EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

Steganographic secret sharing via AI-generated photorealistic images

Kai Gao[1], Ching-Chun Chang[2*], Ji-Hwei Horng[3] and Isao Echizen[2]

*Correspondence: cchang@nii.ac.jp

[1] Department of Information Engineering and Computer Science, Feng Chia University, Taichung, Taiwan
[2] Information and Society Research Division, National Institute of Informatics, Tokyo, Japan
[3] Department of Electronic Engineering, National Quemoy University, Quemoy, Taiwan

Abstract

Steganographic secret sharing is an access control technique that transforms a secret message into multiple shares in a steganographic sense. Each share is in a human-readable format in order to dispel suspicion from a malicious party during transmission and storage. Such a human-readable format can also serve to facilitate data management. The secret can be reconstructed only when a sufficient number of authorized shareholders collaborate. In this study, we use neural networks to encode secret shares into photorealistic image shares. This approach is conceptually related to coverless image steganography in which the data are transformed directly into an image rather than concealed into a cover image. We further implement an authentication mechanism to verify the integrity of the image shares presented in the decoding phase. All coverless image steganography schemes can be used to achieve steganographic secret sharing, but our detection mechanism can further improve the robustness of these schemes. Experimental results confirm the robustness of the proposed scheme against various steganalysis and tampering attacks.

Keywords: Coverless steganography, Generative adversarial networks, Image synthesis, Secret sharing

1 Introduction

With the rapid development of the Internet, communication via social media has become a regular activity in our daily life. Network security, as well as data protection, has become a major concern. A lot of research reports on information security have been published. Access control is an essential technology in data security that enables the authenticated user to access the secret message and keeps the inappropriate user out. As a common technique of access control, the concept of (k, n)-secret sharing was first proposed in 1979 by Shamir [1] and Blakley [2]. Their scheme distributes secret information into n shares and deals them to n participants. When recovering the secret information, any $k (k < n)$ participants can reconstruct the complete secret message by combining their shares. This concept has been applied to the secure transmission of secret information on the Internet. Since then, many scholars have intensively studied this area [3, 4] and proposed visual secret sharing, authenticable secret sharing, and so on.

Steganography is a technique that hides the secret message in a transmission carrier, such as a sound signal, image, and video. Image-based steganography is the most common approach because it is very difficult to detect minor modifications to images. Thus, many scholars use images as the media for secret sharing. In 2008, Chang et al. [5] introduced a Sudoku reference matrix-based secret sharing scheme that can generate two meaningful image shares, but the quality of the image shares was bad. In 2018, Liu et al. [6] proposed a Turtle Shell reference matrix-based secret sharing scheme, which could generate two high-quality image shares when embedding the secret message. In addition, Liu et al. added an authentication mechanism to detect the validity of image shares. In recent years, many scholars have started to study the use of secret sharing to achieve steganography [7–9]

With updates to computer hardware, the deep learning-based steganalysis scheme attracts more and more attention. Such techniques can detect images that have been tampered with by learning the data distribution patterns of natural images. Although traditional secret sharing schemes provide high-quality and meaningful image shares, these image shares are modified from the original image and thus cannot escape detection by steganalysis based on deep learning [10, 11]. Therefore, many scholars have worked to combine deep learning with steganography [12–14] and develop a new kind of steganography that can counteract steganalysis. The essence of this type of scheme is to embed the secret message into the image without changing the latter; this is also known as coverless steganography. Based on the rapid development of deep learning, we want to combine the concept of coverless steganography with secret sharing. However, none of the coverless steganography schemes provide an authentication mechanism. Therefore, we would like to design a convolutional neural network (CNN)-based authentication mechanism to improve the sensitivity of secret sharing.

In this paper, we propose a novel steganographic secret sharing (Stego-SS) scheme that can be used to build an access control mechanism. We first train generator *Gen* and encoder *Enc* to embed the secret share and the authentication code in the image share. Meanwhile, the decoder *Dec* and the extractor *Ext* are trained to extract the authentication code and the secret shares severally. After training, the dealer can use *Gen* and *Enc* to hide the secret share and the authentication code in the generated image shares and distribute them to n participants. Then, the dealer can do access control by choosing the threshold of k. After authentication, any k of participants can work together to extract their secret shares and recover the secret message. The main contributions of this paper are summarized as follows:

1. The concept of coverless steganography is combined with (k, n)-secret sharing.
2. A new CNN-based authentication mechanism is designed to improve the sensitivity of the proposed scheme.
3. The idea of adversarial training is applied to enhance the robustness of the proposed scheme against various types of distortion.

The rest of this paper is organized as follows. Section 2 introduces two types of coverless image steganography. In Sect. 3, we describe the procedures of the proposed steganographic secret sharing in detail. In Sect. 4, experimental results and analysis of

the proposed scheme are provided. Finally, conclusions and prospects are presented in Sect. 5.

2 Preliminaries

Coverless steganography is an advanced data transmission scheme that aims to transmit a secret message without pixel-wise modification of the cover image. In other words, the secret message is directly embedded into the unmodified cover image.

In this section, we briefly review two types of coverless steganography schemes: selective coverless steganography and generative coverless steganography.

2.1 Selective coverless steganography

Selective coverless steganography schemes usually compute the binary hash sequence of each image in a database first. When transmitting a secret message, the image whose hash sequence matches the secret message is selected as the carrier image. Thus, the carrier image requires no further modification. In 2015, Zhou et al. [15] first proposed a selective coverless steganography scheme. A carrier image is divided into multiple blocks, and the average pixel value within each block is calculated first. The hash value is then determined to be 0 or 1 by comparing the average pixel values of adjacent blocks. In the next year, Zhou et al. [16] proposed a coverless steganography scheme based on the bag-of-words model. A carrier image is divided into sub-images, and a corresponding visual word for each sub-image is analyzed. Text information can be communicated by transmitting a series of images containing the sub-images related to the text words. In 2016, Zhou et al. [17] proposed a histogram of oriented gradients (HOG)-based image steganography, in which the hash sequence of a sub-image is generated using HOG. This scheme has good security and strong robustness to various attacks. In the same year, Zhang et al. [18] proposed a robust coverless steganography scheme based on the scale-invariant feature transform (SIFT), in which the hash sequence is calculated using the orientation information of the SIFT feature points. In 2018, Zhang et al. [19] applied the discrete cosine transform and latent Dirichlet allocation (LDA) topic classification to generate the feature sequence of a carrier image. The feature sequence plays the same role as the hash sequence in the previous schemes.

All these selective coverless steganography schemes are based on transmitting the real image with the hash sequence that matches the secret message. However, the hiding capacity of each image is very low. The longer the data length, the harder an image with a matched hash sequence can be found. As the length of the hash sequence grows, the required size of the image database increases significantly. The storage and processing of a large database are inefficient. In addition, since the selective coverless steganography scheme uses the real image to transmit the secret message, this may violate the privacy and portrait rights of the image owner. To solve these problems, generative coverless steganography schemes were proposed.

2.2 Generative coverless steganography

Generative coverless steganography schemes use various types of generative adversarial networks (GANs) to produce a carrier image according to the secret message to be embedded. The generative adversarial network (GAN) is a neural network model that

can generate high-quality images. In 2017, Liu et al. [20] proposed a generative coverless steganography scheme based on the auxiliary classifier generative adversarial network (ACGAN), in which each segment of secret message is mapped to an integer digit from 0 to 9, and the noise and the secret message are fed into the generator network altogether, and then a carrier image related to the secret message is generated through multiple training processes. The embedded secret message can be extracted from the carrier image by using the discriminator network. Inspired by Liu et al., Duan et al. proposed a scheme [21] in which the original secret image is fed into a GAN model directly to generate a meaning-normal image. On the receiver side, the secret image can be roughly recovered from the generated image. However, as the number of secret images increases, more training models are required. In the same year, Hu et al. [22] proposed a coverless steganography scheme based on the deep convolutional generative adversarial network (DCGAN), in which the secret message is mapped into the noise according to a mapping rule. Then, the noise is fed into the DCGAN to generate a natural fake image. On the receiver side, the receiver can use a well-trained extractor to reconstruct the noise from the fake image and then recover the secret message by the mapping rule. In 2019, Zhang et al. [23] provided a new direction for generative coverless steganography schemes. The secret message is hidden by converting the styles of the cover image. In 2020, Chen et al. [13] established a rule between the secret message and the multiple features of a face image. The secret message and a cover face image are fed into the StarGAN, a scalable model capable of learning mappings relationships between multiple domains, to generate a face image with new features.

Compared with the selective coverless steganography scheme, the generative coverless steganography scheme can directly generate diverse images without the need to collect a large number of real images. Furthermore, the generated fake images are less likely to violate the privacy and portrait rights of the image owner.

Since secret sharing can provide higher security than steganography, all coverless steganography schemes can be used to implement secret sharing. However, none of the existing coverless steganography schemes has an authentication mechanism, which leads to no way for participants to verify the authenticity of the share provided by each other. To solve this problem, we design a new CNN-based authentication mechanism and propose a novel steganographic secret sharing scheme.

3 Methods

In our work, we aim to combine the concept of coverless steganography with secret sharing and design a CNN-based authentication mechanism to improve the sensitivity of the proposed scheme. To achieve this goal, we use generator *Gen* and encoder *Enc* to encode secret shares into photorealistic image shares. Then, the generated image shares are distributed to n participants. After that, any $k(k < n)$ participants can work together to restore the secret message with the help of extractor *Ext*. Also, we further implement an authentication mechanism to verify the authenticity of the image shares presented in the decoding phase, which helps to improve the sensitivity of the proposed scheme. All experiments are implemented in PyTorch 1.7 on a PC with Intel® Core™ i9-9900K CPU and accelerated with an NVIDIA RTX 3090 GPU.

3.1 Steganographic secret sharing

Suppose a dealer embeds a secret into n image shares and distributes them to n participants. Any k of authentic participants can recover the secret only by sharing their image shares. The proposed Stego-SS scheme includes a secret sharing stage and a secret restoration stage. The flowcharts of secret sharing and secret restoration stages are shown in Fig. 1. In this figure, $k = 2$ and $n = 3$.

In the secret sharing stage, the secret message is split into n binary secret shares and modulated to obtain n noise-like secret shares of floating points first. Then, the noise-like secret shares are fed to the generator *Gen* to produce n image shares. A binary authentication code is generated by the dealer. After that, the authentication code is converted to two-dimensional and merged with image shares. Finally, n authentic image shares are produced and distributed to n participants by feeding the merged image shares to the encoder *Enc.*

In the secret restoration stage, participants decode their image shares into query codes using decoder *Dec* first and compare them with each other. If the number of similar codes exceeds the threshold, the corresponding image shares are considered authentic. After authentication, participants can extract the noise-like secret shares from their image shares using the extractor *Ext* and de-modulate the noise-like secret shares into secret shares. Finally, the embedded secret message can be obtained by merging any k secret shares.

To enable the whole scheme, as shown in Fig. 2, we have to design the generator *Gen*, encoder *Enc*, decoder *Dec*, and extractor *Ext* properly. This section includes four subsections: neural network training, network structures, secret sharing, and secret restoration. The preparation of four networks is introduced first. Then, details on the secret sharing and the secret restoration are presented. Table 1 shows the main notations used in the proposed scheme.

3.2 Neural network training

The training process of the proposed scheme can be decomposed into three training phases: internal training, external training, and adversarial training.

The internal training phase is trained first with network structures shown in Fig. 2a, b. The encoder *Enc* and decoder *Dec* are jointly trained. An image dataset is applied as the ground truth for training *Enc.* Also, the image share in this dataset is used to simulate the output of the generator *Gen* in the external training phase. Before feeding to *Enc*, the image share is merged with a pseudo-two-dimensional authentication code. The generation of the pseudo-two-dimensional authentication code and the merge process are omitted here. In this phase, we train *Enc* to generate an authentic image share that is close to the image share, while the two-dimensional authentication code can be successfully extracted by *Dec*.

In the external training phase, the generator *Gen*, the discriminator *Dis*, and the extractor *Ext* are jointly trained with the network structures shown in Fig. 2c–e. The noise vector produced by a random number generator is applied to simulate the noise-like secret share. We train *Gen* to generate an image share that is close to the ground-truth dataset like [24]. After that, the image share is merged with a pseudo-two-dimensional

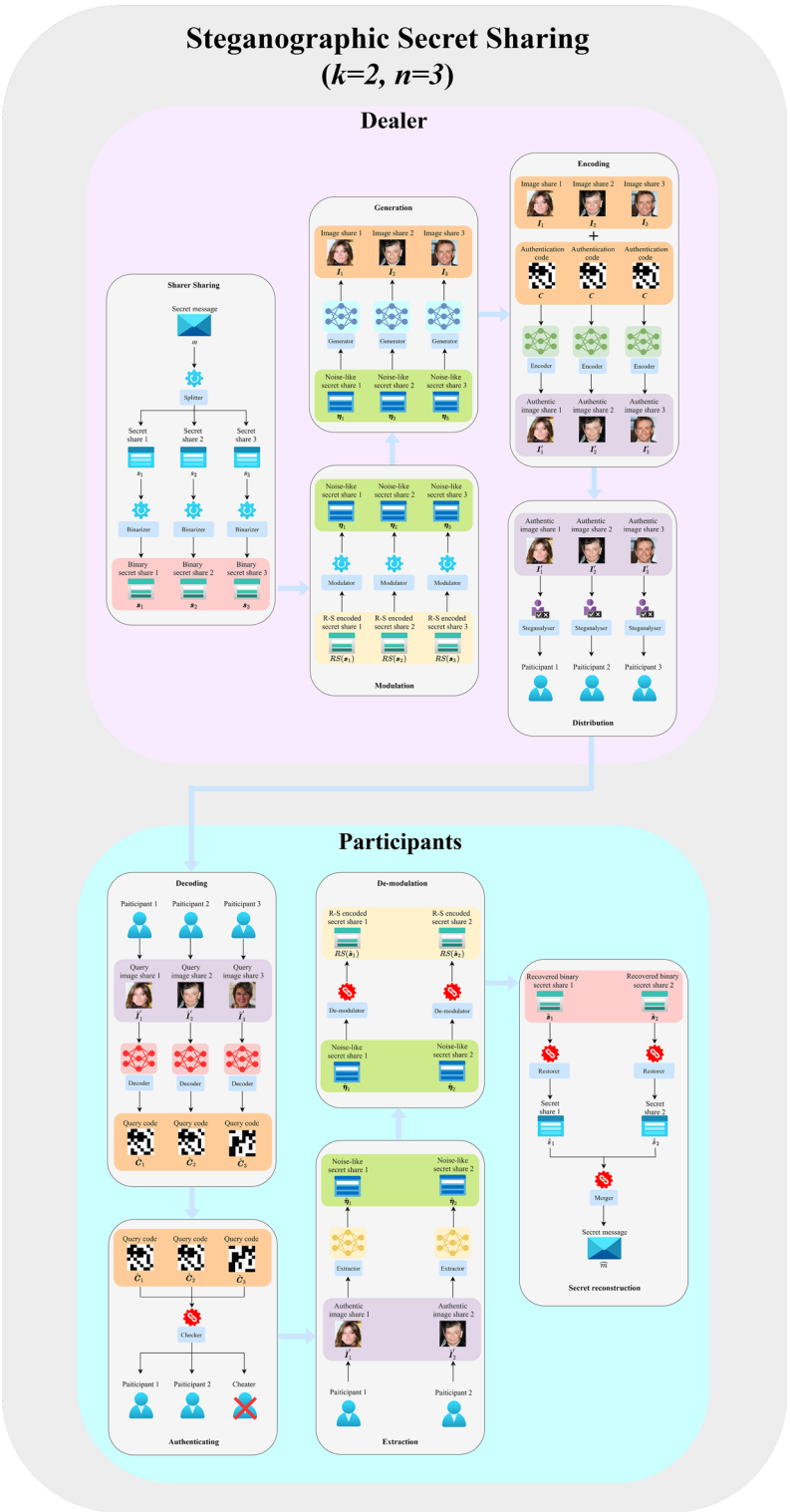

Fig. 1 Flowchart of the proposed scheme

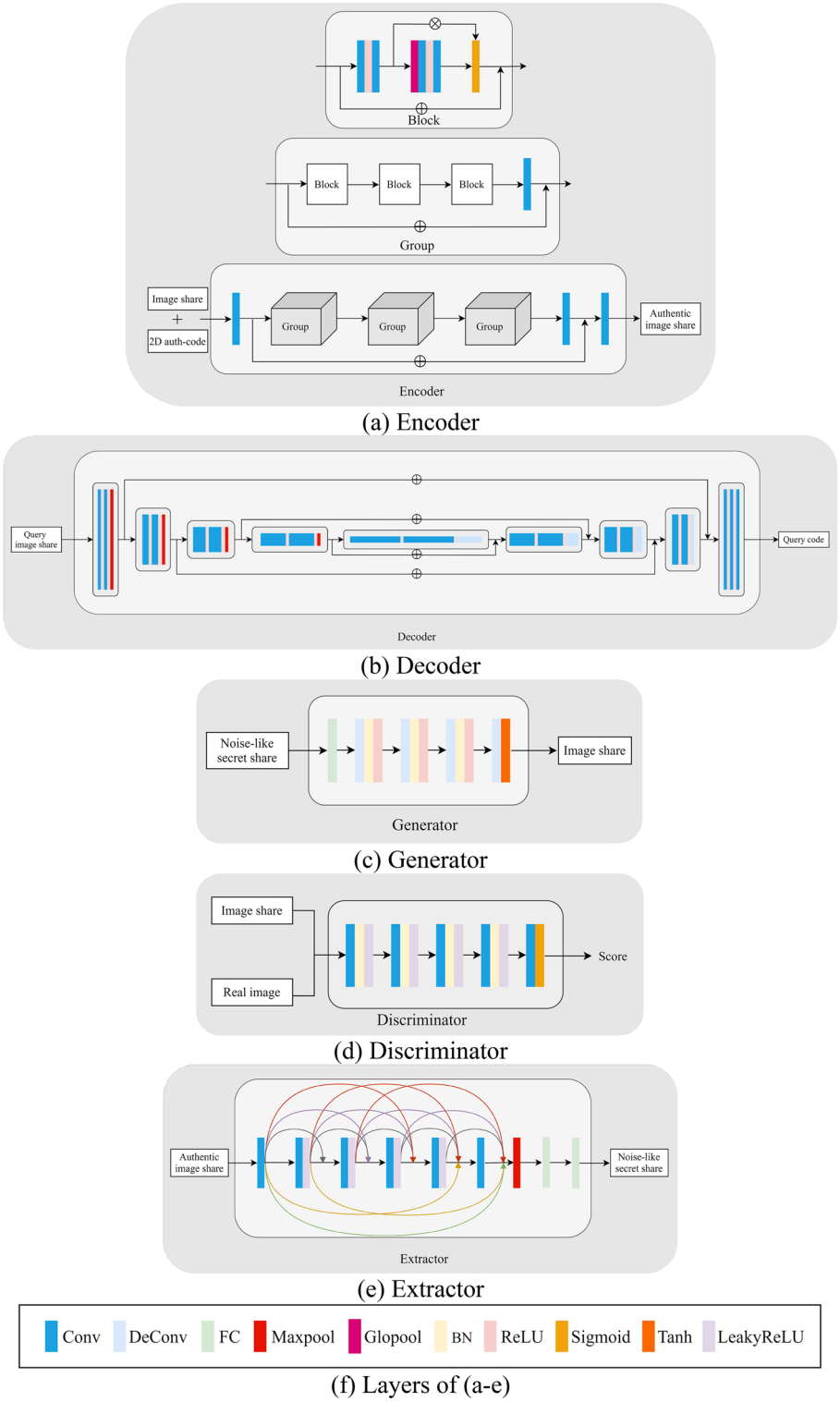

Fig. 2 Architectural details of models for steganographic secret sharing

Table 1 Descriptions of the main notations used in the proposed scheme

Symbols	Descriptions
Gen	Generator
Enc	Encoder
Dec	Decoder
Ext	Extractor
$\boldsymbol{m}$	$1 \times \ell_m$ sized binary secret message
m	Decimal secret message converted by $\boldsymbol{m}$
s	Decimal secret share
$\boldsymbol{s}$	$1 \times \ell_s$ sized binary secret share converted by s
$RS(\boldsymbol{s}_i)$	$1 \times \ell$ sized R–S encoded binary secret share
$\boldsymbol{\eta}$	$1 \times \ell$ sized modulated noise-like secret share
$\boldsymbol{I}$	$64 \times 64 \times 3$ sized image share
$\boldsymbol{c}$	1×64 sized binary authentication code
$\boldsymbol{C}$	64×64 sized binary two-dimensional authentication code
$\boldsymbol{I}'$	$64 \times 64 \times 3$ sized authentic image share
$\hat{\boldsymbol{I}}'$	$64 \times 64 \times 3$ sized query image share
$\hat{\boldsymbol{C}}$	64×64 sized binary two-dimensional query code
$\hat{\boldsymbol{c}}$	1×64 sized binary query code
$\hat{\boldsymbol{\eta}}$	$1 \times \ell$ sized extracted noise-like secret share
$RS(\hat{\boldsymbol{s}})$	$1 \times \ell$ sized de-modulated R–S encoded secret share
$\hat{\boldsymbol{s}}$	$1 \times \ell_s$ sized binary secret share recovered by R-S code
$\hat{s}$	Decimal secret share converted by $\hat{\boldsymbol{s}}$
$\hat{m}$	Recalculated decimal secret message
$\hat{\boldsymbol{m}}$	$1 \times \ell_m$ sized binary secret message converted by $\hat{m}$
ℓ	Length of R–S encoded binary secret share
ℓ_s	Length of binary secret share
ℓ_m	Length of binary secret message

authentication code and fed into the pertained *Enc.* Then, the output of *Enc*, the authentic image share, is fed into *Ext* to recover the noise vector.

In the adversarial training phase, the decoder *Dec* and extractor *Ext* are jointly trained to enhance the ability of models. In particular, we add a noise layer during the training process and then fine-tune the pre-trained models to increase the robustness of *Dec* and *Ext* against slight noise addition and color jittering.

The flowchart of the training phase is shown in Fig. 3, where the blue cube represents the model being trained and the red cube represents the model that is pre-trained and not involved in the training phase.

3.3 Network structures

To embed the two-dimensional authentication code into image share, we design an encoder *Enc* by following the residual channel attention network (RCAN) [25], which is a CNN for image super-resolution. As shown in Fig. 1a, *Enc* is composed of three groups, each of which consists of three blocks. Each block consists of four convolution layers, the first and third convolution layers are followed by rectified linear unit (ReLU) [26] activation function. The feature maps provided in each block are enhanced and reused. We apply L2-norm as the loss function of *Enc*, which is given by

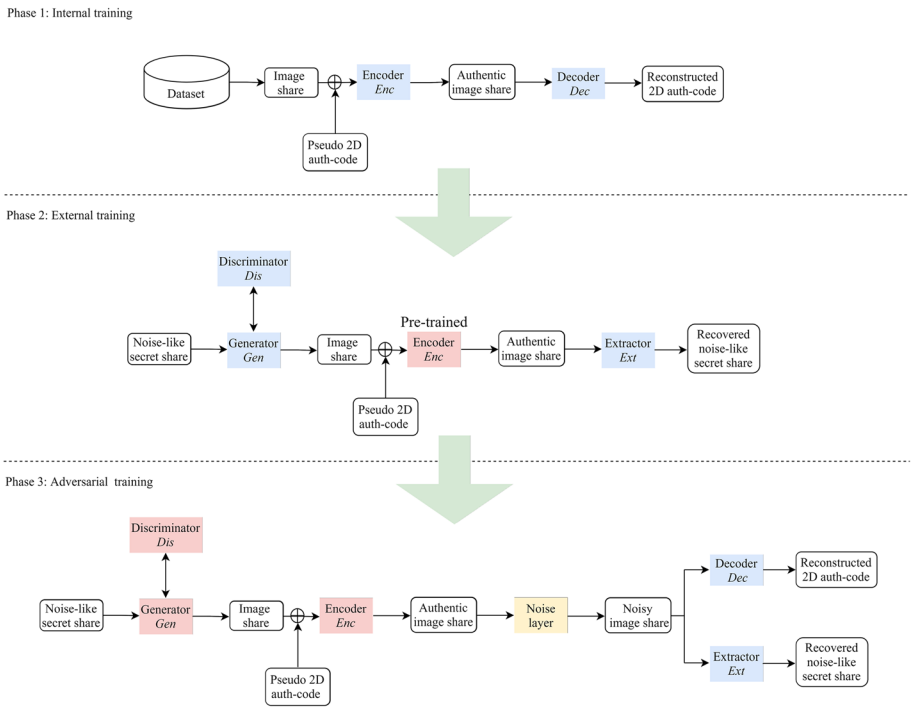

Fig. 3 Internal, external, and adversarial training phases

$$L_{Enc} = \sum_{i,j} \left(I_{ij} - I'_{i,j} \right)^2, \tag{1}$$

where I is the image share and I' is the authentic image share.

To reconstruct the embedded two-dimensional authentication code from the authentic image share, as shown in Fig. 1b, we design a decoder *Dec* by following the structure of U-net [27], which is a CNN for biomedical image segmentation. *Dec* is composed of a contracting path and an expansive path. The contracting path has four down-sampling steps, each of which consists of two convolution layers and a max pooling layer. The expansive path has four up-sampling steps, each of which has two convolution layers, a concatenation module, and a deconvolution layer. In the contracting path, each max pooling layer halves the size of the feature map and doubles the resulting channels. In the expansive path, each deconvolution layer doubles the size of the feature map and halves the resulting channels. All convolution layers are followed by the ReLU activation function. The loss function of *Dec* is a pixel-level binary cross-entropy with logits loss, which is given by

$$L_{Dec} = \sum_{i,j} -\left[C_{i,j} \times \log\left(\sigma\left(\hat{C}_{i,j} \right) \right) + \left(1 - C_{i,j} \right) \times \log\left(1 - \sigma\left(\hat{C}_{i,j} \right) \right) \right] \tag{2}$$

where C is the two-dimensional authentication code, $\hat{C}$ is the recovered two-dimensional query code, and $\sigma(x) = 1/(1 + \exp(-x))$.

The network structures of the generator *Gen* and the discriminator *Dis* follow the typical DCGAN [24]. The network structures of *Gen* and *Dis* are illustrated in Fig. 1c, d, respectively. *Gen* consists of a fully connected layer and four deconvolution layers. *Gen*

is used to learn the data distribution from the real image dataset in order to generate a natural image share. *Dis* consists of four convolution layers and a fully connected layer. *Dis* needs to discriminate whether the input image is real, while *Gen* needs to deceive *Dis* with the generated image share.

The loss function of *Gen* and *Dis* is an adversarial loss, defined as Eq. (3), where $Gen(\cdot)$ and $Dis(\cdot)$ denote the output of *Gen* and *Dis*, respectively; z is a noise vector sized 1×100, and x denotes the real image.

$$\min_{Gen} \max_{Dis} V(Gen, Dis) = E_{x \sim P_{data}(x)}[\log Dis(x)] + E_{z \sim P_z(z)}[\log(1 - Dis(Gen(z)))]. \tag{3}$$

when the training process is completed, *Gen* can produce an image share from a noise-like secret share that is difficult for *Dis* to distinguish from the real image. For the convenience of implementation, the generator loss and the discriminator loss in Eq. (3) are reformulated as

$$L_{Gen} = -[\log Dis(Gen(z))], \tag{4}$$

$$L_{Dis} = -[\log Dis(I) + \log(1 - Dis(Gen(z)))]. \tag{5}$$

To recover the noise-like secret share from the authentic image share, we design an extractor *Ext* by following the network structure of the densely connected convolutional network (DenseNet) [28], which connects each layer forward to all other layers. As shown in Fig. 1e, *Ext* comprises six convolution layers, the second to fifth convolution layers are followed by the leaky rectified linear unit (Leaky ReLU) [29] activation function. The feature maps provided by each convolution layer are reused, and the final feature maps are fed to two fully connected layers. We apply 1-norm as the loss function of *Ext*, which is given by

$$L_{Ext} = \sum_{i=1}^{n} \left| \eta_i - \hat{\eta}_i \right|, \tag{6}$$

where n is the length of the noise-like secret share, η is the input noise-like secret share, and $\hat{\eta}$ is the extracted noise-like secret share. To train the networks above, the adaptive moment estimation (Adam) algorithm [30] with $\beta 1 = 0.9$ and $\beta 2 = 0.999$ is applied until convergence.

3.4 Secret sharing inference

As the generator *Gen*, encoder *Enc*, decoder *Dec*, and extractor *Ext* are constructed and the training processes are completed, we are ready to share a secret by generating the n image shares. Referring back to Fig. 1, binary secret message $\boldsymbol{m}$ is converted to decimal m and split into n decimal secret shares $s_1, s_2, \ldots, s_n$. Each secret share is generated by (k, n)-secret sharing mechanism as illustrated in Eqs. (7–8) and converted to binary.

In order to ensure the correct calculation of the secret message, the Reed–Solomon code (R–S code) [31] can be applied in each binary secret share. Then, each binary secret share is encoded by R–S code and modulated to a noise-like secret share. The noise-like secret shares are severally fed to the pre-trained generator *Gen*. The

resulting image shares are concatenated with the two-dimensional authentication code and fed to the encoder *Enc* individually; therefore, the authentic image shares can be obtained. The detailed steps of the production of authentic image share are as follows.

Step 1. Split m to n secret shares $s_1, s_2, \ldots, s_n$ by

$$\begin{aligned} s_1 &= f(1), \\ s_2 &= f(2), \\ &\vdots \\ s_n &= f(n). \end{aligned} \tag{7}$$

$$f(x) = m + \alpha_1 x + \alpha_2 x^2 + \cdots + \alpha_{k-1} x^{k-1}, \tag{8}$$

where $\alpha_1, \alpha_2, \ldots, \alpha_{k-1}$ are random positive integers.

Step 2. Convert secret shares to binary $\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_n$ and encode them via R–S code.

Step 3. Modulate the R–S encoded secret shares $RS(\boldsymbol{s}_1), RS(\boldsymbol{s}_2), \ldots, RS(\boldsymbol{s}_n)$ to the noise-like secret shares $\boldsymbol{\eta}_1, \boldsymbol{\eta}_2, \ldots, \boldsymbol{\eta}_n$.

$$\begin{aligned} \boldsymbol{\eta}_1(j) &= \begin{cases} \text{rand}(-1, -\delta), \text{if } RS(\boldsymbol{s}_1(j)) = 0 \\ \text{rand}(+\delta, +1), \text{if } RS(\boldsymbol{s}_1(j)) = 1 \end{cases}, \\ \boldsymbol{\eta}_2(j) &= \begin{cases} \text{rand}(-1, -\delta), \text{if } RS(\boldsymbol{s}_2(j)) = 0 \\ \text{rand}(+\delta, +1), \text{if } RS(\boldsymbol{s}_2(j)) = 1 \end{cases}, \\ &\vdots \\ \boldsymbol{\eta}_n(j) &= \begin{cases} \text{rand}(-1, -\delta), \text{if } RS(\boldsymbol{s}_n(j)) = 0 \\ \text{rand}(+\delta, +1), \text{if } RS(\boldsymbol{s}_n(j)) = 1 \end{cases}, \end{aligned} \tag{9}$$

where the symbols $\boldsymbol{\eta}_1(j)$, $\boldsymbol{\eta}_2(j)$, and $\boldsymbol{\eta}_n(j)$ denote the j-th entry of the corresponding noise-like secret share; and the operation rand() generates a random floating-point number within the given range. In the proposed scheme, the range of j is $[1, \ell]$ and ℓ is set to 100.

Step 4. Generate image shares $\boldsymbol{I}_1, \boldsymbol{I}_2, \ldots, \boldsymbol{I}_n$ by feeding the noise-like secret shares to generator *Gen*.

$$\begin{aligned} \boldsymbol{I}_1 &= Gen(\boldsymbol{\eta}_1), \\ \boldsymbol{I}_2 &= Gen(\boldsymbol{\eta}_2), \\ &\vdots \\ \boldsymbol{I}_n &= Gen(\boldsymbol{\eta}_n). \end{aligned} \tag{10}$$

Step 5. Generate a 64-bit binary authentication code $\boldsymbol{c}$. Reshape $\boldsymbol{c}$ to 8×8 and upscale it to a 64×64 two-dimensional authentication code $\boldsymbol{C}$.

Step 6. Concatenate each image share with $\boldsymbol{C}$. Then, encode n concatenated image shares to authentic image shares $\boldsymbol{I}'_1, \boldsymbol{I}'_2, \ldots, \boldsymbol{I}'_n$ by feeding the concatenated images into encoder *Enc*.

$$\begin{aligned} \boldsymbol{I}_1' &= Enc(\boldsymbol{I}_1, \boldsymbol{C}), \\ \boldsymbol{I}_2' &= Enc(\boldsymbol{I}_2, \boldsymbol{C}), \\ &\vdots \\ \boldsymbol{I}_n' &= Enc(\boldsymbol{I}_n, \boldsymbol{C}). \end{aligned} \tag{11}$$

Step 7. Distribute the authentic image shares the secret image shares to participants $1, 2, \ldots, n$, respectively, if the authentic secret shares escape the steganalyzer's detection.

3.5 Secret restoration inference

After receiving the authentic image shares, none of the participants can extract the secret message alone. But, they can provide their image shares for authentication first. After authentication, any authentic k of them can cooperate to recover the secret message. Referring back to Fig. 1, the details of the secret restoration stage are as follows. Participants use decoder *Dec* to decode the two-dimensional authentication code embedded in their query image shares as the query code. If the number of similar codes exceeds the threshold, the corresponding query image shares are considered authentic. Then, any k participants use their trustworthy image shares to restore the secret shares and combine them to calculate the final secret message. The detailed steps of the overall restoration procedures are as follows.

Step 1: Decode two-dimensional query codes $\hat{\boldsymbol{C}}_1, \hat{\boldsymbol{C}}_2, \ldots, \hat{\boldsymbol{C}}_n$ from the corresponding query image shares $\hat{\boldsymbol{I}}_1^{'}, \hat{\boldsymbol{I}}_2^{'}, \ldots, \hat{\boldsymbol{I}}_n^{'}$ using decoder *Dec* and convert to 64-bit binary query codes $\hat{\boldsymbol{c}}_1, \hat{\boldsymbol{c}}_2, \ldots, \hat{\boldsymbol{c}}_n$.

$$\begin{aligned} \hat{\boldsymbol{C}}_1 &= Dec\left(\hat{\boldsymbol{I}}_1^{'}\right), \\ \hat{\boldsymbol{C}}_2 &= Dec\left(\hat{\boldsymbol{I}}_2^{'}\right), \\ &\vdots \\ \hat{\boldsymbol{C}}_n &= Dec\left(\hat{\boldsymbol{I}}_n^{'}\right). \end{aligned} \tag{12}$$

Step 2: Authenticate the binary query codes with each other. If the number of similar codes exceeds the threshold, the corresponding image shares are considered authentic.

Step 3: Extract the noise-like secret share $\hat{\boldsymbol{\eta}}_1, \hat{\boldsymbol{\eta}}_2, \ldots, \hat{\boldsymbol{\eta}}_k$ from any k of authentic image shares $\hat{\boldsymbol{I}}_1^{'}, \hat{\boldsymbol{I}}_2^{'}, \ldots, \hat{\boldsymbol{I}}_k^{'}$ using extractor *Ext*.

$$\begin{aligned} \hat{\boldsymbol{\eta}}_1 &= Ext\left(\hat{\boldsymbol{I}}_1^{'}\right), \\ \hat{\boldsymbol{\eta}}_2 &= Ext\left(\hat{\boldsymbol{I}}_2^{'}\right), \\ &\vdots \\ \hat{\boldsymbol{\eta}}_k &= Ext\left(\hat{\boldsymbol{I}}_k^{'}\right). \end{aligned} \tag{13}$$

Step 4: De-modulate the noise-like secret share $\hat{\boldsymbol{\eta}}_1, \hat{\boldsymbol{\eta}}_2, \ldots, \hat{\boldsymbol{\eta}}_k$ back to the R–S encoded secret shares $RS(\hat{\boldsymbol{s}}_1), RS(\hat{\boldsymbol{s}}_2), \ldots, RS(\hat{\boldsymbol{s}}_k)$, respectively.

$$\begin{aligned} RS(\hat{\boldsymbol{s}}_1(j)) &= \begin{cases} 0, \text{if } \hat{\boldsymbol{\eta}}_1(j) < 0 \\ 1, \text{if } \hat{\boldsymbol{\eta}}_1(j) \geq 0 \end{cases}, \\ RS(\hat{\boldsymbol{s}}_2(j)) &= \begin{cases} 0, \text{if } \hat{\boldsymbol{\eta}}_2(j) < 0 \\ 1, \text{if } \hat{\boldsymbol{\eta}}_2(j) \geq 0 \end{cases}, \\ &\vdots \\ RS(\hat{\boldsymbol{s}}_k(j)) &= \begin{cases} 0, \text{if } \hat{\boldsymbol{\eta}}_k(j) < 0 \\ 1, \text{if } \hat{\boldsymbol{\eta}}_k(j) \geq 0 \end{cases}. \end{aligned} \tag{14}$$

Step 5: Recover the R–S encoded secret shares to $\hat{\boldsymbol{s}}_1, \hat{\boldsymbol{s}}_2, \ldots, \hat{\boldsymbol{s}}_k$ and convert them to decimal $\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_k$.

Step 6: Combine the recovered secret shares $\hat{s}_1, \hat{s}_2, \ldots, \hat{s}_k$ to recalculate the secret message $\hat{m}$ by Eqs. (7) and (8).

Step 7: Convert $\hat{m}$ to binary $\hat{\boldsymbol{m}}$.

4 Experimental results and discussion

We analyze the performance of our scheme from several viewpoints. First, the experimental data for training the whole network are presented. Then, the accuracy of the decoder *Dec* and extractor *Ext* as well as the characteristics of the proposed scheme are analyzed. Finally, the sensitivity and robustness of the proposed scheme are tested and discussed. All experiments are implemented in PyTorch 1.7 on a PC with Intel® Core™ i9-9900 K CPU and accelerated with an NVIDIA RTX 3090 GPU.

4.1 Dataset

We train the models on a face dataset "CelebA," which contains 202,599 face images of 10,177 celebrity identities. By leveraging biological diversity, we can generate various face images which can mimic the real ones. Samples of ground-truth images from "CelebA" dataset are shown in Fig. 4.

Fig. 4 Samples of ground-truth images from "CelebA" dataset

4.2 Experimental setup

The internal training phase is trained on the face dataset, including training of the encoder *Enc* and decoder *Dec*. *Enc* can produce authentic image shares that are close to the image share. Besides, *Dec* is jointly trained to ensure the recoverability of the two-dimensional authentication code. Before feeding to *Enc*, the image is concatenated with the two-dimensional authentication code. The dataset of face images sized 64×64 is applied as the ground truth. The loss functions for training *Enc* and *Dec* are shown in Eqs. (1) and (2). The learning rate of *Enc* is set to 0.001, the batch size is 8, and the applied epoch is 200.

By feeding the output of *Enc* to *Dec*, the expected output is the embedded two-dimensional authentication code. In the training procedure of *Dec*, the learning rate is set to 0.002 and halved every 50 epochs.

The training of the external phase includes training of the generator *Gen*, the discriminator *Dis*, and the extractor *Ext*. The learning rates of *Gen* and *Dis* are set to 0.001 and 0.003, respectively, the batch size is 64, the dimension of the noise is 1×100, and the number of epochs is 200.

The objective of the extractor *Ext* is to recover the input noise-like secret share from the authentic image share generated by *Enc*. The entry value of the noise-like secret share ranges from -1 to $+1$. As mentioned above, the 1-norm of the error vector is applied as the loss function. In the training procedure of *Ext*, the learning rate is set to 0.0002 and the batch size is 64.

At the end of the internal and external training phases, *Dec* and *Ext* are jointly trained for 20 epochs. Besides, a noise layer is added during the training process, and then the pre-trained models are fine-tuned to increase the robustness against slight noise addition and color jittering. This noise layer randomly adds Gaussian noise with a mean of 0 and a variance of 58.52 to the image share or reduces the contrast of the image share with a contrast parameter of 0.5.

Figure 5 shows some samples of the generated authentic image shares with a resolution of 64×64.

4.3 Accuracy analysis

Each 8*8 sized block in the two-dimensional query code represents a 1-bit query code by pixel voting; if the number of black pixels is larger than the white pixels in it, this

Fig. 5 Samples of image shares generated by encoder

block represents 0, otherwise, it represents 1. The decoding accuracy of *Dec* is the key to authenticating the image shares, which is defined as

$$DecAcc = \left(1 - \frac{Ham(\boldsymbol{c}_i, \hat{\boldsymbol{c}}_i)}{||\hat{\boldsymbol{c}}_i||}\right) \times 100\%, \tag{15}$$

where $Ham(\mathscr{a}, \mathscr{b})$ represents the Hamming distance between vectors $\mathscr{a}$ and $\mathscr{b}$; $||\mathcal{V}||$ represents the length of vector $\mathcal{V}$. Figure 6 visualizes an example of the decoding ability of *Dec*, images from the left to right are two authentic image shares, the original two-dimensional authentication code, and two two-dimensional query codes decoded by *Dec* from two authentic image shares, respectively. The numerical data in the figure represent the decoding accuracy of the query code. As shown in the figure, the decoding ability of *Dec* is almost perfect, which is sufficient for a practical application of a 64-bit authentication code.

In order to ensure the correct recalculating of the secret message $\hat{m}$, R-S code is applied in each binary secret share. R–S code is a subset of error correction codes that offer the following guarantee: Given a binary secret share size of length ℓ_s, R–S code adds check bits to the share and extends the length of the share to ℓ (ℓ is set to 100 in the proposed scheme). Consequently, $\frac{\ell-\ell_s}{2}$ erroneous bits can be detected and recovered. Thus, the extraction accuracy of *Ext* is directly related to the length of the R–S code, which can be defined as

$$ExtAcc = \left(1 - \frac{Ham(s_i, \hat{s}_i)}{||\hat{s}_i||}\right) \times 100\%. \tag{16}$$

We would want the number of incorrect bits to be less than or equal to the number of bits we can recover:

$$(1 - ExtAcc) \times \ell \leq \frac{\ell - \ell_s}{2}, \tag{17}$$

where

$$\ell_s = \ell \times (2 \times ExtAcc - 1). \tag{18}$$

The higher the extraction accuracy, the less R–S code is needed. To know the effect of the gap value δ on *Gen*, we set δ from 0.1 to 0.9 and list the extraction accuracy and the number of bits required for the R–S code (i.e., $\ell - \ell_s$) in Table 2. The extraction accuracy increases slightly with increasing gap value δ. This is because the larger the δ, the smaller the floating range of the randomly generated noise values.

Fig. 6 A visual example of the recovery ability of decoder

Table 2 Extraction accuracy under different δ

δ	0	0.1	0.3	0.5	0.7	0.9
Accuracy	87%	90%	91%	92%	92%	92%
R–S code (bits)	26	20	18	16	16	16
ℓ_s(bits)	74	80	82	84	84	84

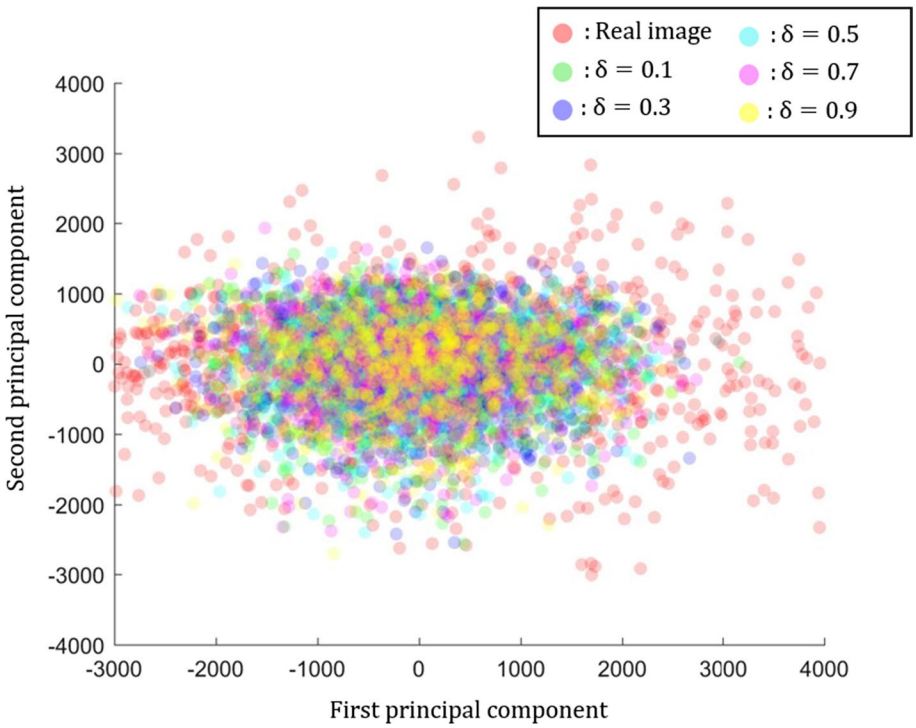

Fig. 7 Data distribution projected onto the first two principal components

4.4 Diversity analysis

Theoretically, the larger the δ, the lower the diversity of the generated images. To estimate and visualize the similarity between the distributions of real and generated images, we project the image samples onto a two-dimensional space via principal component analysis (PCA). The data points in Fig. 7 represent the first two principal components of real images and generated images with different δ settings ($\delta = 0.1$, 0.3, 0.5, 0.7, and 0.9). The distribution of the real image has the widest spread, indicating a wide variety of image content, while that of the generated images has a slightly narrower spread. As shown in the figure, the value of δ does not seem to have a significant effect on diversity. In addition, we do not feel much difference in diversity when we directly observe the actual generated images with different δ settings. Therefore, δ is set to 0.5 in subsequent experiments because a larger value of δ can slightly improve the extraction accuracy of *Ext* and does not affect the diversity of generated image too much.

Table 3 Comparison of embedding capacity

Scheme	Carrier size	Relative capacity (bits/pixel)	Extraction accuracy (%)
Ours (*Gen* + *Enc/Dec* + *Ext*)	64 × 64 × 3	**0.0081**	**92**
[23] (*Gen/Ext*)	256 × 256 × 3	0.0006	**92**
[22] (*Gen/Ext*)	64 × 64 × 3	**0.0081**	90
[13] (*Gen/Ext*)	128 × 128 × 3	0.0008	89

Bold values represent the best data in the comparison term

Table 4 Comparison of scheme characteristics

Scheme	Steganographic methodology	Application	Cheater detectability	Access threshold
Ours	Generative	Access control	✓	(*k*, *n*)
[23]	Generative	Access control	✓	(2, 2)
[22]	Generative	Convert communications	–	–
[13]	Selective	Convert communications	–	–
[14]	Selective	Convert communications	–	–
[17]	Selective	Convert communications	–	–
[19]	Selective	Convert communications	–	–

4.5 Characteristics analysis

The proposed Stego-SS scheme aims to distribute a secret into n image shares while the pixel values of the carrier images remain unchanged. The embedding capacity of such schemes is much lower than the conventional secret sharing schemes based on pixel modification. Thus, we compare our performance with some state-of-the-art coverless steganography schemes [14, 22] and a similar Stego-SS scheme [23] in the same dataset. In our scheme, each image share is embedded with 100 bits of the R-S encoded secret share. As shown in Table 3, the proposed scheme and Hu's scheme [22] have the largest embedding capacity because of the smaller carrier size and the larger amount of data embedded in a single carrier. As for the data extraction stage, both our scheme and the compared schemes use a CNN-based extractor to recover the embedded data. The extractor designed by us has a more complex network structure, which leads to slightly higher accuracy in extracting the embedded data than others [13, 22].

Table 4 shows the characteristics of the proposed schemes and the compared schemes. The secret sharing mechanism proposed in our scheme and Chen et al. [23] scheme can be applied to access control, but the coverless steganography schemes [13, 14, 17, 19, 22] can only be applied to convert communications, which represents a higher level of protection for the secret message in our and Chen et al.'s schemes. Also, both we and Chen et al. [23] provide an authentication mechanism, which is missing in [13, 14, 17, 19, 22]. Compared to Chen et al.'s scheme [23], the (2, 2)-secret sharing mechanism proposed in their scheme does not generalize to the (k, n)-secret sharing mechanism, which means that there is no possibility of recovering the secret message when a participant encounters a cheater. Thus, the proposed scheme is more applicable than the compared schemes.

4.6 Payload analysis

In the proposed scheme, each decimal secret share s_i consists of the decimal secret message m and random positive integer $\alpha_1, \alpha_2, \ldots, \alpha_{k-1}$ To estimate the effective number of bits per pixel that can be reliably conveyed under different access thresholds for the (k, n)-secret sharing mechanism, we refer to the embedding rate as given by

$$ER = \frac{\ell_m}{k \times (64 \times 64 \times 3)} (\text{bits/pixel}), \tag{19}$$

where $(64 * 64 * 3)$ represents the total pixels of each image share. ER is calculated based on the parameter k and the maximum value of α.

The maximum value of α is derived based on the choice of (k, n)-threshold and the maximum value of decimal secret message $m_{\max}$. As mentioned above, the length of each binary secret share $\boldsymbol{s}_i$ is ℓ_s bits, and therefore the maximum value of the decimal secret share $s_{\max}$ is $2^{\ell_s} - 1$, which implies

$$m + \alpha_1 x + \alpha_2 x^2 + \cdots + \alpha_{k-1} x^{k-1} \leq 2^{\ell_s} - 1. \tag{20}$$

In this case, the maximum payload (1-bit secret message), the maximum decimal value of secret message $m_{\max}$ is 1. For (k, n)-threshold the maximum value of x is n. Finally, we derive $\alpha_{\max}$ by

$$\begin{aligned}
&m_{\max} + \alpha_{\max} \times n + \alpha_{\max} \times n^2 + \cdots + \alpha_{\max} \times n^{k-1} = 2^{\ell_s} - 1; \\
&\alpha_{\max} \times \left(n + n^2 + \cdots n^{k-1} \right) = 2^{\ell_s} - 1 - m_{\max}; \\
&\alpha_{\max} \times \frac{n\left(1 - n^{k-1}\right)}{1 - n} = 2^{\ell_s} - 1 - m_{\max}; \\
&\alpha_{\max} = \frac{(2^{\ell_s} - 1 - m_{\max})(1 - n)}{n\left(1 - n^{k-1}\right)}.
\end{aligned} \tag{21}$$

Thus, the dealer can only choose α in the range of $[0, \alpha_{\max}]$. Suppose the selected range of α is $[0, \alpha_{up}]$, where $\alpha_{up} \leq \alpha_{\max}$, the maximum length of ℓ_m can be calculated as

$$\ell_m = \log_2 \left(2^{\ell_s} - 1 - \left(\alpha_{up} n + \alpha_{up} n^2 + \cdots + \alpha_{up} n^{k-1} \right) \right). \tag{22}$$

We list five cases ((1, 1), (2, 2), (2, 4), (3, 5), and (4, 6)) in Table 5. In these cases, the range of α is set to [0, 2048]. As shown in the table, the ER decreases as k increases. Therefore, the dealer can select suitable parameters depending on the actual situation.

Table 5 Embedding rate of different parameters

(k, n)	**(1, 1)**	**(2, 2)**	**(2, 4)**	**(3, 5)**	**(4, 6)**
ℓ_m(bit)	84	83	83	83	83
$\alpha_{\max}$	0	9.6×10^{24}	4.8×10^{24}	6.4×10^{23}	1.4×10^{22}
Range of α	[0, 2048]	[0, 2048]	[0, 2048]	[0, 2048]	[0, 2048]
ER (bits/pixel)	0.0068	0.0034	0.0034	0.0023	0.0017

4.7 Steganalysis

To evaluate the risk of the stego image being detected by the steganalyzer, three steganalysis models are selected to test the stego image. In this experiment, steganographic images using least significant bit (LSB) replacement under 1 bpp payload are generated and used to train three steganalysis models (Xu-Net [10], Zhu-Net [32], and SR-Net [33]) to distinguish between the steganographic image and the real image. After that, 100 stego images generated, respectively, in different schemes are tested by three models. The stego images in [13, 22, 23] are generated by the GANs (PGGAN, DCGAN, and StarGAN); the stego images obtained by LSB replacement are generated by encoding the cover image; the stego images obtained in the proposed scheme are generated by DCGAN and encoded by RCAN. The detection rate is utilized to detect the steganographic images under different schemes, which can be defined as

$$DetRate = \frac{ND}{NT} \times 100\%, \tag{23}$$

where ND represents the number of stego images being detected as the steganographic images and NT represents the number of total stego images. We expect the detection rate to be close to 50%, which indicates that models cannot correctly verify the stego images, resulting in random guesses for the detection results. As shown in Table 6, three models have high detection rates when testing real steganographic images (under 1 bpp payload and 3 bpp payload). However, in the proposed scheme and the compared schemes [13, 22, 23], images are generated using the neural network. When the neural network learns the data distribution of the real images, steganalysis models are unable to detect the authenticity of the stego images correctly, resulting in a detection rate floating around 50%. This experimental result shows that the images generated by the neural network are difficult to arouse the suspicion of steganalyzers during transmission.

4.8 Sensitivity and robustness analysis

In the proposed scheme, the two-dimensional authentication code is embedded into an image share to verify the identity of participants. In this section, we analyze the sensitivity and robustness of the proposed scheme.

Before extracting the secret message, the participants can authenticate each other by decoding their query image shares to two-dimensional query codes. So participants cannot be cheaters. To know the authentication results under different conditions, we tamper with the image shares using five different ways (inauthentic image share replacement,

Table 6 Detection rates by three steganalysis models

Scheme	Stego image	Xu-Net (%)	Zhu-Net (%)	SR-Net (%)
Ours	*Gen*(DCGAN) + *Enc*(RCAN)	**52**	**51**	53
[23]	*Gen*(PGGAN)	54	52	55
[22]	*Gen*(DCGAN)	54	**49**	**49**
[13]	*Gen*(StarGAN)	47	48	54
LSB (1 bpp)	Cover("CelebA") + *Enc*(LSB)	90	81	93
LSB (3 bpp)	Cover("CelebA") + *Enc*(LSB)	96	86	97

Bold values represent the best data in the comparison term

irrelevant image share replacement, patch removal, noise addition, and color jittering). An experimental result is shown in Fig. 8, where images in each row from the left to right are the authentic image share, the image share under various conditions, two two-dimensional query codes recovered by *Dec* from two image shares, and the authentication result, respectively. Numerical data in the two-dimensional query code represent the decoding accuracy of *Dec*. In the authentication result, the mismatched query codes are displayed in red and numerical data represent the matching rate which can be defined as

$$MatRate = \left(1 - \frac{Ham(\hat{\boldsymbol{c}}_i, \hat{\boldsymbol{c}}_j)}{||\hat{\boldsymbol{c}}_i||}\right) \times 100\%, \tag{24}$$

where $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$ represent the query codes of two participants.

The sensitivity for authentic image share and impersonated image share of the proposed authentication mechanism is evaluated first. In our hypothesis, there are two

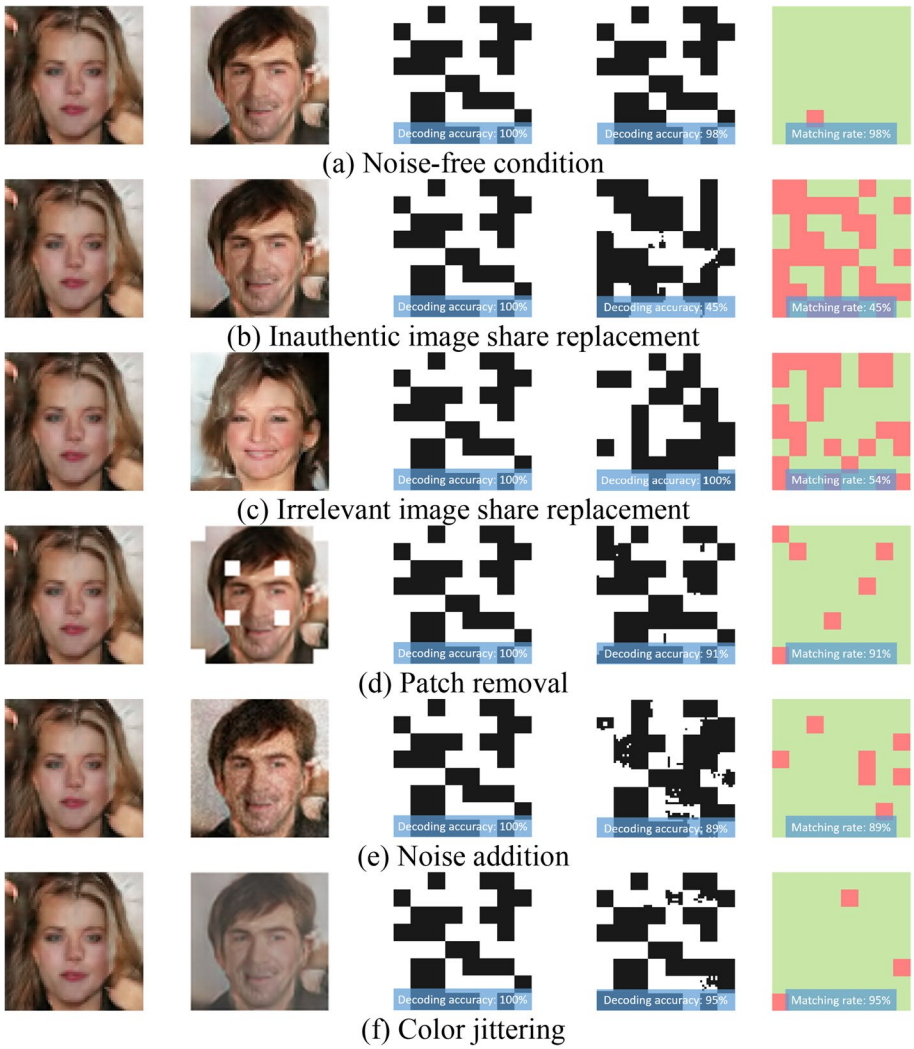

Fig. 8 Examples of sensitivity and robustness against cheater and distortion

kinds of impersonated image share: 1. inauthentic image share, which indicates that the image share is not embedded with the authentication code; 2. irrelevant image share, which indicates that the image share is embedded with the irrelevant authentication code. In Fig. 8 (a), when image shares are given by two honest participants, the matching rate of two query codes is 98%. When a cheater presents an impersonated image share to try to get other authentic image shares, the matching rate of two query codes is low (see Fig. 8b, c). These three results show that the proposed authentication mechanism is sensitive to impersonated image share. Without the knowledge about decoder *Dec*, the cheater has no clue to generate a trustworthy image share.

The authentication results under slight image distortion are listed in rows (d) to (f), respectively. When the quality of image share is corrupted due to improper image preservation by participants (e.g., noise addition, patch removal, and color jittering), the extracted query codes can still obtain a high matching rate (around 90%) with each other, which reflects the robustness of our scheme. In general, the proposed authentication mechanism is highly sensitive to the impersonated image share and highly robust to the slight distortion of image share.

To further verify the robustness of the proposed scheme, we remove the adversarial training phase of the training process and observe the decoding accuracy and data extraction accuracy. As shown in Table 7, without the adversarial training phase, the accuracy of the decoder and extractor is greatly affected by noise addition and color jittering. But after the adversarial training, the decoder and extractor have the ability to resist slight noise addition and color jittering. However, the adversarial training phase slightly affects the extraction accuracy of the extractor under a noise-free condition, so the dealer can decide whether the adversarial training phase is necessary according to the actual situation.

5 Conclusions

In this paper, we propose a novel steganographic secret sharing via AI-generated photorealistic images. First, we use the generator and encoder to encode the secret shares into the authentic image shares. After that, the authentic image shares are distributed to n participants. Finally, participants can verify each other and any k of authentic participants can collaborate to restore the secret message. The proposed scheme successfully applies the concept of coverless image steganography to implement an authenticable Stego-SS. Experimental results show that the pre-trained decoder and extractor can effectively restore the authentication code and the secret share. Also, sensitivity and robustness analysis confirm that the proposed scheme is resistant to various tampering attacks. In comparison with the state-of-the-art schemes, the embedding capacity of our

Table 7 Ablation study for adversarial training phase

	Decoding accuracy		Extraction accuracy	
Adversarial training phase	✓	×	✓	×
Noise addition	89%	67%	86%	63%
Color jittering	92%	65%	89%	87%
Noise-free condition	96%	96%	89%	92%

scheme outperforms other schemes. Also, the (k, n)-secret sharing and the authentication mechanism proposed in our scheme are more robust than that of other schemes. However, the extractor cannot recover 100% of the secret share embedded in a single image share, which leads us to sacrifice the embedding rate of the secret message in order to add the error correction code in the secret share. Thus, our future work will focus on designing better algorithms to improve the extraction accuracy of the proposed scheme.

Abbreviations

CNN	Convolutional neural network
Stego-SS	Steganographic secret sharing
HOG	Histogram of oriented gradients
SIFT	Scale-invariant feature transform
LDA	Latent Dirichlet allocation
GAN	Generative adversarial network
ACGAN	Auxiliary classifier generative adversarial network
DCGAN	Deep convolutional generative adversarial network
RCAN	Residual channel attention network
ReLU	Rectified linear unit
DenseNet	Densely connected convolutional network
Adam	Adaptive moment estimation
PCA	Principal component analysis
LSB	Least significant bit

Acknowledgements
Not applicable.

Author contributions
KG and C-CC contributed to conceptualization; KG contributed to methodology, software, investigation, data curation, writing—original draft preparation; J-HH, KG, and C-CC validated the study and visualized the study; KG and J-HH performed formal analysis; C-CC and IE contributed to resource and supervised the study; J-HH, KG, C-CC, and IE performed writing—review and editing; C-CC contributed to project administration. All authors have read and agreed to the published version of the manuscript.

Funding
This work was partially supported by KAKENHI Grants (JP16H06302, JP18H04120, JP20K23355, JP21H04907, and JP21K18023) from the Japan Society for the Promotion of Science (JSPS) and CREST Grants (JPMJCR18A6 and JPMJCR20D3) from the Japan Science and Technology Agency (JST).

Availability of data and materials
Not applicable.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 10 March 2022 Accepted: 9 October 2022
Published online: 12 December 2022

References

1. A. Shamir, How to share a secret. Commun. ACM **22**(11), 612–613 (1979)
2. G.R. Blakley, Safeguarding cryptographic keys. In 1979 International workshop on managing requirements knowledge (MARK), (1979), pp. 313–317.
3. T.-H. Chen, K.-H. Tsao, Visual secret sharing by random grids revisited. Pattern Recogn. **42**(9), 2203–2217 (2009)
4. C.-C. Chang, Y.-H. Chen, H.-C. Wang, 'Meaningful secret sharing technique with authentication and remedy abilities. Inf. Sci. **181**(14), 3073–3084 (2011)
5. C.-C. Chang, P.-Y. Lin, Z.H. Wang, M.C. Li, A sudoku-based secret image sharing scheme with reversibility. J. Commun. **5**(1), 5–12 (2010)
6. Y.J. Liu, C.-C. Chang, A turtle shell-based visual secret sharing scheme with reversibility and authentication. Multimed. Tools Appl. **77**(19), 25295–25310 (2018)
7. J.W. Cheng, X.H. Yan, L.T. Liu, Y.Y. Sun, F.Y. Xing, Comprehensive reversible secret image sharing with palette cover images. J. Inf. Secur. Appl. **68**, 103233 (2022)

8. L.L. Li, Y.L. Lu, L.T. Liu, Y.Y. Sun, J.Y. Wang, Practical secret image sharing based on the Chinese remainder theorem. Mathematics **10**(12), 1959 (2022)
9. Z.Y. Hua, Y.X. Wang, S. Yi, Y.C. Zhou, X.H. Jia, Reversible data hiding in encrypted images using cipher-feedback secret sharing. IEEE Trans. Circuits Syst. Video Technol. (Early Access) **32**(8), 4968–4982 (2022)
10. G. Xu, H.-Z. Wu, Y.-Q. Shi, Structural design of convolutional neural networks for steganalysis. IEEE Signal Process. Lett. **23**(5), 708–712 (2016)
11. J. Zeng, S. Tan, B. Li, J. Huang, Large-scale JPEG image steganalysis using hybrid deep-learning framework. IEEE Trans. Inf. Forens. Secur. **13**(5), 1200–1214 (2018)
12. C.-C. Chang, Adversarial learning for invertible steganography. IEEE Access **8**, 198425–198435 (2020)
13. X. Chen, Z. Zhang, A. Qiu, Z. Xia, N. Xiong, A novel coverless steganography method based on image selection and StarGAN. IEEE Trans. Netw. Sci. Eng. (2020). https://doi.org/10.1109/TNSE.2020.3041529
14. L. Qiang, X.Y. Xiang, J.H. Qin, Y. Tan, J.S. Tan, Y.J. Luo, Coverless steganography based on image retrieval of DenseNet features and DWT sequence mapping. IEEE Trans. Inf. Forens. Secur. **13**(5), 1200–1214 (2018)
15. Z.L. Zhou, H.Y. Sun, R. Harit, X.Y. Chen, X.M. Sun, Coverless image steganography without embedding. In International conference on cloud computing and security, (2015), pp. 123–132
16. Z.L. Zhou, Y. Cao, X.M. Sun, Coverless information hiding based on bag-of-words model of image. J. Appl. Sci. **34**(5), 527–536 (2016)
17. Z.L. Zhou, J. Wu, C.-N. Yang, X.M. Sun, Z.Q. Pan, Coverless image steganography using histograms of oriented gradients-based hashing algorithm. J. Internet Technol. **18**(5), 1177–1184 (2017)
18. S.L. Zheng, L. Wang, B.H. Ling, D.H. Hu, Coverless information hiding based on robust image hashing. In International conference on intelligent computing, (2017). pp. 536–547
19. X. Zhang, F. Peng, M. Long, Robust coverless image steganography based on dct and lda topic classification. IEEE Trans. Multimed. **20**(12), 3223–3238 (2018)
20. M.M. Liu, M.Q. Zhang, J. Liu, Y.N. Zhang, Y. Ke, Coverless information hiding based on generative adversarial networks. *arXiv* preprint (2017). http://arxiv.org/abs/1704.04861
21. X.T. Duan, H.X. Song, Coverless information hiding based on generative model (2018). http://arxiv.org/abs/1802.03528
22. D.H. Hu, L. Wang, W.J. Jiang, S.L. Zheng, B. Li, A novel image steganography method via deep convolutional generative adversarial networks. IEEE Access **6**, 38303–38314 (2018)
23. S.S. Chen, C.-C. Chang, I. Echizen, Steganographic secret sharing with GAN-based face synthesis and morphing for trustworthy authentication in IoT. IEEE Access **9**, 116427–116439 (2019)
24. A. Radford, L. Metz, S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks (2015). http://arxiv.org/abs/1511.06434
25. Y.L. Zhang, K.P. Li, K. Li, L.C. Wang, B.N. Zhong, Y. Fu, Image super-resolution using very deep residual channel attention networks. In The European conference on computer vision (ECCV), (2018), pp. 286–301
26. X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks. In the fourteenth international conference on artificial intelligence and statistics. JMLR workshop and conference proceedings, (2011), pp. 315–323
27. O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation. In International conference on medical image computing and computer-assisted intervention, (2015), pp. 234–241
28. G. Huang, Z. Liu, L. Van Der Maaten, K.Q. Weinberger, Densely connected convolutional networks. In IEEE conference on computer vision and pattern recognition, (2017), pp. 4700–4708
29. B. Xu, N. Wang, T. Chen, M. Li, Empirical evaluation of rectified activations in convolutional network (2015). http://arxiv.org/abs/1505.00853
30. D.P. Kingma, J. Ba, Adam: a method for stochastic optimization (2014). http://arxiv.org/abs/1412.6980
31. I.S. Reed, G. Solomon, Polynomial codes over certain finite fields. J. Soc. Ind. Appl. Math. **8**(2), 300–304 (1960)
32. R. Zhang, F. Zhu, J.Y. Liu, G.S. Liu, Deep learning hierarchical representations for image steganalysis. IEEE Trans. Inf. Forens. Secur. **15**, 1138–1150 (2019)
33. B. Boroumand, M. Chen, J. Fridrich, Deep residual network for steganalysis of digital images. IEEE Trans. Inf. Forens. Secur. **14**(5), 1181–1193 (2018)

Publisher's Note