric and asymmetric steganography trained with DeepKeyStego on these two security properties.

In the context of image steganography, the confidentiality of communication is faced with two major kinds of threats: (i) steganalyzers attempt to detect the presence of sensitive information, i.e. to distinguish stego images from innocent ones; (ii) malicious eavesdroppers try to recover the content of embedded information from any received images or known stego images. Hence we evaluate the confidentiality of trained steganography from three perspectives: invisibility, the alteration made to cover images; undetectability, the stego images can evade steganalysis detection; unrecoverability, the degree to which attackers can recover the content of secret messages. Besides, we evaluate integrity by analyzing the performance of the decoder on reconstructing sensitive information from stego images in a practical situation.

**Invisibility.** Before evaluating our trained steganographic framework against current steganalysis methods, we discuss the alterations made to cover images first. We assume that the more similar a stego image with sensitive information is with its original image in appearance, the more undetectable it is against steganalysis. The alteration introduced by steganographic encoding can be intuitively observed with the aid of difference image by subtracting the original image from its stego image, in which the values of pixels imply the distortions made to cover images and a black pixel (0, 0, 0) in the difference image implies that the corresponding pixels in the stego image and cover image are totally the same. In Table I, there show two example images that have been embedded with randomly generated messages by symmetric and asymmetric steganography with a bitrate of 0.5 bpp. As can be observed, the generated stego images are visually indistinguishable from corresponding cover images, and the differences between them are too tiny to be seen by humans. After being magnified by 5 times, the revealed differences illustrate that the distributions of distortions introduced by our trained steganographic schemes appear to be related to the natures of cover images like complexity and texture.

Since classical steganographic methods are based on exact rules and their alterations to images are traceable, we only compare our trained schemes with a scheme trained in the manner of ste-GAN-ography that trains steganographic algorithms with deep neural networks as well. The alterations made to cover images are quantitatively measured by three criteria: MSE, SSIM, and PSNR. As seen in Fig. 5, the lower MSE, and higher PSNR demonstrate that each of our schemes produced smaller alterations to cover images than the one trained by ste-GAN-ography. Further, the higher SSIM also implies that our generated images are more similar to original images in structure. Since all steganographic schemes involved in this comparison are neural networks iteratively trained with an adversarial steganalyzer, such discrepancy in image alteration between our work and ste-GAN-ography mainly comes from

TABLE I  
Visual effects of resultant steganographic encoders.

| Cover   | Symmetric(stego / difference $\times 1$ / difference $\times 5$ )                 |   | Asymmetric(stego / difference $\times 1$ / difference $\times 5$ )                |  |
|---|---|---|---|--|
|  |  |  |  |  |
|  |  |  |  |  |

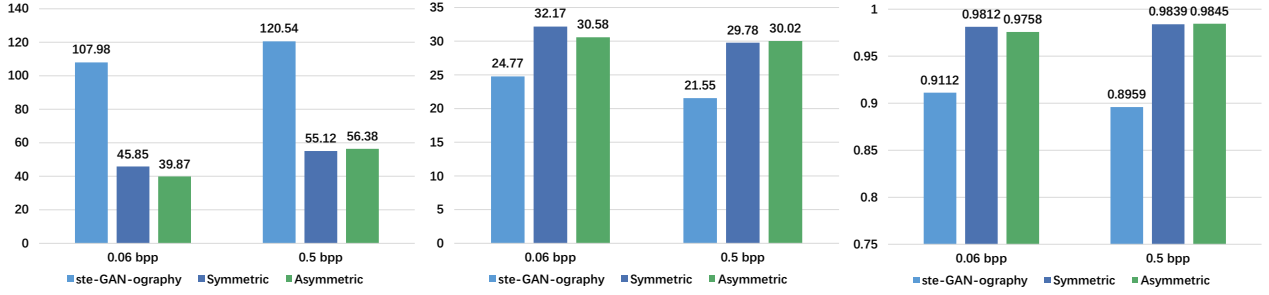


Fig. 5. Results of evaluation on MSE (left), PSNR (middle) and SSIM (right). The evaluation uses 200 cover images and 200 corresponding steganographic images to calculate the average value of each metric.

our novel network structures.

**Undetectability.** The goal of evaluating the undetectability of steganography trained by DeepKeyStego is to measure whether the steganographic images can be accurately identified or detected by attackers. We compare against various classical methods: S-UNIWARD, WOW, and HUGO, which are proposed for the spatial domain of images. Since the ImageNet dataset is constructed using compressed images based on the frequency domain. To fairly compare the undetectability of diverse methods, we evaluate them on gray-scale images of BOSS dataset [27]. The gray-scale images are commonly tested for classical methods. As depicted in Fig. 6, the resultant images generated by S-UNIWARD and DeepKeyStego are visually indistinguishable from the cover image.

We compare our results against a strong steganalyzer, ATS [16]. ATS repeatedly applies a steganographic algorithm to the label-free dataset (500 cover images), generating an eponymous dataset (500 stego images). Then we train a supervised classifier that distinguishes stego images from cover images by utilizing these images as the training set (500 cover and 500 stego images). In order to repeatedly apply steganography algorithm to build the artificial training set, we consider such an assumption that the attackers know the dataset, hyperparameters and exact architecture of DeepKeyStego, but cannot access precise network weights. We first train a DeepKeyStego model with a random seed for the ATS to build the artificial

training set. Then, we use the new random initialization to train another model for evaluation. As depicted in Table II, when we test ATS on the second model, the detection accuracy is about 50%, whether it is symmetric or asymmetric setting, which is a completely random guess. In contrast, ATS achieves detection accuracy of 75% for WOW and more than 80% for HUGO and S-UNIWARD. The detection rate indicates an advantage of learning steganographic algorithms: unlike static classical algorithms, it is undetectable by ATS if the attackers can't access the weights of DeepKeyStego directly. The reason behind this is that the distortion mode made upon the cover image by each DeepKeyStego is unique.

The above obviously indicates that our resultant algorithms have outstanding performance better than classical steganographic methods. Furthermore, in order to further test the undetectability of our steganographic images, we use another popular open-source steganalyzer—StegExpose. It combines Primary Sets [17], ChiSquared Attack [18], RS Analysis [19], and Sample Pairs [20], all of which are existing steganographic analysis techniques. We utilized StegExpose to inspect four image pools, each pool consisting of 2000 clean images randomly selected from the test set, and 2000 stego images separately generated with symmetric or asymmetric steganography with both embedding rates of 0.06 bpp or 0.5 bpp. As depicted in Fig. 7, the ROC curve of our DeepKeyStego demonstrate that the performance of StegExpose is close to



TABLE II  
Accuracy of distinguishing between cover and stego images for the steganalyzers ATS

| Method                    | Bits per pixel | Detection Rate (ATS) (%) |
|---------------------------|----------------|--------------------------|
| S-UNIWARD                 | 0.5            | 83%                      |
| WOW                       | 0.5            | 75%                      |
| HUGO                      | 0.5            | 85%                      |
| DeepKeyStego (symmetric)  | 0.5            | <b>51%</b>               |
| DeepKeyStego (asymmetric) | 0.5            | <b>50%</b>               |

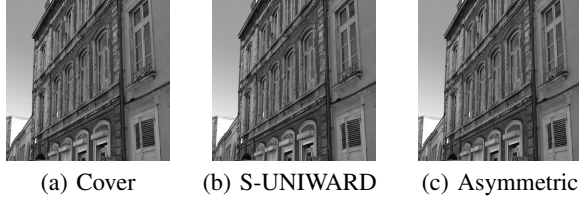


Fig. 6. Example of steganographic images for BOSS

random guessing with an auROC curve of 0.57 on average. The results convincingly show that the stego images generated by DeepKeyStego can successfully defend multiple steganalysis techniques. Our model meets the minimum requirements as a feasible steganographic technique.

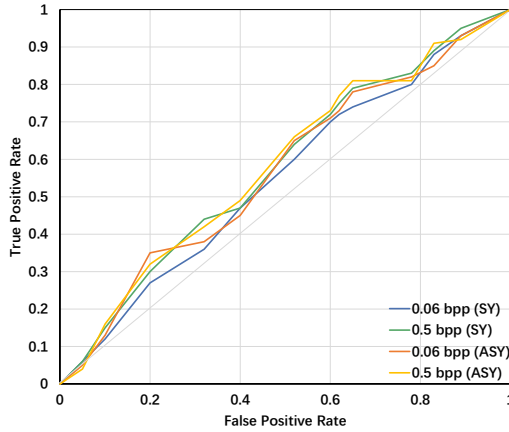


Fig. 7. The receiver operating characteristic (ROC) curve produced by the StegExpose library.

**Unrecoverability.** The above evaluations demonstrate that our steganographic algorithms based on deep learning make fewer alterations to cover images and can better protect the presence of the embedded messages against steganalysis than existing steganographic methods. However, an attacker may try to recover the message from every eavesdropped image or may already be informed of the presence of secret messages. The encoding algorithms in traditional steganography are easy to reverse, if an attacker knows the algorithm used for a given stego image, the embedded message can be recovered consequently. For steganography trained by deep learning, like

ste-GAN-ography, where the algorithms are implemented by deep neuron network and are scarcely possible to be reversed, the access to the decoder can extremely increase the feasibility of such attacks.

Nowadays, it is a non-trivial task to build a deep learning model, especially a production-level model. We need to utilize human expertise, powerful computing resources, and large-scale datasets to train a high-performance model. For most of the common users like individuals and small enterprises, it is unmanageable to train their own steganographic models, so that different users may use the same encoding and decoding models pre-trained by a third-party MLaaS (machine learning as a service) provider. In other words, there may be other users legally possessing the same decoding model besides the target receiver. For the steganography trained in the manner of ste-GAN-ography, if the stego images are transmitted through a communication channel accessible for any malicious eavesdropper that also has the decoding model, the content of embedded messages is at risk of revealing. It is the same for the situation that the decoder model is stolen by or leaked to an attacker [22]–[26]. To address this threat, we introduce secret keys to the steganography trained by deep learning. Secret keys are randomly or designedly generated for decoding messages into images. Even if an attacker has access to the decoder model, the messages hidden in stego images can never be recovered without corresponding secret keys. For an example shown in Table III (bottom line), the accuracy of recovering a binary message without the secret key is 50.71% that seems to be a random guess.

**Integrity.** To evaluate the integrity of the steganography trained with DeepKeyStego, we have simulated a practical situation where the communication channel between the message sender and receiver is publicly accessible. We had three machines playing the roles of the sender (Machine A), receiver (Machine B) and eavesdropper (Machine C) separately, where Machine C has been a negative attacker who only received images from the channel and never sent anything. We trained asymmetric steganography to avoid the transmission of secret keys and set the bitrate as 0.6 bpp. Machine A was assigned the encoder model, Machine B was assigned the decoder model as well as the pK generator model, and Machine C had the same decoder model as Machine B.

As shown in Table III, a 128-character message was converted to 1024 binary digits and Machine A encoded it into a cover image with a public key. The public key was generated by Machine B from a secret key and was transmitted to Machine A, which was also available to Machine C. The stego image generated by Machine A was sent to both Machine B and Machine C. All transmissions in this simulation were implemented via E-mail. As a result of respectively decoding the received stego image by Machine B and Machine C, Machine B has successfully recovered 99.8% of the binary message, and without the corresponding secret key, Machine C failed in recovering with an accuracy of 50.71%. The binary message reconstructed by Machine B can be converted a readable English sentence with negligible typos, while the binary

TABLE III  
An example in simulated communication

| Secret Message  | Before encode   | Secret Key | After decode      | Decoded Message   |
|---|---|------------|-------------------|---|
| Two months ago, across an assembly-room table in a factory in Jacksonville, Fla., President Barack Obama was talking to me about... | 01010100 01110111<br>...<br>01110100 01100001<br>...<br>01110101 01110100 | Correct    | 01010100 01110111 | Two months ago, across an assembly-room table in a factory in Jacksonville, Fla., President Barack Obama was talking to me about... |
|   |   |            | ...               |   |
|   |   |            | 01110100 01100011 |   |
|   |   |            | ...               |   |
|   |   | Wrong      | 10111111 10101111 | Two months ago, across an assembly-room table in a factory in Jacksonville, Fla., President Barack Obama was talking to me about... |
|   |   |            | ...               |   |
|   |   |            | 11010100 10101001 |   |
|   |   |            | ...               |   |
|   |   |            | 00111010 1100101  |   |

message reconstructed by Machine C can only be converted to a ridiculous gibberish. It states that our trained steganography has enough integrity for effective communication in a practical situation.

## VI. CONCLUSION

In this paper, we first introduced the keys to the steganography based on deep networks and proposed the DeepKeyStego, a novel key-dependent steganographic framework that achieves steganographic objectives with adversarial training. Symmetric (secret-key) and Asymmetric (public-key) steganographic scheme are separately proposed and each scheme is successfully designed and implemented. We showed that based on deep convolutional networks and novel network structures, DeepKeyStego achieved excellent invisibility and outstanding undetectability better than previous methods. Furthermore, we discussed and analyzed the importance of key usage in the method based on deep learning model, and convincingly showed that the embedded information will never be recovered without secret keys. Finally, we simulated our scheme in a real situation, which indicates that our scheme is effective and practical for information hiding in communication.

For future work, we expect this work to be expanded to other cover media such as audio and video and expect it to be implemented on other forms of deep learning e.g. recurrent neural networks, etc.

## REFERENCES

- [1] Holub, V., Fridrich, J. J. (2012). Designing steganographic distortion using directional filters. IEEE International Workshop on Information Forensics Security. IEEE.
- [2] Tom Pevn, Tom Filler, Bas, P. (2010). Using High-Dimensional Image Models to Perform Highly Undetectable Steganography. Information Hiding. Springer Berlin Heidelberg.
- [3] Vojtech Holub, Fridrich, J., Tom Denemark. (2014). Universal distortion function for steganography in an arbitrary domain. EURASIP Journal on Information Security, 1(1), 1.
- [4] Hayes, J., Danezis, G. (2017). Generating Steganographic Images via Adversarial Training. NIPS 2017: Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, California, USA.
- [5] O'Sullivan, J. A., Moulin, P., Ettinger, J. M. (1998). Information theoretic analysis of steganography. IEEE International Symposium on Information Theory. IEEE.
- [6] Ahn, L. V., Hopper, N. J. (2004). Public-Key Steganography. Advances in Cryptology - EUROCRYPT 2004. Springer Berlin Heidelberg.
- [7] Mielikainen, J. (2006). Lsb matching revisited. IEEE Signal Processing Letters, 13(5), 285-287.
- [8] Zhang, J., Dan, Z. (2009). Detection of lsb matching steganography in decompressed images. IEEE Signal Processing Letters, 17(2), 141-144.
- [9] Qin, J., Xiang, X., Wang, M. X. (2010). A Review on Detection of LSB Matching Steganography. Information Technology Journal, 9(8), 1725-1738.
- [10] Goodfellow, I. J., Pougetabadi, J., Mirza, M., Xu, B., WardeFarley, D., Ozair, S., ... Bengio, Y. (2014). Generative Adversarial Nets. neural information processing systems.
- [11] Volkonskiy, D., Nazarov, I., Borisenko, B., Burnaev, E. (2017). Steganographic Generative Adversarial Networks. arXiv: Multimedia, Zhao, H., Gallo, O., Frosio, I., Kautz, J. (2017). Loss Functions for Image Restoration With Neural Networks. IEEE Transactions on Computational Imaging, 3(1), 47-57.
- [12] Wang, Z., Bovik, A. C., Sheikh, H. R., Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing, 13(4), 600-612.
- [13] APAKetkar, N. (2014). Stochastic gradient descent. Optimization.
- [14] Liu, Z., Luo, P., Wang, X., Tang, X. (2014). Deep learning face attributes in the wild.
- [15] Lerch-Hostalot, D., Megas, David. (2016). Unsupervised steganalysis based on artificial training sets. Engineering Applications of Artificial Intelligence, 50, 45-59.
- [16] Dumitrescu, S., Wu, X., Wang, Z. (2003). Detection of LSB steganography via sample pair analysis. IEEE Transactions on Signal Processing, 51(7), 1995-2007.
- [17] Westfeld, A., Pfitzmann, A. (1999). Attacks on Steganographic Systems. International Workshop on Information Hiding.
- [18] Fridrich, J., Goljan, M., Du, R. (2002). Reliable detection of lsb steganography in color and grayscale images. Mm Sec Proceedings of the Workshop on Multimedia Security New Challenges, 8, 22-28.
- [19] Dumitrescu, S., Wu, X., Memon, N. D. (2002). On steganalysis of random LSB embedding in continuous-tone images. international conference on image processing.
- [20] Zhang, L., Zhang, L., Mou, X., Zhang, D. (2012). A comprehensive evaluation of full reference image quality assessment algorithms. international conference on image processing.
- [21] Uchida, Y., Nagai, Y., Sakazawa, S., Satoh, S. (2017). Embedding Watermarks into Deep Neural Networks. international conference on multimedia retrieval, 269-277.
- [22] Rouhani, B. D., Chen, H., Koushanfar, F. (2018). Deepsigns: a generic watermarking framework for ip protection of deep learning models.
- [23] Zhang, J., Gu, Z., Jang, J., Wu, H., Stoecklin, M. P., Huang, H., et al. (2018). Protecting Intellectual Property of Deep Neural Networks with Watermarking. The (pp.159-172).
- [24] Adi, Y., Baum, C., Cisse, M., Pinkas, B., Keshet, J. (2018). Turning your weakness into a strength: watermarking deep neural networks by backdooring.
- [25] Guo, J., Potkonjak, M. (2018). Watermarking deep neural networks for embedded systems. international conference on computer aided design.
- [26] Bas, P., Filler, T., Pevn, T. (2011). Break Our Steganographic System: The Ins and Outs of Organizing BOSS. International Conference on Information Hiding.