Neurocomputing 585 (2024) 127593

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Proactive image manipulation detection via deep semi-fragile watermark

Yuan Zhao [a,b], Bo Liu [a,b,*], Tianqing Zhu [e], Ming Ding [c], Xin Yu [d], Wanlei Zhou [e]

[a] Centre for Cyber Security and Privacy, University of Technology Sydney (UTS), Ultimo, Sdyney, 2134, NSW, Australia
[b] The School of Computer Science, University of Technology Sydney (UTS), Ultimo, Sdyney, 2134, NSW, Australia
[c] CSIRO Data61, Eveleigh, Sdyney, 2015, NSW, Australia
[d] School of Information Technology and Electrical Engineering, University of Queensland (UQ), St Lucia, Brisbane, 4072, QLD, Australia
[e] Faculty of Data Science, City University of Macau (CityU), Taipa, Macau, 999078, Australia

ARTICLE INFO

Communicated by J. Chen

Keywords:
Semi-fragile watermark
Invisible watermark
Image tampering detection
Manipulation localization

ABSTRACT

Malicious image tampering refers to intentionally manipulating images to make them harmful to the owners or users. It has become one of the most severe challenges to image authenticity. Conventional methods for detecting tampering by identifying visual artifacts and distortions have limitations due to the rapid advancement of image manipulation techniques, which leave fewer detectable traces. To address these challenges, we propose a proactive media authentication method using deep learning-based semi-fragile watermarks. The designed scheme utilizes deep neural networks to embed an invisible watermark into a target image that is pixel-by-pixel entangled with it, which acts as an indicator of tampering trails. Once the watermarked image is counterfeited, the embedded watermark will exhibit changes accordingly, so we can locate the tampered regions by comparing retrieved and original watermarks. This proactive authentication mechanism makes our method effective against various image tamper techniques, including image copy&move, splicing and in-painting. Although our watermark is designed to be fragile to malicious tampering operations, it remains robust to benign image-processing operations such as JPEG compression, scaling, saturation, contrast adjustments, etc. This design enables our watermark to retain effectiveness when shared over the internet. Extensive experiments demonstrate that our method achieves state-of-the-art forgery detection with superior robustness, imperceptibility and security performance.

1. Introduction

Digital images have become an essential medium for information transmission in our society. However, technical advancement makes tampering images imperceptibly, which can be exploited for malicious intentions, e.g., creating fake news and Internet rumors. Therefore, detecting the tampered regions in an image is essential to protect image authenticity.

State-of-the-art detection methods leverage deep learning techniques by distinguishing feature distribution inconsistency [1–3] or boundary discrepancy [4,5] in an image to identify the forgery or any manipulation. Those methods assume that image manipulation techniques may inevitably produce detectable artifacts in their outputs. For example, [3] detects forgery pixels by identifying local anomalous features in suspicious images. However, this prerequisite might lead to several inherent drawbacks. First, as image manipulation techniques progressively evolve, fake images exhibit less noticeable artifacts. As a result, detection methods developed to detect certain artifacts would fail with a high chance. Moreover, existing methods trained on seen tampering types might fail to detect unseen counterfeits. Besides, some methods [6–8] detect malicious tampering in a proactive style. Those methods embed invisible tags into images. Then, according to the extracted tag, they determine whether a suspicious image has been forged. However, these methods cannot pinpoint the tampered region in a forgery image.

To overcome these issues, we propose a proactive image authentication method based on semi-fragile watermarks generated through deep learning techniques. Our specialized watermark is robust to benign image post-processing operations while fragile to malicious tampering. By employing this watermark, our method can provide accurate and generalized tampering detection performance, not limited to a specific forgery or manipulation type. Moreover, it can pinpoint the tampered pixels rather than merely identify whether an image is a forgery.

The pipeline of our method involves converting a secret image into an invisible watermark, which we embed pixel-by-pixel into a cover

* Correspondence to: Building 11, University of Technology Sydney, 15 Broadway, Ultimo NSW 2007, Australia.
E-mail addresses: yuan.zhao@student.uts.edu.au (Y. Zhao), Bo.Liu@uts.edu.au (B. Liu), tqzhu@cityu.edu.mo (T. Zhu), Ming.Ding@data61.csiro.au (M. Ding), Xin.Yu@uq.edu.au (X. Yu), wlzhou@cityu.edu.mo (W. Zhou).

https://doi.org/10.1016/j.neucom.2024.127593
Received 28 October 2023; Received in revised form 10 January 2024; Accepted 19 March 2024
Available online 26 March 2024

image to create a watermarked image (known as the container image). The container image remains perceptually identical to the original cover image, allowing us to replace the original cover and use the container image in scenarios with a risk of malicious tampering. Any manipulation of the container image will inevitably affect the same region of the embedded watermark, causing damage to the secret image in that area. Consequently, we can precisely locate the tampered region by comparing the decoded secret image from the container image to the original secret image.

The designed framework consists of three modules to achieve the above function: a hiding network, an attack module and a revealing network. We adopt a cover-agnostic style hiding network that generates watermarks according to different secret images, independent of cover images. This design allows us to directly add the watermark to an arbitrary cover image to construct the container image, significantly improving our method's generalization. Besides, by minimizing the perceptual differences between cover and container images, the hiding network learns to encode secret images as invisible watermarks with remarkable imperceptibility.

In addition, to make our watermark fragile to malicious tampering approaches but robust to conventional image post-processing operations, we introduce an attack module in the training process. It consists of horizontally combined distortion and tamper layers that simulate tampering and post-processing manipulations. By applying these manipulations to container images, the attack module can strengthen the semi-fragility of our watermark. The attack module is only employed to co-adapt training with hiding and revealing networks and is not included in our method's inference step.

The revealing network aims to recover the secret image from the container image to locate the tampered regions within it. We thus train the revealing network using the masked secret image with the same tampered regions in the processed container image as the label. Consequently, when tampering occurs within the container image, the revealing network will only restore the untouched area instead of reconstructing the entire embedded secret image. This mechanism allows us to locate the tampered region accurately by comparing the original and recovered secret images.

Experiment results demonstrate that our designed scheme can achieve an average detection AUC of nearly 0.95 across a wide range of image manipulations. It turns the open-world image manipulation detection problem into a trivial watermark retrieval task, allowing for greater tamper detection accuracy. Our contributions are summarized as follows:

- We develop a novel deep learning-based semi-fragile image watermarking framework, which can serve as a proactive defense against malicious tampering.
- Our watermark achieves a balanced trade-off between detection performance and imperceptibility, so there is no influence on watermarked images' real-world usage.
- We have comprehensively evaluated the proposed method, analysing its performance across various aspects. We not only compared our method with SOTA detection methods on multiple datasets to detect various types of tampering to reflect its detection capability, but we also conducted a deep analysis of its watermark and assessed its performance in terms of robustness and security. Our experiment can serve as a template for similar research in the future.

2. Related work

Three main research areas are relevant to this work: digital watermarking, tamper detection and image hiding. In what follows, we give a brief description of each topic.

2.1. Digital watermarking

Digital watermarking involves embedding information into an image imperceptibly. This field primarily focuses on developing three different types of watermarks: fragile [9,10], robust [11–15] and semi-fragile [16–18] watermarks. Fragile and semi-fragile watermarks are primarily used to certify the integrity and authenticity of image data. Specifically, fragile watermarks are designed to achieve accurate digital media authentication, where even a one-bit change to an image will lead it to fail the certification system. In contrast, robust watermarks aim to withstand various image manipulations, allowing media producers to assert ownership over their content, regardless of redistribution and modification.

Semi-fragile watermarks are a hybrid approach, merging the advantages of both robust and fragile watermarks. They are mainly used for fuzzy authentication of digital images and identification of image tampering [18]. The rationale behind semi-fragile watermarks lies in the typical lossy nature of image and video transmission and storage, which should not disrupt the watermark. However, when the image gets tampered with, the watermark should also get damaged, indicating image tampering.

2.2. Image manipulation detection

Developing image editing techniques makes tampered images widely available and more realistic. Currently, the research community defines three common types of image tampering, which are: Copy-move (i.e., copying and moving elements from one region to another region in a given image), splicing (i.e., copying elements from one image and pasting them on another image), and inpainting (i.e., removal of unwanted elements). All these manipulations could lead to misinterpretation of the visual content.

Image manipulation detection aims at detecting and localizing these tamperings. At first, many previous studies in this field are based on hand-crafted features, such as local noise analysis [19], CFA artifacts [20], and illumination variance analysis [21]. With the success of deep learning techniques in various computer vision tasks, researchers try to bring deep neural networks to conduct image manipulation detection (e.g. CNNs [1] and GANs [22]). Specifically, Li et al. [2] propose to utilize an FCN's first convolutional layer with trainable high-pass filters to capture the tampering features for detection. Beyond filtering learning, Zhang et al. [23] employ a stacked autoencoder to learn context features to detect image manipulation. Bayar et al. [24] improve the detection performance by replacing the low-pass filter layer with an adaptive kernel layer to learn the filtering kernel used in tampered regions. Several methods combine hand-crafted features and learning features for image forensics. For instance, Wu et al. [3] and Hu et al. [1] use both BayarConv and SRM features as detection clues. Given features from distinct views, they develop a two-stream network, which inputs the RGB image and its feature counterpart generated by the SRM filter to identify the tampered pixels. MVSS-Net [25] replaces the non-trainable bilinear pooling used in previous works with Dual Attention. It further concatenates edge features to adaptively predict a tampered area and its boundary.

Meanwhile, some proactive measures [6–8] are fighting malicious tampering by embedding an invisible tag into the original image by the user, which can remain retrievable after the manipulation process, so the user can retrieve the tag and block the dissemination of fake information. However, these methods cannot pinpoint the tampered region.

2.3. Image hiding

Image hiding is an important research direction that attempts to conceal a whole image into another rather than simply hiding a binary message. According to the employed techniques, this research direction can be divided into traditional and deep image hiding.

Traditional image hiding methods rely on conventional image features, such as those in spatial and frequency domains. Least Significant Bit (LSB) [26] is the most traditional spatial domain-based method for concealing images. It replaces the n least significant bits of the cover image with the most significant n bits of the secret image. In addition to LSB, there are many methods proposed to embed information in frequency domains, such as the discrete Fourier transform (DFT) domain [27], discrete cosine transform (DCT) domain [28], and discrete wavelet transform (DWT) domain [29]. For instance, JSteg [30] conceals data within the LSBs of the host image's DCT coefficients. However, our experiments reveal significant limitations of traditional methods, including visible watermarks, image distortions, and low robustness against compression techniques like JPEG. Additionally, these methods are not optimized for image manipulation detection and localization.

More recently, deep learning techniques have been applied to image-hiding [31,32], which achieved impressive results. This kind of method is also referred to as deep image hiding. Baluja [33] is one of the first deep-learning solutions for concealing a whole RGB image within another. They adopt a preparation network to extract useful features of the secret image and then employ a hiding network to fuse the features of the secret image within the cover image. Finally, a revealing network is adopted to recover the original secret image. Based on [33], Rahim et al. [32] added a regular loss to ensure joint end-to-end training. However, both of them have the problem of color distortion. Zhang et al. [34] mitigated this impact by decreasing the payload of the secret image, i.e., only embedding the gray-scale image. Zhang et al. [35] propose a novel universal deep hiding (UDH) meta-architecture to disentangle the encoding of the secret image from the cover image, improving the generalization and interpretability of image hiding. Luo et al. [36] improve watermark robustness by introducing adversarial training and channel coding in their framework. Based on image texture features, [37] formulates adaptive payload distribution to achieve steganography of multiple images. HiNet [38] adopts an inverse learning mechanism combined with a novel low-frequency wavelet loss, achieving impressive high-quality image hiding.

The previous works demonstrate that deep image hiding has a significant advantage in preserving the container image's visual quality, inspiring us to employ this technique to protect the image's authentication while guaranteeing the target image's utility.

3. Methodology

This section explains how to implement the proposed method. Given an image of size $W \times H$, the goal of our method is not only to determine if the image has been tampered with but also to locate the altered regions. The main notations used in the rest of the paper are listed in Table 1.

3.1. Motivation and threat model

We begin by elucidating the motivation and threat model of our method. Fig. 1 presents the common threat model to image-sharing platforms like Facebook or Instagram. Attackers in this model aim to tamper images and spread the forgery to produce reputation losses for the victim or obtain benefits from the forgery. We assume they have the same access rights as the victim, enabling them to obtain the victim's posted images and share the tampered images on the same platform.

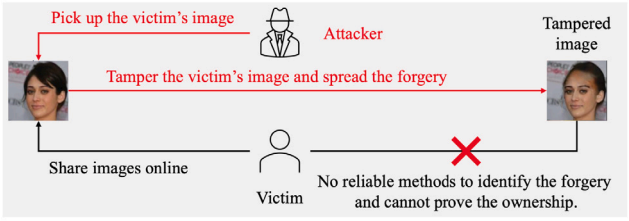
In this scenario, the victim shares their images without any reliable precautionary measure. Once the image is uploaded online, there is no mechanism to ensure its authenticity and integrity, rendering it vulnerable to malicious tampering attacks. The attackers can easily pick the victim's photos, manipulate them, and release the tampered results while falsely claiming them to be authentic. Such misinformation can lead to severe reputational damage for the victim and raise security and privacy concerns. Worse still, it is hard for the victim to disclose the forgery since there are no reliable third-party identification methods.

To address the above problems, we design a solution for proactively protecting the authenticity of images. We transform the secret image into a semi-fragile and invisible watermark and then embed it into the target image. This watermark is designed to be fragile to malicious manipulations or tampering, while simultaneously remaining robust to benign image-processing operations such as compression, scaling and color adjustment. In this way, our methodology enables the identification of tampered regions or pixels by comparing the recovered secret image from the container against the original secret image. This mechanism equips image owners with a reliable means of proactively protecting their images' authenticity and integrity before sharing them online.

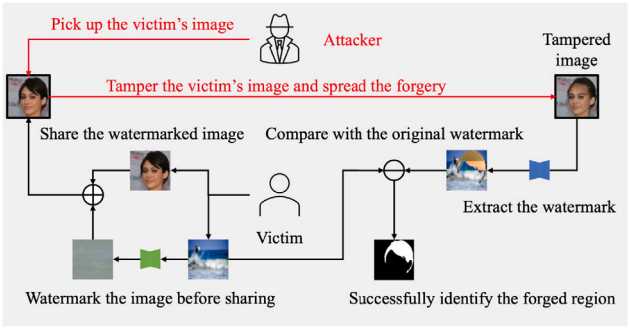
Additionally, our approach distinguishes itself from passive detection methods by utilizing the embedded watermark as detection clues instead of artifacts left by tampering operations. This feature makes our method agnostic to the evolution of image manipulation techniques. So, it has reliable performance when detecting unknown and novel tampering methods.

Table 1
Summary of notations in this paper.

Notation	Description
x_{secret}	Secret image: the image to be hidden, serving as the source of a watermark.
x_{cover}	Cover image: the image to hide the secret image, also the image we want to protect from tampering.
$x_{container}$	Container image: the image with x_{secret} embedded, which is exposed to potential tampering methods.
$x_{retrieved}$	Retrieved secret: recovered secret image from container, comparing it with x_{secret} can identify the tampered area.



(a) The platform without protection.



(b) The platform with our method's protection.

Fig. 1. One potential application scenario of our method is protecting users' images on some public platforms, such as Instagram. Our method can give users a reliable approach to verifying the pixel-level authenticity of their images.

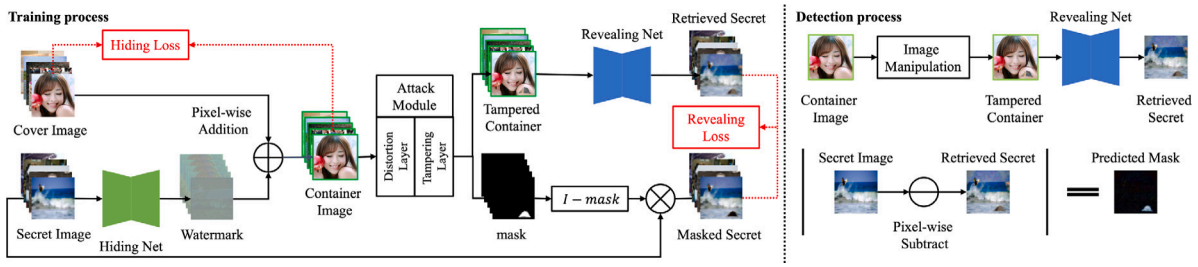


Fig. 2. Overview of the proposed framework. The black arrows refer to data flows, and the red dashed lines show the loss flows. Our framework comprises three modules: hiding network, attack module and revealing network. The hiding network generates a watermark from the secret image, which is embedded in the cover image to produce the container image. The attack module manipulates the container image to create a processed image and corresponding mask. Finally, the revealing network retrieves the secret image from the processed container and determines the tampered region by comparing it with the original secret image. As illustrated in the detection process (right side), we can thus pinpoint the tampered region (predicted mask) by pixel-to-pixel comparing the retrieved and original secret images.

The bottom panel of Fig. 1(b) illustrates how to deploy our method in the real world. Before sharing images online, users can use our method to embed the personalized, human-invisible watermark into their images. These watermarked images are virtually indistinguishable from the originals, thereby having a negligible impact on their visual quality and utility. Once the watermarked image has been tampered with maliciously, users can employ our method to verify the tampered region and declare the forgery.

3.2. Network architecture

As outlined in the motivation section, the objective of our networks is to convert secret image as semi-fragile and invisible watermark, which is fragile to malicious tampering but withstand conventional image processing.

To this end, our networks comprise three distinct modules, as depicted in Fig. 2: (1) Hiding network: This module transfers the secret image as the human-invisible watermark. (2) Attack module: The attack module perturbs the container image that co-adapts with the training of hiding and revealing networks to strengthen our watermark's robustness and improve our method's tamper detection accuracy. (3) Revealing networks: This module recovers the secret image from the container and allows it to be compared with the original secret for identifying the tampered regions.

3.3. Hiding network

Previous learning-based image hiding techniques [31,35] utilize the frequency discrepancy between the cover image and watermark to achieve effective hiding that is invisible to humans. Drawing inspiration from these methods, we adopt the same U-Net style architecture with full-convolution layers as the backbone for our hiding network. This architecture has performed excellently in extracting the secret image's representative features and encoding them into the high-frequency (HF) domain to generate the corresponding watermark. Unlike nature images, which mainly consist of low-frequency (LF) information, the HF watermark is invisible to human observers.

Consequently, embedding it in the cover image will not produce virtually noticeable alternations in the corresponding result, i.e., the container image in our context. Besides, the pixel-wise addition establishes the simple but effective one-to-one corresponding relationship between the watermark and the container image pixels. As a result, pixel changes in the container image directly affect the corresponding pixel in the embedded watermark, which will be further exhibited in the recovered secret image. Additionally, our watermarks are directly embedded in the pixel values of the cover images, eliminating the need for special handling of various image formats. This feature significantly broadens the applicability of our approach. The full-convolutional architecture also enables us to adjust the number of layers of the hiding network to produce the watermark with the same resolution size and the number of channels to fit different cover images.

Moreover, ensuring that the embedded watermark is semantically independent of the container image is crucial in achieving practical pixel-level tamper detection. An adversary may exploit the semantic information in the container image to hide tampering traces by modifying the container image according to its content. However, it is not possible to hide the variation caused by the tampering process in the embedded watermark when the watermark is semantically independent of the container image. Therefore, we adopt the cover-agnostic framework proposed by [35], which enables us to generate the watermark only with secret image input without any information from the cover image. This approach allows our hiding network to embed any secret image into any cover image without re-training. As a result, even when advanced tampering operations can generate visually realistic outputs, the tampered region remains detectable from the secret image, enabling practical pixel-level tampering detection.

3.4. Attack module

Our semi-fragile watermark should be robust against various image post-processing and be sensitive to tampering operations. Prior research [39] explicitly incorporates benign and malicious image transformation functions in their training pipeline to achieve selective fragility and robustness of the watermark. However, this approach requires applying diverse differentiable image transformations to watermarked images during training, which is hard to exhaustive, leading to a lack of generalization to unseen manipulations.

To overcome this limitation, we utilize the observation that benign manipulation typically involves global alterations to images, such as Gaussian blurring or JPEG compression. In contrast, malicious tampering often targets only specific parts of the images. Leveraging this distinction, we have developed an innovative attack module. This module consists of a horizontally combined distortion layer and tampering layer, which applies global and local manipulation on container images to achieve selective fragility and robustness of the watermark.

Distortion layer. According to previous research [40], inserting a distortion simulation layer after the watermarked image can effectively improve the pipeline robustness against common image distortions. Therefore, inspired by prior works [31,35], we designed a distortion layer to apply various distortions on container images. By co-adapting the training of our networks with these relevant distortions, we can improve our watermarks' robustness against these distortions.

Temper simulation layer. To improve the tamper detection accuracy, we also designed a tamper layer that imitates tampering operations on the container images. This design represents an advancement over previous methods by avoiding the inclusion of specific tamper manipulations in the training pipeline, which could potentially limit the generalizability against different tamper operations. Instead, it randomly selects and modifies a region in the container image to

replicate the effect of tampering, subsequently using this modification as a ground-truth label during training.

The rationale of the tampering layer is that it can generate a tampered container image with a corresponding mask, allowing us to introduce simple tampering operations into our network training procedures. By enforcing the retrieved secret image with the same tampered region as the container image, the hiding network learns to embed the secret image sensitive to the container image's pixel variation. Similarly, the revealing network learns to extract the secret image with the same variation region based on changes in the container image rather than simply reconstructing the entire secret image. We can thus pinpoint the tampered region in the container image by comparing the original secret image with the retrieved secret image.

In conclusion, our attack module, comprising the distortion and tampering layers, is critical to our deep hiding methodology. It is integrated between the hiding and revealing networks during training but is excluded during the inference step. A comprehensive analysis of the attack module's effectiveness is presented in Section 4.2, demonstrating our success in achieving semi-fragile watermarking through this innovative approach.

3.5. Revealing network

As explained in the preceding section, the hidden watermarks and cover images typically occupy different frequency domains within container images, with the watermark mainly residing in the HF and the original cover image in the LF. From the perspective of the embedded watermark, the information of the cover image can be perceived as a frequency disturbance. The revealing network thus aims to retrieve the watermark under this disturbance and recover it as the secret image.

To this end, we employ a convolutional framework with residual connect as the backbone of our revealing network, which functions well when the inputs and outputs are distinct [41]. We jointly train the revealing network with the hiding network to pay more attention to the high-frequency spectrum of the embedded watermark. Such a design can significantly limit the impact of the disturbance of the cover image on the watermark retrieval and secret image recovery, resulting in superior performance in both concealing and revealing.

In addition, different from the conventional deep hiding methods aiming to reconstruct the secret image from the container image as high-fidelity as possible, our revealing process aims to recover the secret image with the same tampered region as in the container image. Therefore, we use the tampering layer masks to mask the secret image during training as the recovery label for the revealing network instead of the original secret image.

Retrieving the secret image rather than the original image from the container image for tamper detection streamlines and enhances the practicality of our method. If we choose to use the original image as a reference for locating tampered regions, it would necessitate storing every original image for future comparison. This approach is not only costly but also impractical for a large-scale application. Moreover, this method poses a reliability issue in the authentication process. A tamperer could falsely claim that their forged image is authentic, undermining the credibility and effectiveness of the authentication.

Another critical consideration is the nature of the watermark embedding. If we replace the secret image with the original image and embed it as a watermark pixel-by-pixel into the container image, any damage to the container image would be mirrored in the retrieved original image. Consequently, this would render it ineffective for identifying the tampered regions. Lastly, suppose we embed the original image as a watermark in some container image area and retrieve it for tampering location. In that case, it might be risky that this area be damaged during the tampering, making us fail to retrieve the embedded image.

Therefore, employing a secret image as a watermark is a more practical and reliable approach in our context. This strategy balances the dual objectives of effective tamper detection and efficient, secure watermark embedding.

3.6. Loss functions

To minimize the difference between the cover and container image, and enforce the retrieved secret to reflect the container's tampered pixels accurately, we adopt the following losses:

Hiding secret loss. We define a simple but effective pixel similarity loss function between the cover and container image to optimize the hiding network to achieve indistinguishable image hiding:

$$\mathcal{L}_{Hiding} = \|x_{container} - x_{cover}\|_2, \tag{1}$$

where the $\mathcal{L}_{Hiding}$ adopts l_2-norm.

Revealing secret loss. Given the tampered container image from the attack module, we train the revealing network using the same loss function as the Hiding loss. Nevertheless, to ensure that the retrieved secret image accurately reflects the tampered area in the container image, we do not simply reconstruct the entire secret image. Instead, we use the masked secret image as the ground truth label to train the revealing network to recover the secret image with the same tampered region applied to the container image. This function is defined as follows:

$$\mathcal{L}_{Revealing} = \|x_{secret} \times M - x_{retrieved}\|_2, \tag{2}$$

where $\mathcal{L}_{Revealing}$ also adopts l_2 norm, and M is the absolute residual between the mask from the tampering layers and identity matrix, i.e., $|I - mask|$.

Total loss function. The total loss function $\mathcal{L}_{Total}$ is a weighted sum of $\mathcal{L}_{Hiding}$ and $\mathcal{L}_{Revealing}$, as follows:

$$\mathcal{L}_{Total} = \lambda_H \mathcal{L}_{Hiding} + \lambda_R \mathcal{L}_{Revealing}, \tag{3}$$

where λ_H and λ_R are weights used to balance different loss terms.

It should be noted that all losses can adopt the l_1 norm or a combination of different norms. However, our validation results have shown that the choice of the norm does not significantly impact our method's performance. Therefore, we have adopted the l_2 norm for uniformity and convenience.

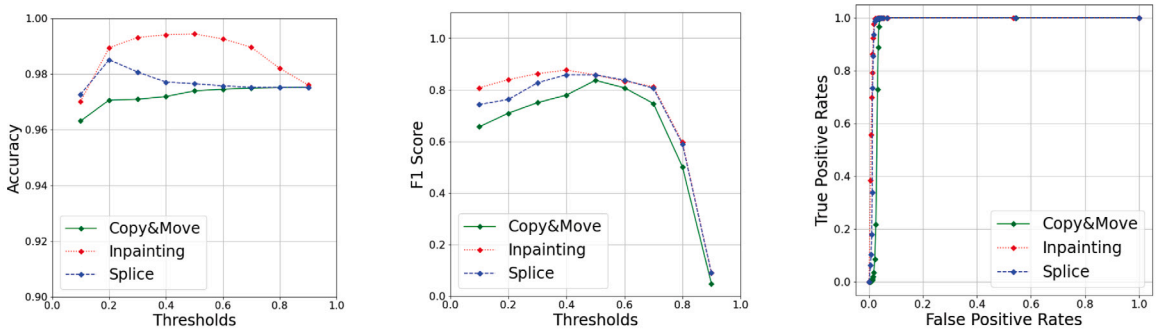
4. Experiments

We conduct extensive experiments to evaluate our method's performance from the following aspects: (1) impact of different components and decision thresholds; (2) effectiveness in image tamper detection; (3) robustness against conventional image post-processing; (4) imperceptibility of the embedded watermark; (5) security under threats and countermeasures. Furthermore, we implemented detailed analyses of the watermark that we embedded.

4.1. Experimental setup

Dataset. We train our networks on the widely used face image dataset FFHQ [42] and test its performance on other datasets. The gap between training and test datasets can validate our method's generalization. Note that choosing the FFHQ dataset as the training set is not based on considering performance differences.

Evaluation Metrics. To evaluate the performance of our method, we employed several evaluation metrics for each aspect of our experiments. Peak-Signal-to-Noise Ratio **(PSNR)** and Structural Similarity Index Measure **(SSIM)** are used to measure the similarity between the watermarked (container) images and original (cover) images to reflect the imperceptibility of the watermark and fidelity of the watermarked (container) images. We also calculate the Area Under the receiver operating characteristic curve **(AUC)** as the primary evaluation metric to reflect the image tamper detection performance.



(a) Accuracy (b) F1 score (c) ROC

Fig. 3. The tampering detection Accuracy, F1 Score and ROC of our method under different decision thresholds.

4.2. Ablation study

We first perform ablation studies to evaluate the impact of distinct components and decision thresholds on the performance of our method.

Impact of decision thresholds. Our method classifies the genuine and forged pixels based on the disparity between the recovered and original secret images. Thus, the decision threshold for our method is the threshold value that determines whether two pixels are dissimilar. To assess the impact of different decision thresholds on our method, we embed the same secret image into 1k Celeba-HQ [43] images using our method. Next, we apply three typical tampering operations, i.e., copy-move, inpainting and splice, to generate the corresponding test sets. Then, we employ our method to detect the tampered regions in each set while varying the decision threshold from 0.1 to 0.9 with a step size of 0.1. We calculate the corresponding accuracy and F1 score according to the detection results and plot the ROC. We do not calculate AUC values here because it is a threshold-irrelevant metric that cannot reflect the fluctuations in detection performance with decision thresholds varying. The results are presented in Fig. 3.

The results in Fig. 3(a) show high detection accuracy across different sets, even under extremely low or high thresholds. This abnormal phenomenon is due to the imbalanced data problem between forgery and authentic pixels in each image, which rendering accuracy cannot distinguish the performance of our method under varying decision thresholds. Nevertheless, according to the trend of F1 Score curves, we can find that our method exhibits superior detection ability when the threshold is below 0.7, but beyond this threshold, the performance decreases significantly. This phenomenon occurs because a high threshold ignores correctly identified tampered pixels, leading to poor detection results. The ROC presented in Fig. 3(c) proves the outstanding detection capability of our method. All three curves are close to the top left, and the corresponding AUCs are higher than 0.95.

Visualized predicted masks in Fig. 4 also verify the inference in the above quantitative evaluation. The predicted tampered areas are smaller than the ground truth when decision thresholds are higher than 0.6, while predicted tampered areas with lower thresholds tend to produce false alarms on authentic pixels. We can safely conclude that our method can effectively classify genuine and forgery pixels under the appropriate decision thresholds. Unless otherwise specified, we will set the decision threshold to 0.5 in the following experiments.

Impact of different components. To assess the influence of individual components of our method, we evaluate the designed scheme's performance under various configurations. All configurations are trained on the complete FFHQ dataset and tested on a random sample of 1k CelebaHQ images. For the sake of simplicity, we compute the average pixel-level detection AUC values of Copy-Move, Inpainting and Splice. Additionally, we employ three commonly used image post-processing methods, namely Blur (with kernel sigma 1.0), JPEG (with compression factor 70) and Crop (random crop 70% image), to attack the container images of each configuration. We calculate the detection

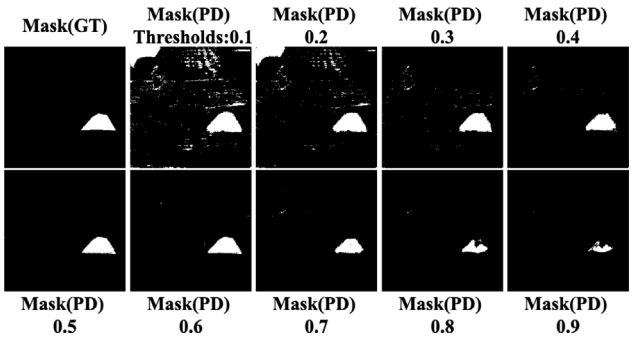


Fig. 4. Samples of pixel-level detection results of our method in varied decision thresholds. Mask (GT) represents the ground truth of tampered masks. Mask (PD) represents the predicted tampered results, and the values under Mask (PD) are the corresponding decision threshold.

AUC on these images to gauge each configuration's robustness against post-processing. The results are summarized in Table 2, while a more detailed analysis is presented below.

(1) Influence of different secret image formats. As stated in the methodology section, the secret image used in our method is independent of the cover images. Thus, we investigate the impact of different formats of secret images on our method's performance. We select images from four common types of images as secret images, including RGB, grayscale, QR code, and Gaussian noise. Different secret images were embedded into the same 1k CelebaHQ images using the same networks and trained models, resulting in four distinct groups. Next, we evaluated the performance of each group to determine if there were any differences in the method's performance.

The evaluation results for the various secret image formats are presented in the last four rows of Table 2 and 2nd to 5th columns in Fig. 5. The metrics values for each group are similar, with a maximum difference of 10%, while the qualitative results are almost identical. These findings suggest that the choice of different secret image formats does not significantly impact the performance of our proposed method. This characteristic is particularly practical because users of our method can freely choose any image as their secret image making it challenging for potential adversaries to obtain users' secret images falsely, thus increasing the security of our method. In the following experiment, we will use RGB images as the secret image for simplicity and uniformity.

(2) Influence of attack module. Finally, we use different components combinations in the attack module to reveal their effect (1st to 3rd rows in Table 2 and last three columns in Fig. 5).

Without equipping the distortion layers, networks achieve high detection AUCs on no-distorted container images in the first and fourth rows. However, when applying image post-processing methods to the container images, the networks experience a significant performance

Table 2
Ablation studies for different model designs.

Secret	Attack module			Effectiveness & Robustness				Stealthiness	
Image	None	Distortion	Tampering	AUC ↑	AUC(Blur1.0) ↑	AUC(JPEG70) ↑	AUC(RC0.7) ↑	SSIM ↑	PSNR ↑
RGB	✓			0.9368	0.5658	0.6111	0.9360	**0.95**	**39.65**
		✓		0.9397	0.7410	0.9095	0.9285	0.93	38.01
			✓	0.9755	0.6092	0.6992	0.9745	0.93	37.79
		✓	✓	**0.9793**	**0.7340**	**0.9650**	**0.9750**	0.94	38.05
Gray		✓	✓	0.9689	0.7213	0.9342	0.9815	0.94	38.01
QR code		✓	✓	0.9745	0.7460	0.9676	0.9745	0.93	37.89
Noise		✓	✓	0.9780	0.7130	0.9456	0.9667	0.94	38.10

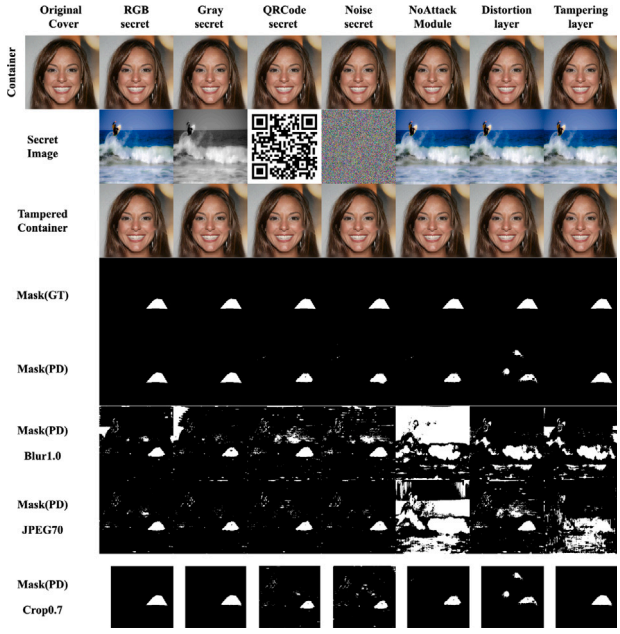


Fig. 5. Samples of pixel-level manipulation detection results of our method in varied setups. Mask (PD) with the name of post-processing methods or without name means the predicated tampered mask from the corresponding post-processed or original tampered container, respectively.

drop. In contrast, networks with distortion layers in their attack module achieve higher detection AUCs on distorted container images, as shown in the second and third rows. It can also be observed in Fig. 5 that the networks without the distortion layer will produce more errors in the predicted mask when the container images are distorted. These performance fluctuations indicate that the distortion layers can improve our method's robustness against image distortion attacks.

Furthermore, by comparing the models equipping the tampering layers (third and fourth rows) to those without tampering layers (first and second rows), we can observe that tampering layers can further improve the method's tampering detection accuracy, especially on the non-distorted container images. As illustrated in Fig. 5, networks with tampering layers produce more accurate predictions of tampered regions, with fewer false negatives than networks without attack modules. Conversely, networks without attack modules tend to underestimate the extent of tampered areas, resulting in reduced detection performance. Networks equipped only with distortion layers exhibit a high rate of false alarms, likely due to the perturbations introduced by the distortion layers.

Overall, the full framework with both distortion and tampering layers in the attack module achieved the best performance on almost all detection metrics, justifying the necessity of all components in the attack module.

4.3. Effectiveness

In this section, we compare our method with other tamper detection methods to reflect our method's advantages and disadvantages in tamper detection. Three passive methods, HP-CNN [2], MVSS-Net [25], and ManTraNet [3], and two proactive methods, PIMD [6], AFG [8], are selected for comparison with our method, for they all have publicly available pre-trained models and source codes.

Pixel-Level comparison. We first detect pixel-level tampering on CASIA 1 [44], CASIA 2 [44], Columbia [45], and 10k randomly selected MS-COCO [46] datasets. We embedded the watermark into these images and applied different tampering operations. Then, we used our method to detect tampering in these images. Since the proactive methods are limited to image-level detection, we only employed three passive methods for detecting forgery on non-watermarked images. As shown in Table 3, our method outperforms other methods on all datasets by a significant margin. It exhibits nearly perfect detection performance on some sets, e.g., IP in CASIA 1 or SP in MS-COCO. Additionally, all set results indicate that our method has a more stable detection capability across all different datasets, while others have extremely higher performance fluctuations.

Image-Level comparison. We then conduct the image-level detection comparison. Our method can achieve image-level tamper detection by setting a threshold that the image will be identified as a forgery when the number of pixels within this image recognized as forgeries is over this threshold. Specifically, we fix to tamper the 30% area of each image, and an image will be classified as tampered when 10% of its pixels are recognized as forgery pixels. We increase this decision threshold from 10% to 30% with step 2% to calculate corresponding detection AUC values. Besides, the same number of authentic images is mixed with tampered in each set to test different methods' real/fake classification ability.

Table 4 summarizes the performance of distinct models. Interestingly, all baseline proactive methods, except ours, almost failed in tampering detection compared with passive methods. This underperformance is believed to stem from their reliance on generative models for training, which limits their effectiveness against unfamiliar tampering operations. This limitation significantly hinders their practical application. In contrast, our method, agnostic to cover images and tampering operations, demonstrates versatility across different scenarios.

As for the comparison between pixel-level and image-level detection accuracy, despite the degradation of AUC scores across all methods compared to pixel-level detection results, our method again emerges as the top performer. These findings demonstrate our method's superior detection performance at pixel and image levels. Notably, the MVSS-Net and ManTraNet exhibit more severe degradation in detection performance than others. We attribute this phenomenon to false-alarm problems of these methods on the authentic pixels/images. We calculate each method's pixel-level detection results' averaged False-Positive Rate (**FPR**) to verify this hypothesis. According to Table 5, the baseline methods' FPR is significantly higher than ours. These results suggest that other methods produce more false alarms on the authentic pixels than ours, leading to unreliable detection results that limit their practical reliability. On the contrary, our method can provide more reliable detection effects on both tampered and authentic pixels/images.

Table 3
The averaged pixel-level tamper detection **AUC** ↑ results from different methods.

Detection	CASIA 1			CASIA 2			Columbia			MS-COCO		
methods	CM	IP	SP	CM	IP	SP	CM	IP	SP	CM	IP	SP
MVSS-Net	0.9118	0.9376	0.9063	0.9118	0.9243	0.9168	0.9350	0.9104	0.9261	0.9050	0.9199	0.9108
HP-FCN	0.5995	0.6837	0.5906	0.5885	0.6734	0.5909	0.5580	0.6816	0.5845	0.4085	0.6912	0.4050
ManTraNet	0.9064	0.8987	0.8862	0.8774	0.8966	0.8788	0.9161	0.9205	0.8851	0.9411	0.9049	0.8890
Ours	**0.9793**	**0.9930**	**0.9959**	**0.9778**	**0.9924**	**0.9959**	**0.9719**	**0.9936**	**0.9906**	**0.9663**	**0.9991**	**0.9915**

Note: the abbreviation **CM** represents CopyMove, IP represents Inpainting, **SP** represents Splice and **DF** represents DeepFake.

Table 4
The averaged image-level tamper detection **AUC** ↑ results from different methods.

Detection	CASIA 1			CASIA 2			Columbia			MS-COCO		
methods	CM	IP	SP	CM	IP	SP	CM	IP	SP	CM	IP	SP
MVSS-Net	0.8010	0.7945	0.8134	0.8201	0.7847	0.8130	0.8210	0.8014	0.8203	0.7900	0.7887	0.8108
HP-FCN	0.5159	0.5837	0.5100	0.5534	0.5974	0.6100	0.5050	0.5860	0.5145	0.5085	0.5901	0.4950
ManTraNet	0.7850	0.7780	0.7915	0.8074	0.8090	0.7880	0.8105	0.8130	0.7905	0.8330	0.8158	0.7930
PIMD	0.6854	0.6870	0.6951	0.6274	0.6390	0.6686	0.6725	0.6738	0.6609	0.6839	0.6585	0.6590
AFG	0.6050	0.6078	0.6175	0.6054	0.6238	0.6180	0.6105	0.6170	0.6205	0.6030	0.6144	0.6087
Ours	**0.9139**	**0.9353**	**0.9291**	**0.9210**	**0.9213**	**0.9312**	**0.9152**	**0.9301**	**0.9445**	**0.9236**	**0.9080**	**0.9210**

Note: the abbreviation **CM** represents CopyMove, **IP** represents Inpainting, **SP** represents Splice and **DF** represents DeepFake.

Table 5
The averaged pixel-level tamper detection **FPR** ↓ from different methods.

Detection methods	CopyMove	Inpainting	Splice
MVSS-Net	0.1921	0.0526	0.077
HP-FCN	0.0419	0.0413	0.0388
ManTraNet	0.1444	0.0419	0.0988
Ours	**0.0144**	**0.0176**	**0.0124**

Computation efficiency comparison. We also measure the detection efficiency in terms of Frames Per Second (FPS). Tested on NVIDIA Tesla K80 GPU, HP-FCN, MVSS-Net, and ManTra-Net run at FPS of 3.91, 19.84 and 2.73, respectively. The watermarking stage of our method runs at 9.88 FPS, while the detection stage runs at 36.71 FPS. Our method's primary time consumption is mainly associated with embedding the watermark into images. In practice, this stage can be executed before tamper detection. Therefore, only considering the detection stage, the FPS of our method is sufficient for real-time application.

Qualitative comparison. Finally, we visualize each method's detection results in Fig. 6. Consistent with qualitative results, the figure illustrates that our method can accurately locate the tampered regions with significantly fewer false positive pixels, especially when detecting authentic images. While other methods can also pinpoint the tampered regions, they are accompanied by some false alarm results in both tampered and authentic images. Therefore, our method's detection results are more reliable.

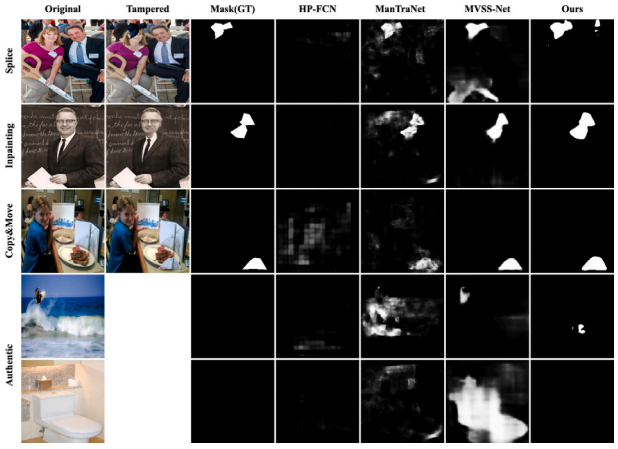


Fig. 6. Examples of mask prediction from different detection methods. Compared with other methods, ours can accurately pinpoint the tampered pixels and produce significantly fewer false alarms on the authentic pixels.

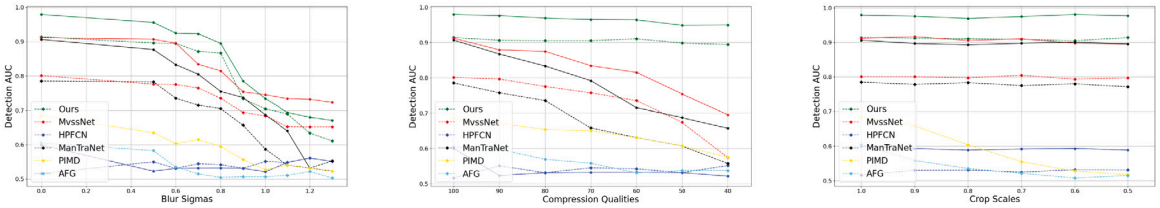
4.4. Robustness

In real life, one may disguise a tampered image with additional post-processing to evade detection or apply various post-processing on the container image for different applications. The detection methods should remain effective against these processing techniques. Here, we consider five typical post-processing techniques and also present how we apply them in the following experiment:

- **Gaussian blur** the images with kernel standard deviation ranging from 0.5 to 1.3 with a step size of 0.1.
- **JPEG compress** all images with quality factors ranging from 40 to 100 with a step size of 10.
- **Crop** the image to a smaller size ranging from 100% to 50% with a step size of 10%.
- **Horizontally flip** all images.
- **Color adjusts** all images' brightness, contrast, saturation and hue randomly.

We apply the above post-processing to the tampered images and then employ different methods to detect the tampered regions. Our watermarks are directly embedded into the pixel values of cover images, eliminating the need for specialized embedding processes for different types of post-processed images. As shown in Table 6 and Fig. 7, our method is immune to crop, JPEG compression and horizontal flip, whereas it is susceptible to Gaussian blur and color adjustment. Especially for the Gaussian blur, the overall performance almost drops linearly. The main reason is that these processing methods would modify the images' pixel value, further distorting our embedded watermark so that they will significantly impact our method's detection capability.

However, compared to the robust performance of other methods, which still fall short of our results and are prone to similar distortions, our method demonstrates superior detection performance. This output indicates its reliability and effectiveness in detecting tampering under real-world distortive conditions.



(a) GaussianBlur (b) JPEGCompress (c) Crop

Fig. 7. The averaged image/pixel-level tamper detection **AUC** ↑ from different methods against Gaussian blur, JPEG compress and crop. The solid line indicates the pixel-level detection results, while the dashed line represents the image-level detection results. Our method is robust against crop and JPEG compress but sensitive to Gaussian blur. However, our method still achieves better detection performance than other methods under these post-processing distortions.

Table 6
The averaged image/pixel-level detection **AUC** ↑ results from different methods under color adjustment and horizontal flipping.

	Detection methods					
	MVss-Net	HP-FCN	ManTraNet	PIMD	AFG	Ours
Color	**0.77/0.88**	0.53/0.55	0.73/0.86	(N/A)/0.65	(N/A)/0.59	0.78/0.84
Flip	0.79/0.91	0.54/0.58	0.78/0.89	(N/A)/0.67	(N/A)/0.61	**0.92/0.98**

Table 7
The averaged similarity between original and watermarked images from different image hiding methods.

Quality	Deep hiding techniques			LSB	Ours
metrics	HiNet	DDH	UDH		
SSIM ↑	**0.97**	0.71	0.84	0.78	0.94
PSNR ↑	**40.17**	26.24	29.77	28.05	38.05

4.5. Imperceptibility

In this section, we evaluate the visual quality of the container image to reflect the stealthiness and imperceptibility of our watermark, thereby ensuring that it does not compromise the utility of the watermarked image. As there is no available deep watermark-based tamper detection method for comparison, we compare our method with SOTA invisible watermarking methods. To ensure a fair comparison, we only select methods with a payload capacity higher than 24 bits per pixel (bpp), which can hide an entire image into a cover image. As a result, HiNet [38], DDH [47], and UDH [35] are selected as the learning-based baselines, and 4bit-LSB [26] which is a traditional watermarking technique, is also selected as a reference in our comparison.

To assess the imperceptibility of our method and all baselines, we use them to embed the same secret image into 1k randomly selected MS-COCO images to generate corresponding container image sets. We then calculate PSNR and SSIM between each method's container and cover images. The numerical results are summarized in Table 7. From the table, we observe that HiNet achieves exceptionally high values on both metrics, while our method ranks second with slightly inferior performance (SSIM 0.04 lower and PSNR 2.0 dB lower), which still outperforms other methods by a large margin.

While our method may not have the best quantitative imperceptibility performance, the qualitative results in Fig. 8 demonstrate that the container images of our method are sufficiently naturalistic for human observers. Both HiNet and our method generate high visual quality container images with more visual-realistic and accurately preserve the original's hue and light in corresponding watermarked images. In contrast, UDH and DDH introduce apparent artifacts in their watermarked images. LSB outputs have obvious color distortions. The high visual quality of the container image can also verify our watermark's imperceptibility.

Besides evaluating the similarity between the container and cover images, we employ an open-source steganalysis tool called StegExpose to measure the anti-steganalysis ability of each method's watermark.

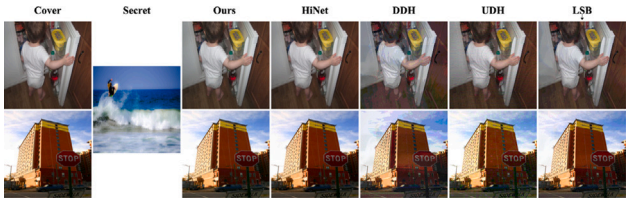


Fig. 8. Qualitative comparison between our method and SOTA hiding techniques' watermarked images. Our outputs are perceptually identical to the original, while the results of LSB have slight color distortion, and DDH and UDH introduce noticeable artifacts in their outputs.

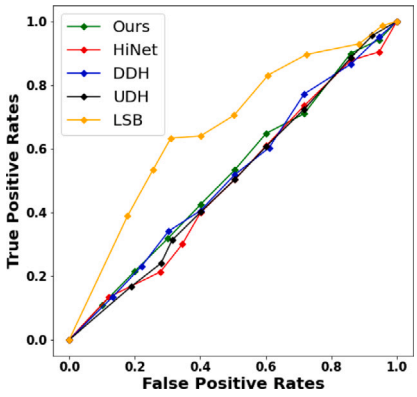


Fig. 9. ROC curves of StegExpose for detecting different image-hiding methods. The detection accuracy on all learning-based methods is close to random guess, demonstrating the highly imperceptible to fool the Statical steganalysis tool.

Fig. 9 shows the ROC curve of each method. We can see that the StegExpose detection accuracy on learning-based methods is quite close to the random guess, indicating their watermarked images are highly imperceptible to fool the Statistical steganalysis tool.

In summary, qualitative and quantitative results demonstrate that our method can generate high visual-quality container images and imperceptible watermarks, ensuring that our watermarked image can still be used normally in real-world scenarios.

4.6. Security

This section investigates potential security risks that the attackers may exploit to compromise the effectiveness of the proposed method.

The security of the secret image. Our initial investigation focuses on determining whether adversaries can acquire users' secret images by analyzing information from container or cover images.

The first line of defense against this threat is the imperceptibility of our watermark. As demonstrated by the results of our imperceptibility experiments, adversaries cannot distinguish between the container and

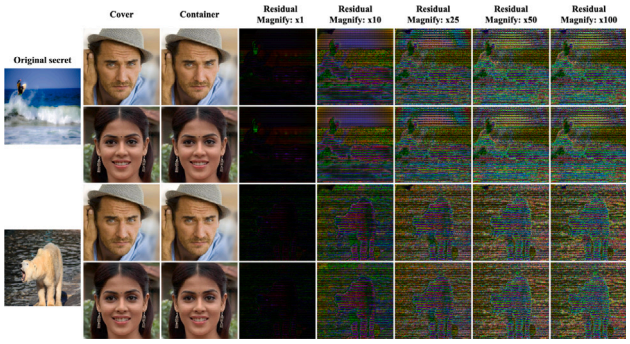


Fig. 10. Residuals between our method's container and cover images. As expected, the residuals without magnification are nearly equal to blank. Magnifying the residuals may reveal the embedded secret images' outlines, but still very ambiguous.

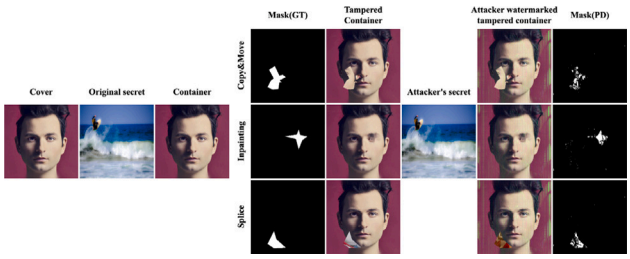


Fig. 11. Samples of the experiment result when the secret image is leaked. Embedding the same secret image into the tampered container images will lead our method to miss some tampered pixels in detection results, but the major tampered regions are still accurately localized. Additionally, this cloak action will also produce apparent artifacts in the outputs.

cover images using quantitative, qualitative or steganalysis-based analyses. Consequently, it would be challenging for adversaries to locate images containing secret information, let alone obtain them.

In our method's application scenario, users share only the container images on platforms with risks of malicious tampering while keeping their cover images private. As a result, adversaries cannot access original cover images without users' permission. To fully validate the security of our method, we investigate whether the secret image is safe when the cover images leak. As illustrated in Fig. 10, the residuals between the container and cover images are exceedingly thin, and even with 50 or 100 times magnification, they remain ambiguous. Consequently, adversaries cannot obtain the secret images according to the container and cover images.

Risk of the secret image leakage. We then consider the scenario in which users' secret images are compromised while the pre-trained models remain secure. Adversaries will use their trained models to embed these secret images into their tampered images to conceal the tampered area and evade our detection.

To simulate this attack, we trained two different groups of models on an identical dataset to represent the user and adversary, respectively. We first employ user networks to embed the secret image into 1k randomly selected CelebaHQ images, followed by tampering operations. We then use adversary networks to embed the same secret image into these tampered images to simulate the adversary's cloaking action. Finally, we utilize the user's networks to detect tampered and cloaked images, testing whether our method is still effective. Table 8 summarizes the detection FPR, TPR, Precision and F1 Score, while Fig. 11 provides visual representations of the results.

Our findings indicate that this attack reduces the detection performance of our method, particularly by generating more false negative tampered pixels. However, the TPR suggests that more than 65% of tampered areas can still be successfully localized, demonstrating that

Table 8
Pixel-level detection results when the secret image leaks. w/o represents without, and w represents with.

	TPR ↑	FPR ↓	Precision ↑	F1 Score ↑
CopyMove				
w/o cloak	0.7035	0.0140	0.9490	0.8080
w cloak	0.6114	0.0162	0.9441	0.7422
Inpainting				
w/o cloak	0.7271	0.0195	0.9273	0.8151
w cloak	0.6114	0.0162	0.9441	0.7422
Splice				
w/o cloak	0.7209	0.0131	0.9518	0.8204
w cloak	0.6114	0.0162	0.9441	0.7422

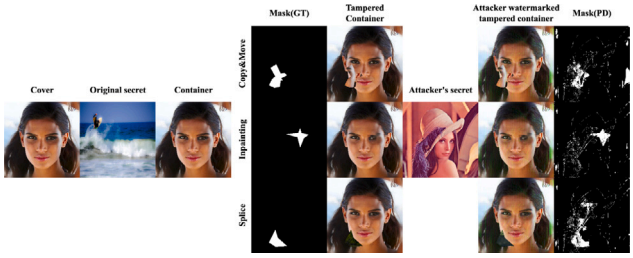


Fig. 12. Samples of the experiment result when the pre-trained model is leaked. Embedding the different secret images using the same models into the tampered containers will make our method produce more false alarms when detecting authentic pixels. However, the tampered areas are still accurately identified, and apparent artifacts appear in the cloaked images.

adversaries cannot wholly evade detection. Furthermore, the qualitative results presented in Fig. 11 validate the above inference that, despite reduced detection accuracy, the primary tampered regions in the container images are still accurately identified. We attribute this to the inability of different network groups to replace the watermarked secret images in each container image perfectly.

Additionally, we observe that cloaked images exhibit apparent artifacts, likely due to the overflow of embedding the extra secret image into already watermarked images. This phenomenon renders the attack meaningless again, as cloaked images are easy to identify and forbidden.

Risk of the pre-trained model. Finally, we analyze the situation where users' pre-trained models are compromised while their secret images remain safe. Adversaries will use the same networks to embed different secret images into their tampered images to conceal the tampered regions and evade our detection.

To simulate this attack, we first employ our networks to embed a secret image (secret a) into the cover images and then perform tamper operations on these images. Next, we use the same networks to embed another secret image (secret b) into these tampered images to imitate the adversaries' concealing process. Finally, we conduct tamper detection on these images based on the secret image a.

The results are provided in Table 9 and Fig. 12. Similar to the previous, this attack also declined our method's detection performance, but as evidenced by FPR values, it will lead to more false alarms on the authentic pixels in detection results rather than falsely negating tampered pixels. The results in Fig. 12 are consistent with quantitative results, as the tampered regions can be accurately identified, but together with some false alarms on the authentic pixels.

We infer that this phenomenon occurs because the newly embedded secret image destroys the original watermark in the container image. Therefore, when comparing the original secret image with the secret image recovered from this cloaked container, the destroyed area in the watermark will result in false alarms. Nevertheless, as the results show, the tampered regions can still be identified.

Table 9
Pixel-level detection results when pre-trained models leaked. w/o represents without, and w represents with.

	TPR ↑	FPR ↓	Precision ↑	F1 Score ↑
CopyMove				
w/o cloak	0.7094	0.0144	0.9473	0.8113
w cloak	0.6750	0.0236	0.9141	0.7766
Splice				
w/o cloak	0.7232	0.0182	0.9323	0.8146
w cloak	0.6760	0.0292	0.8924	0.7693
Splice				
w/o cloak	0.7258	0.0138	0.9489	0.8225
w cloak	0.6356	0.0220	0.9218	0.7524

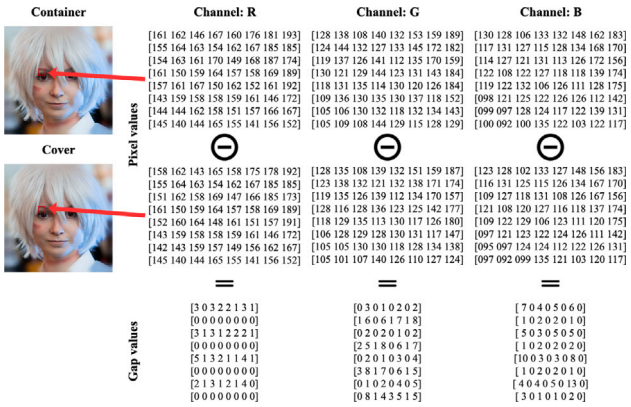


Fig. 13. The partial pixel values of cover and container images and the values gap between them. Since adding with watermark in it, the container image's pixel values are different from the cover image. Nevertheless, the values gap is tiny, so there is no perceptual difference between cover and container images, indicating our method's perfect concealing ability.

In summary, we have validated our method's security, which can keep users' secret images safe in the practical application scenario and sustain detection performance even if some credential information is compromised. However, considering the performance degradation when credentials are leakage, it is still essential for users of our method to keep their secret images and pre-trained models private.

4.7. Analysis

According to the above examinations, it is clear that our method is capable of reliable pixel and image-level tamper detection. However, it is still insufficient to understand our watermark and the watermarked image. In this section, we will implement an investigation to analyze how our method works.

Pixel value Analysis. We begin our investigation by analyzing the pixel values of our watermark. Our method incorporates the watermark into the cover image by directly pixel-wise addition to generate the corresponding container image. Consequently, this process inevitably introduces pixel differences between the cover and container images. However, the Hiding secret loss of our method enforces the container to be identical to the cover image. This design ensures that the Hiding network produces the watermark with minimal pixel values to include the secret image's information.

As a result, there is no noticeable distinction between cover and container images, even though their pixel values are slightly different. To support our hypothesis, we provide Fig. 13 as evidence. We can observe that although the pixel values of the cover and container images within the red box differ in all three RGB channels, this difference is very slight. Therefore, the changes in pixel values resulting from the watermark would not produce distinguishable perceptual alterations in the container image.

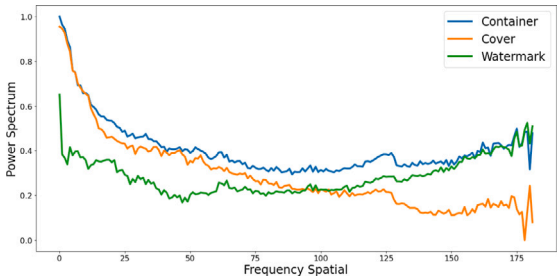


Fig. 14. The averaged Azimuthal Integral values of different cover, container and watermark images. The watermarks mainly consist of a high-frequency spectrum. The container images' low-frequency distribution is almost aligned with that of the cover images while having a higher high-frequency spectrum. So, it would be straightforward to conclude that this high-frequency difference is due to the embedded watermarks on containers.

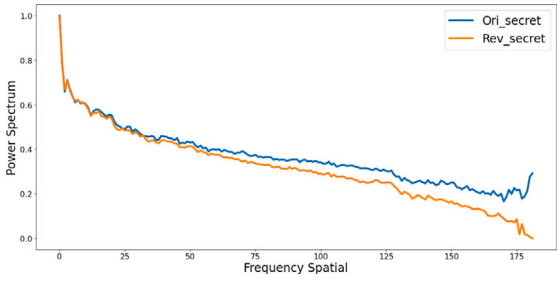


Fig. 15. The frequency distribution of the original and recovered secret images. They are almost perfectly aligned in most frequency domains but only differ in the high-frequency.

Frequency analysis. We then compute the averaged Azimuthal Integral (**AI**) [48] values of different images that appear in our method to explore their frequency properties. In brief, Azimuthal Integral returns the relative frequency intensity distribution spectrum, where the intensity begins with the highest value at the lowest frequency and decreases as the frequency increases. The outcomes are plotted in Figs. 14 and 15.

From Fig. 14, it can be observed that the frequency distribution of the container image in the low-frequency domain almost aligns with that of the cover images. Since the high-frequency contents in images are generally imperceptible to human observers, it explains why container images are perceptually indistinguishable from their corresponding cover images. Similarly, the watermark images, which are almost perceptually invisible, mainly consist of high-frequency distribution with a significantly weak low-frequency spectrum. Consequently, the container images embedded with these watermark images will only exhibit some high-frequency distortion compared to the cover images but no noticeable artifacts. The secret images' frequency distributions in Fig. 15 are almost perfectly aligned in most frequency domains but only slightly differ in the high-frequency domain. These imperfections could be attributed to the imperfection reconstruction of our method, but based on the previous experimental results, we consider them acceptable.

5. Conclusion and discussion

The task of image tamper detection has become increasingly challenging due to the constant evolution of image editing and synthesizing techniques. Confirming the authenticity of images by identifying artifacts left from the manipulation process has become more complex than ever before. To address these challenges, we propose a proactive

method to protect the authenticity of images by embedding a semi-fragile and invisible watermark into each target image. This watermark serves as an indicator to verify the authenticity of the image's pixels.

The experiment results have demonstrated that our method performs exceptionally well across all evaluation metrics. However, it should be noted that our method follows a distinct pipeline from other current image tamper detection methods, which requires additional steps. Given the perfect detection performance of our method, we believe this overhead is acceptable. At present, the proposed method is best suited for protecting critical information. For example, public celebrities can use our method to add personalized watermarks to their images to prevent malicious image forgery and tampering. Only images with their watermarks can be considered authoritative, while images without the corresponding watermarks will be assumed to come from unofficial channels. They can also verify the tampered regions and declare forgery to reduce reputation loss.

Overall, this work represents a new direction for proactively fighting against malicious tampering operations on image data.

CRediT authorship contribution statement

Yuan Zhao: Data curation, Formal analysis, Investigation, Methodology, Visualization, Writing – original draft, Writing – review & editing. **Bo Liu:** Funding acquisition, Methodology, Supervision, Validation, Writing – review & editing, Formal analysis. **Tianqing Zhu:** Funding acquisition, Methodology, Supervision, Writing – review & editing. **Ming Ding:** Formal analysis, Methodology, Validation, Writing – review & editing. **Xin Yu:** Methodology, Writing – review & editing. **Wanlei Zhou:** Funding acquisition, Project administration, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work is supported by two ARC Projects (LP220200808 and DP230100246) from the Australian Research Council, Australia.

Appendix A. Network architecture

We illustrate the detailed structure of our networks in Fig. A.16. Our method consists of two U-Net style pure convolutional networks, the Hiding network and the Revealing network, respectively.

The left side of Fig. A.16 depicts the architecture of the hiding network, which is symmetric with down- and up-sampling blocks. The features extracted by the down-sampling blocks are passed on to the up-sampling blocks for further processing. Specifically, the down-sampling block consists of a 4×4 *Conv2d* layer with a stride of 2, a *LeakyReLU* layer with a negative sloop of 0.1, and a *BatchNorm* layer. The cover image is firstly down-sampled to a tensor with a size of $1024 \times 2 \times 2$, which is then up-sampled to obtain the watermark. The up-sampling block includes a 4×4 *ConvTransposed2d* with a stride of 2, a *ReLU* layer, and a *BatchNorm* layer. At the end of the up-sampling process, the sigmoid function projects the outputs pixel value to -1 to 1 as the subsequent embedding process's watermark. This architecture enables the up-sampling blocks to share the features extracted by the down-sampling blocks, thereby preserving the input information for generating the output.

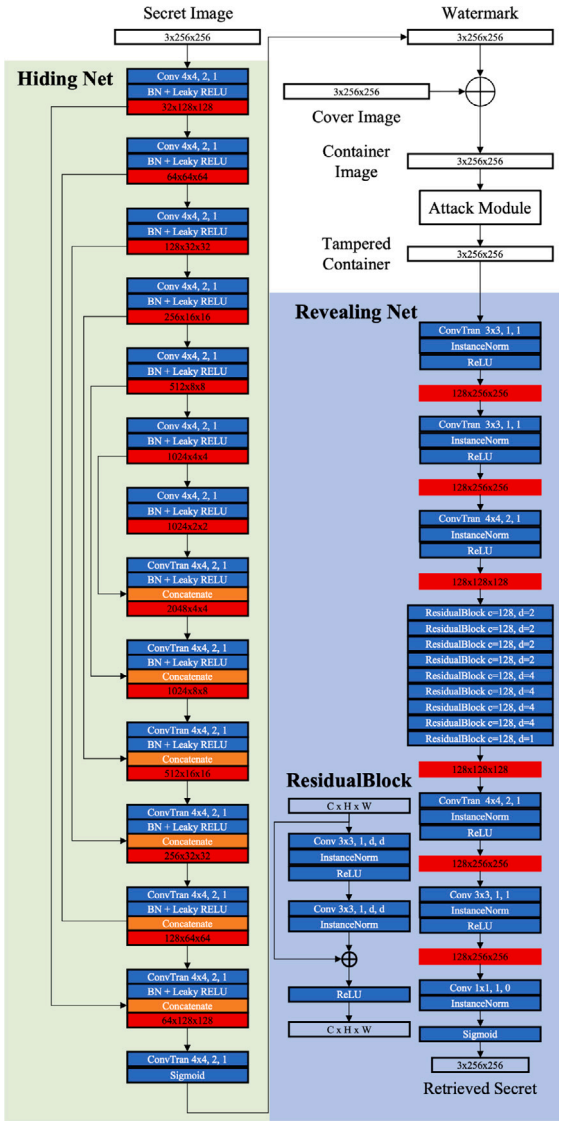


Fig. A.16. Network architecture. *Conv k,s,p* represents a Convolutional Layer with kernel size *k*, stride *s* and padding *p*. *ConvTran k,s,p* represents a Transposed Convolutional Layer with kernel size *k*, stride *s* and padding *p*. All *Leaky ReLUs* have $\alpha = 0.1$.

The revealing network's architecture illustrated on the right side of Fig. A.16 comprises three parts: the down-sampling, the residue, and the up-sampling. In the down-sampling part, the unit block consists of a 3×3 *Conv2d* layer, a *BatchNorm* layer, and a *ReLU* layer. Specifically, the stride is set to two in the last block of the down-sampling part to enlarge the receptive field. In the residue part, nine residue blocks, comprising 18 convolution layers, generate residual features. In the up-sampling part, the residual features are up-sampled to recover the secret image. The unit block here is similar to the down-sampling part, with only differences in the number of input and output channels. For up-sampling, the first block's *Conv2d* layer has a stride and kernel size of 2 and 4×4, respectively. Finally, the sigmoid function is applied to output the final recovered secret image. This architecture is based on CEILNet, which performs well when its output differs significantly from its input, making it suitable for retrieving the secret image from the container image.

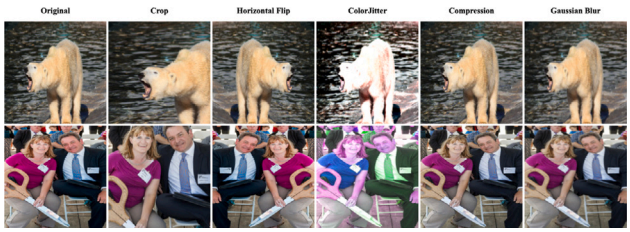


Fig. C.17. Samples of each post-processing method are adopted in the robustness evaluation. Note that the Crop's samples are resized in this figure for better visualization.

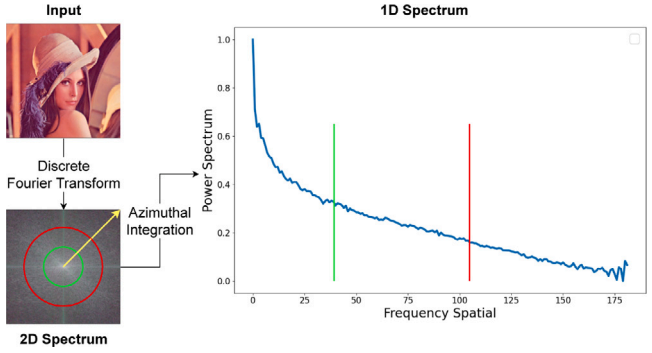


Fig. D.18. Azimuthal Integral. The input image is transformed into a 2D spectrum using discrete Fourier transform. Then, the integration is conducted from the inside of the 2D spectrum to the outside, along the yellow arrow circularly. Eventually, the 1D spectrum is derived, where the red and green lines represent the sum of the pixel values on the red and green circles.

Appendix B. Implementation details

Our method is implemented in PyTorch and trained on an NVIDIA Tesla K80 GPU. The image size in the experiment is set to 256×256. We use an Adam optimizer whose learning rate periodically decays from 10e–4 to 10e–7. We set the two weights in the combined loss as $\lambda_H = 1$ and $\lambda_R = 0.75$, according to the model performance on a held-out validation set from FFHQ.

Appendix C. Post processing samples

We present samples of different post-processing operations' results in Fig. C.17. These samples provide a visual representation of the impact of various post-processing techniques on the images.

Appendix D. Azimuthal integral

In brief, Azimuthal Integral [49] computes the radial integral over the 2D discrete Fourier Transform spectrum along the spatial frequency. Given a square image I of size $M \times N (M = N)$, the spectral representation is computed from the discrete Fourier Transform (DCT)

$$\mathrm{DCT}(I)(k,l) = \sum_{m=1}^{M}\sum_{n=1}^{N} e^{-2\pi i \cdot \frac{jk}{M}} e^{-2\pi i \cdot \frac{jl}{N}} \cdot I(m,n), \quad \text{for} \quad k = 1,\dots,M, \quad l = 1,\dots,N, \tag{D.1}$$

via Azimuthal Integration over radial frequencies ϕ

$$\mathrm{AI}(\omega_k) = \int_0^{2\pi} \|\mathrm{DCT}(I)(\omega_k \cdot \cos(\phi), \omega_k \cdot \sin(\phi))\|^2 d\phi \quad \text{for} \quad k = 1,\dots,M/2. \tag{D.2}$$

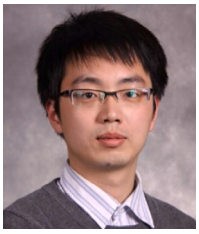
As depicted in Fig. D.18, the 1D Azimuthal Integral power spectrum reflects the relative intensity of the 2D spectrum at a certain frequency spatial coordinate.

References

[1] X. Hu, Z. Zhang, Z. Jiang, S. Chaudhuri, Z. Yang, R. Nevatia, SPAN: Spatial pyramid attention network for image manipulation localization, in: European Conference on Computer Vision, Springer, 2020, pp. 312–328.
[2] H. Li, J. Huang, Localization of deep inpainting using high-pass fully convolutional network, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 8301–8310.
[3] Y. Wu, W. AbdAlmageed, P. Natarajan, Mantra-net: Manipulation tracing network for detection and localization of image forgeries with anomalous features, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 9543–9552.
[4] R. Salloum, Y. Ren, C.-C.J. Kuo, Image splicing localization using a multi-task fully convolutional network (MFCN), J. Vis. Commun. Image Represent. 51 (2018) 201–209.
[5] P. Zhou, B.-C. Chen, X. Han, M. Najibi, A. Shrivastava, S.-N. Lim, L. Davis, Generate, segment, and refine: Towards generic manipulation segmentation, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, 2020, pp. 13058–13065.
[6] V. Asnani, X. Yin, T. Hassner, S. Liu, X. Liu, Proactive image manipulation detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 15386–15395.
[7] R. Wang, F. Juefei-Xu, M. Luo, Y. Liu, L. Wang, Faketagger: Robust safeguards against deepfake dissemination via provenance tracking, in: Proceedings of the 29th ACM International Conference on Multimedia, 2021, pp. 3546–3555.
[8] N. Yu, V. Skripniuk, S. Abdelnabi, M. Fritz, Artificial fingerprinting for generative models: Rooting deepfake attribution in training data, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 14448–14457.
[9] F. Di Martino, S. Sessa, Fragile watermarking tamper detection via bilinear fuzzy relation equations, J. Ambient Intell. Humaniz. Comput. 10 (2019) 2041–2061.
[10] S. Bhalerao, I.A. Ansari, A. Kumar, A secure image watermarking for tamper detection and localization, J. Ambient Intell. Humaniz. Comput. 12 (1) (2021) 1057–1068.
[11] I.J. Cox, J. Kilian, F.T. Leighton, T. Shamoon, Secure spread spectrum watermarking for multimedia, IEEE Trans. Image Process. 6 (12) (1997) 1673–1687.
[12] S. Pereira, J.J.O. Ruanaidh, F. Deguillaume, G. Csurka, T. Pun, Template based recovery of Fourier-based watermarks using log-polar and log-log maps, in: Proceedings IEEE International Conference on Multimedia Computing and Systems, Vol. 1, IEEE, 1999, pp. 870–874.
[13] N. Bi, Q. Sun, D. Huang, Z. Yang, J. Huang, Robust image watermarking based on multiband wavelets and empirical mode decomposition, IEEE Trans. Image Process. 16 (8) (2007) 1956–1966.
[14] A. Shehab, M. Elhoseny, K. Muhammad, A.K. Sangaiah, P. Yang, H. Huang, G. Hou, Secure and robust fragile watermarking scheme for medical images, IEEE Access 6 (2018) 10269–10278.
[15] S. Pereira, T. Pun, Robust template matching for affine resistant image watermarks, IEEE Trans. Image Process. 9 (6) (2000) 1123–1129.
[16] E.T. Lin, C.I. Podilchuk, E.J. Delp III, Detection of image alterations using semifragile watermarks, in: Security and Watermarking of Multimedia Contents II, vol. 3971, SPIE, 2000, pp. 152–163.
[17] R. Sun, H. Sun, T. Yao, A SVD-and quantization based semi-fragile watermarking technique for image authentication, in: 6th International Conference on Signal Processing, 2002, Vol. 2, IEEE, 2002, pp. 1592–1595.
[18] X. Yu, C. Wang, X. Zhou, Review on semi-fragile watermarking algorithms for content authentication of digital images, Future Internet 9 (4) (2017) 56.
[19] D. Cozzolino, D. Gragnaniello, L. Verdoliva, Image forgery localization through the fusion of camera-based, feature-based and pixel-based techniques, in: 2014 IEEE International Conference on Image Processing, ICIP, IEEE, 2014, pp. 5302–5306.
[20] P. Ferrara, T. Bianchi, A. De Rosa, A. Piva, Image forgery localization via fine-grained analysis of CFA artifacts, IEEE Trans. Inf. Forensics Secur. 7 (5) (2012) 1566–1577.
[21] T.J. De Carvalho, C. Riess, E. Angelopoulou, H. Pedrini, A. de Rezende Rocha, Exposing digital image forgeries by illumination color classification, IEEE Trans. Inf. Forensics Secur. 8 (7) (2013) 1182–1194.
[22] A. Islam, C. Long, A. Basharat, A. Hoogs, Doa-gan: Dual-order attentive generative adversarial network for image copy-move forgery detection and localization, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 4676–4685.
[23] Y. Zhang, J. Goh, L.L. Win, V.L. Thing, Image region forgery detection: A deep learning approach., SG-CRC 2016 (2016) 1–11.
[24] B. Bayar, M.C. Stamm, A deep learning approach to universal image manipulation detection using a new convolutional layer, in: Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security, 2016, pp. 5–10.
[25] C. Dong, X. Chen, R. Hu, J. Cao, X. Li, MVSS-Net: Multi-view multi-scale supervised networks for image manipulation detection, IEEE Trans. Pattern Anal. Mach. Intell. (2022).

[26] A.A. Tamimi, A.M. Abdalla, O. Al-Allaf, Hiding an image inside another image using variable-rate steganography, Int. J. Adv. Comput. Sci. Appl. (IJACSA) 4 (10) (2013).
[27] J. Ruanaidh, W. Dowling, F.M. Boland, Phase watermarking of digital images, in: Proceedings of 3rd IEEE International Conference on Image Processing, Vol. 3, IEEE, 1996, pp. 239–242.
[28] C.-T. Hsu, J.-L. Wu, Hidden digital watermarks in images, IEEE Trans. Image Process. 8 (1) (1999) 58–68.
[29] M. Barni, F. Bartolini, A. Piva, Improved wavelet-based watermarking through pixel-wise masking, IEEE Trans. Image Process. 10 (5) (2001) 783–791.
[30] N. Provos, P. Honeyman, Hide and seek: An introduction to steganography, IEEE Secur. Priv. 1 (3) (2003) 32–44.
[31] J. Zhu, R. Kaplan, J. Johnson, L. Fei-Fei, Hidden: Hiding data with deep networks, in: Proceedings of the European Conference on Computer Vision, ECCV, 2018, pp. 657–672.
[32] S.-M. Mun, S.-H. Nam, H.-U. Jang, D. Kim, H.-K. Lee, A robust blind watermarking using convolutional neural network, 2017, arXiv preprint arXiv:1704.03248.
[33] S. Baluja, Hiding images in plain sight: Deep steganography, Adv. Neural Inf. Process. Syst. 30 (2017).
[34] R. Zhang, S. Dong, J. Liu, Invisible steganography via generative adversarial networks, Multimedia Tools Appl. 78 (7) (2019) 8559–8575.
[35] C. Zhang, P. Benz, A. Karjauv, G. Sun, I.S. Kweon, Udh: Universal deep hiding for steganography, watermarking, and light field messaging, Adv. Neural Inf. Process. Syst. 33 (2020) 10223–10234.
[36] X. Luo, R. Zhan, H. Chang, F. Yang, P. Milanfar, Distortion agnostic deep watermarking, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 13548–13557.
[37] X. Liao, J. Yin, M. Chen, Z. Qin, Adaptive payload distribution in multiple images steganography based on image texture features, IEEE Trans. Dependable Secure Comput. (2020).
[38] J. Jing, X. Deng, M. Xu, J. Wang, Z. Guan, HiNet: deep image hiding by invertible network, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 4733–4742.
[39] P. Neekhara, S. Hussain, X. Zhang, K. Huang, J. McAuley, F. Koushanfar, FaceSigns: semi-fragile neural watermarks for media authentication and countering deepfakes, 2022, arXiv preprint arXiv:2204.01960.
[40] C. Zhang, A. Karjauv, P. Benz, I.S. Kweon, Towards robust deep hiding under non-differentiable distortions for practical blind watermarking, in: Proceedings of the 29th ACM International Conference on Multimedia, 2021, pp. 5158–5166.
[41] Q. Fan, J. Yang, G. Hua, B. Chen, D. Wipf, A generic deep architecture for single image reflection removal and image smoothing, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 3238–3247.
[42] T. Karras, S. Laine, T. Aila, A style-based generator architecture for generative adversarial networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 4401–4410.
[43] T. Karras, T. Aila, S. Laine, J. Lehtinen, Progressive growing of gans for improved quality, stability, and variation, 2017, arXiv preprint arXiv:1710.10196.
[44] J. Dong, W. Wang, T. Tan, Casia image tampering detection evaluation database, in: 2013 IEEE China Summit and International Conference on Signal and Information Processing, IEEE, 2013, pp. 422–426.
[45] T.-T. Ng, J. Hsu, S.-F. Chang, Columbia Image Splicing Detection Evaluation Dataset, DVMM lab. Columbia Univ CalPhotos Digit Libr, 2009.
[46] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C.L. Zitnick, Microsoft coco: Common objects in context, in: European Conference on Computer Vision, Springer, 2014, pp. 740–755.
[47] X. Weng, Y. Li, L. Chi, Y. Mu, High-capacity convolutional video steganography with temporal residual modeling, in: Proceedings of the 2019 on International Conference on Multimedia Retrieval, 2019, pp. 87–95.
[48] C. Liu, H. Chen, T. Zhu, J. Zhang, W. Zhou, Making DeepFakes more spurious: evading deep face forgery detection via trace removal attack, 2022, arXiv preprint arXiv:2203.11433.
[49] R. Durall, M. Keuper, J. Keuper, Watch your up-convolution: Cnn based generative deep neural networks are failing to reproduce spectral distributions, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 7890–7899.

Yuan Zhao received his B.Sc. in Internet of Things (IoT) Engineering from Northwest University, China, in 2017 and his M.Sc. in Information Technology (IT) from The University of Sydney (USYD), Australia. He is pursuing his Ph.D. in Computer Science at the Faculty of Information and Technology (FEIT), University of Technology Sydney (UTS), Australia. His research interests focus on leveraging deep learning and machine learning techniques to enhance visual data's privacy and security against emerging AI-based threats.



Bo Liu received the BEng degree from the Department of Computer Science and Technology, Nanjing University of Posts and Telecommunications, Nanjing, China, in 2004. He then received the MEng. and PhD. Degrees from the Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2007 and 2010, respectively. He is currently a Senior Lecturer with the University of Technology Sydney, Australia. His research interests include cybersecurity and privacy, location privacy and image privacy, privacy protection and machine learning.

Tianqing Zhu holds BEng and MEng degrees from Wuhan University, Wuhan, China and a Ph.D. in Computer Science from Deakin University, Australia (2014). She is currently an Associate Professor with the School of Computer Science at the University of Technology Sydney, Australia. Prior to that, she was a Lecturer with the School of Information Technology, Deakin University, from 2014 to 2018. Her research interests include privacy-preserving, data mining, and network security.

Ming Ding received the B.S. and M.S. degrees (with first-class Hons.) in electronics engineering from Shanghai Jiao Tong University (SJTU), Shanghai, China, and the Doctor of Philosophy (Ph.D.) degree in signal and information processing from SJTU, in 2004, 2007, and 2011, respectively. From April 2007 to September 2014, he worked at Sharp Laboratories of China in Shanghai, China as a Researcher/Senior Researcher/Principal Researcher. Currently, he is a senior research scientist at Data61, CSIRO, in Sydney, NSW, Australia. His research interests include information technology, data privacy and security, and machine learning and AI. He has authored more than 150 papers in IEEE journals and conferences, all in recognized venues, and around 20 3GPP standardization contributions, as well as a book "Multi-point Cooperative Communication Systems: Theory and Applications" (Springer, 2013). Also, he holds 21 US patents and has co-invented another 100+ patents on 4G/5G technologies. Currently, he is an editor of IEEE Transactions on Wireless Communications and IEEE Communications Surveys and Tutorials. Besides, he has served as a guest editor/co-chair/co-tutor/TPC member for multiple IEEE top-tier journals/conferences and received several awards for his research work and professional services.

Xin Yu received his B.S. degree in Electronic Engineering from University of Electronic Science and Technology of China, Chengdu, China, in 2009, and received his Ph.D. degree in the Department of Electronic Engineering, Tsinghua University, Beijing, China, in 2015. He also received a Ph.D. degree in the College of Engineering and Computer Science, Australian National University, Canberra, Australia, in 2019. He is currently a lecturer in University of Technology Sydney. His interests include computer vision and image processing.

Wanlei Zhou received his B.S. degree in Electronic Engineering from University of Electronic Science and Technology of China, Chengdu, China, in 2009, and received his Ph.D. degree in the Department of Electronic Engineering, Tsinghua University, Beijing, China, in 2015. He also received a Ph.D. degree in the College of Engineering and Computer Science, Australian National University, Canberra, Australia, in 2019. He is currently a lecturer in University of Technology Sydney. His interests include computer vision and image processing.