Neural Computing and Applications (2022) 34:5743–5756
https://doi.org/10.1007/s00521-021-06725-w

ORIGINAL ARTICLE

Protection of image ROI using chaos-based encryption and DCNN-based object detection

Wei Song[1] · Chong Fu[1,2] · Yu Zheng[3] · Lin Cao[4] · Ming Tie[5] · Chiu-Wing Sham[6]

Received: 27 March 2021 / Accepted: 2 November 2021 / Published online: 12 January 2022

Abstract
Images always contain sensitive information, e.g., a clear face on a photo, which needs to be protected. The simple way is to encrypt the whole image for hiding "everything" securely, but it brings huge amounts of unnecessary encryption operations. Considering the most sensitive regions of an image, this paper focuses on protecting the important regions, thus reducing the redundant encryption operations. This paper employs the latest DCNN-based object detection model (YOLOv4) for choosing regions (i.e., multiple objects) and chaos-based encryption for fast encryption. We analyze object detection algorithm from a security perspective and modify YOLOv4 to guarantee that all areas of the detected objects are contained in the output regions of interest (ROI). Later, we propose a multi-object-oriented encryption algorithm to protect all the detected ROI at one go. We also encrypt the ROI coordinates and embed them into the whole image, relieving the burden of distributing ROI coordinates separately. Experimental results and security analyses show that all the detected objects are well protected.

Keywords Image encryption · Region of interest · Chaos · Object detection

1 Introduction

With the advancement of cloud storage and social media on the Internet, massive digital images containing sensitive information are created and shared daily. Image encryption protects raw images being accessed by an adversary who tries to intercept sensitive information. Compared with textual messages, digital images have the characteristics, such as a large amount of redundancy and bulk data. Popular block ciphers such as 3-DES, AES are designed to encrypt the textual information that consists of a set of words. Yet, they are not efficient enough to encrypt images [1]. Recently, applying chaos theory to image encryption has gained great attention due to the intrinsic properties of chaos, such as extreme sensitivity to initial conditions and pseudo-random behavior. Existing algorithms [2–8] encrypt an entire image without considering common cases in which only a certain region of the image is sensitive. We employ the notion of region of interest (ROI) for representing the sensitive region to be encrypted in an image. The ROI coordinates are referred to as ROI auxiliary information. In this paper, our research focuses on ROI-based encryption.

ROI-based encryption algorithms perform encryption operation on particular regions that have multiple detected objects. Two main problems existing in detection are (1) how to accurately locate the objects and identify their categories from a vast range of categories; and (2) how to accomplish the detection process efficiently. Many object

✉ Chong Fu
fuchong@mail.neu.edu.cn

1 School of Computer Science and Engineering, Northeastern University, Shenyang 110819, China

2 Engineering Research Center of Security Technology of Complex Network System, Ministry of Education, Shenyang, China

3 Department of Information Engineering, The Chinese University of Hong Kong, Sha Tin, Hong Kong SAR, China

4 School of Information and Communication Engineering, Beijing Information Science and Technology University, Beijing 100101, China

5 Science and Technology on Space Physics Laboratory, Beijing 100076, China

6 School of Computer Science, The University of Auckland, Auckland, New Zealand

detection algorithms aim to solve these two problems, which are classified into geometric representations, statistical classifiers, applying handcrafted feature descriptors, and discriminative classifiers. In 2012, a deep learning model called AlexNet [9] achieved a qualitative leap in classification accuracy in the Large Scale Visual Recognition Challenge (ILSVRC) [10]. Since then, lots of following works on object detection have been proposed [11–16]. These models are regarded as a promising tool to provide great help for ROI encryption [17–23].

1.1 Explicit motivation

The focus of most existing ROI encryption methods is mainly on the design of ROI encryption strategy. Albeit their success, current usage of object detection algorithms overlook *taking care of security flaws* brought by themselves. (a). From a security perspective, the output bounding box does not contain the entire object in most cases, leading to missing some edge areas of the object. Under this circumstance, undetected edge area of ROI still leaks some confidential information, as shown in Fig. 1. Therefore, we need to modify existing object detection algorithms to make the bounding box contain the entire object as much as possible. (b). The second flaw appears in the decryption process. To decrypt ciphered ROI correctly, ROI auxiliary information needs to be sent to the receiver side. Almost all the existing ROI-based encryption algorithms directly send the ROI auxiliary information to the receiver. However, any eavesdropper can easily know ROI positions of an targeted image. We need to encrypt the ROI auxiliary information before sending them. (c). The third concern is that most ROI encryption algorithms are designed for encrypting one object at one go. These schemes cannot be extended to a multi-object scenario since protecting multi-object in one image brings extra difficulty in processing overlapping areas.

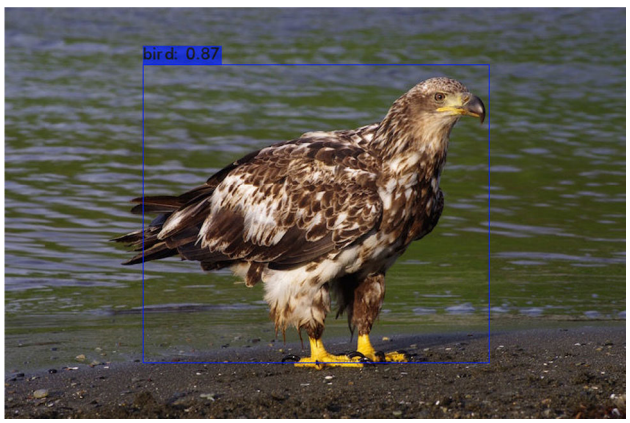

Fig. 1 An example of leaking edge area of the detected object

1.2 Our solution and contribution

To the ends aforementioned, we propose a multi-object encryption algorithm based on chaos and coordinates hiding. The latest object detection algorithm, YOLOv4 [24], which has high accuracy and speed, is employed as our building block. It is modified to contain the entire object of a ROI. Notably, our method is generic for applying to other DCNN-based object detection models. In the encryption process, all the pixels are marked in all ROI, thus enabling both overlapping and non-overlapping areas to be encrypted with the same number of rounds. This step guarantees same encryption strength for all ROI. Then, we encrypt the ROI auxiliary information and employ difference expansion to embed the ciphered coordinates into the entire image. The embedding positions are controlled by the data hiding key. Both the encryption key and the data hiding key are required to decrypt ciphered image correctly at a receiver side. The above three parts are integrated into a *hybrid-secure ROI solution*. To our best knowledge, we are the first to analyze object detection from a security perspective and propose a solution, protecting both ROI and its auxiliary information.

Summary of our contribution.

- From a security perspective, the bounding boxes output by our modified YOLOv4 can contain all areas of objects.
- We provide hybrid-protection to protect both sensitive ROI and its auxiliary information.
- Our encryption supports to protect multiple objects at one go, and guarantees same encryption strength to resist various security attacks.
- Embedding ROI auxiliary information removes the trouble of separately distributing the image and its ROI auxiliary information.
- For completeness in practical and theoretical parts, we conduct experimental evaluation and security analyses and the results shows the all ROI are well-protected.

The rest of this paper is organized as follows. Section 2 provides related works of ROI encryption, YOLO's evolution, and reversible data hiding. In Sect. 3, we explain the security concerns caused by current DCNN-based object detection algorithms. Then, we introduce our algorithm in details. Experimental results and security analyses are reported in Sect. 4. Section 5 concludes this paper.

2 Background and related works

This section comprises three parts: (1) Review of ROI encryption; (2) evolution of YOLO-based object detection and our choice; (3) introduction to reversible data hiding and our selected algorithm.

2.1 The ROI-based encryption algorithms

In [17], the target regions are detected by a geometric active contour model. In [18], authors divide the medical image into several blocks. They use a statistical measure on each block to determine whether it is ROI or not. One problem is over-broad identification of ROI caused by involving meaningless ROI, such as date and tags. ROI with irregular shapes are chosen and detected arbitrarily in [19]. Yet, we need an automatic detection tool that saves time and effort. In [20], the ROI is detected by [25]. After encryption, steganography is utilized to protect the significant bits of encrypted pixels. In the decryption process, a receiver obtains an encrypted image and its corresponding ROI boundary from the sender. The concern is no protection of ROI boundaries in the distribution process. In [21, 22], ROI is detected via a Gaussian mixture model and HOG feature extraction. In [23], YOLOv3 [26] and UNet [27] are used for ROI detection. But security flaws mentioned in Sect. 1 still remain.

For the ROI encryption, the security requirement should not be limited to the design of encryption algorithm. Otherwise, the encryption scheme is incomplete. Whether the detected ROI contains the entire object and the protection of ROI auxiliary information should also be considered. In this paper, our goal is to design a complete ROI encryption algorithm. The specific details of the proposed algorithms are stated in Sect. 3.1.

2.2 The evolution of YOLO-based object detection

YOLO treats object detection as a regression problem. The features of the entire image are used to make predictions. YOLO provides high accuracy and good generalization ability. Given YOLO, many improved algorithms are proposed, such as YOLOv2 [28], YOLOv3, YOLO3D [29], and YOLO-LITE [30].

Different from the above algorithms consuming lots of hardware resources or with unsatisfactory accuracy, YOLOv4 can achieve a good trade-off between accuracy and speed with only a 1080Ti GPU. Thus, we employ it as our tool for fast detection. Its output bounding boxes are modified to make the detected ROI contain the entire objects as much as possible. Detailed operations are explained in Sect. 3.2.

2.3 The reversible data hiding

Reversible data hiding means that we can embed/extract some confidential messages losslessly into/from cover media. In recent years, many excellent reversible data hiding algorithms have been proposed [31–38]. These works aim to embed as much data as possible in plain domain or encrypted domain. Yet, our encryption algorithm outputs a partially encrypted image, in which only ciphered ROI auxiliary information needs to be embedded. If we embed them into plain regions or encrypted regions of image, the embedding operation must be done on the basis of knowing the boundary information of ROI. However, a receiver cannot extract the ciphered ROI auxiliary information without ROI boundaries, since the sender has embedded them into the whole image.

Our goal is to design a data hiding algorithm that can embed/extract the secret information into/from any regions with low computation complexity. As the amount of data to be embedded is small, we want to divide them into many parts and randomly embed them into the whole image and the embedding position can be controlled by the data hiding key. Luckily, the reversible data hiding using a difference expansion (DE) [39] can meet our requirements. The method uses the redundancy between pixel pairs to embed data. As long as the embedding condition (no overflow and underflow problems after embedding a bit into the difference value of two adjacent pixels) is satisfied, we can embed the data regardless of the pixel pairs position. The detailed embedding process is described in Sect. 3.3.

3 The proposed encryption algorithm

To our best knowledge, almost no ROI-based encryption algorithms provide security by protecting ROI and its auxiliary information at the meantime. Current schemes are designed for encrypting only one object, which is not easy to extent to multi-object settings. To solve this problems, we design a scheme for a multi-object setting without leaking any ROI information.

The encryption pipeline is shown in Fig. 2a. We first use our modified YOLOv4 to get the ROI auxiliary information. Then we use Key_{ROI} and Key_{eninfo} to encrypt the ROI and its corresponding auxiliary information, respectively. At last, we use Key_{embed} to embed the ciphered ROI auxiliary information into the cipher-image for obtaining the marked cipher-image. The decryption process is roughly the reverse of encryption process as shown in Fig. 2b.

This section is organized as follows, we first analyze the security flaws in existing algorithms and describe how we solve them. In the second part, the multi-object-oriented encryption algorithm is introduced. Next, we explain how we embed the ciphered ROI auxiliary information into the cipher-image. The last part is the decryption process.

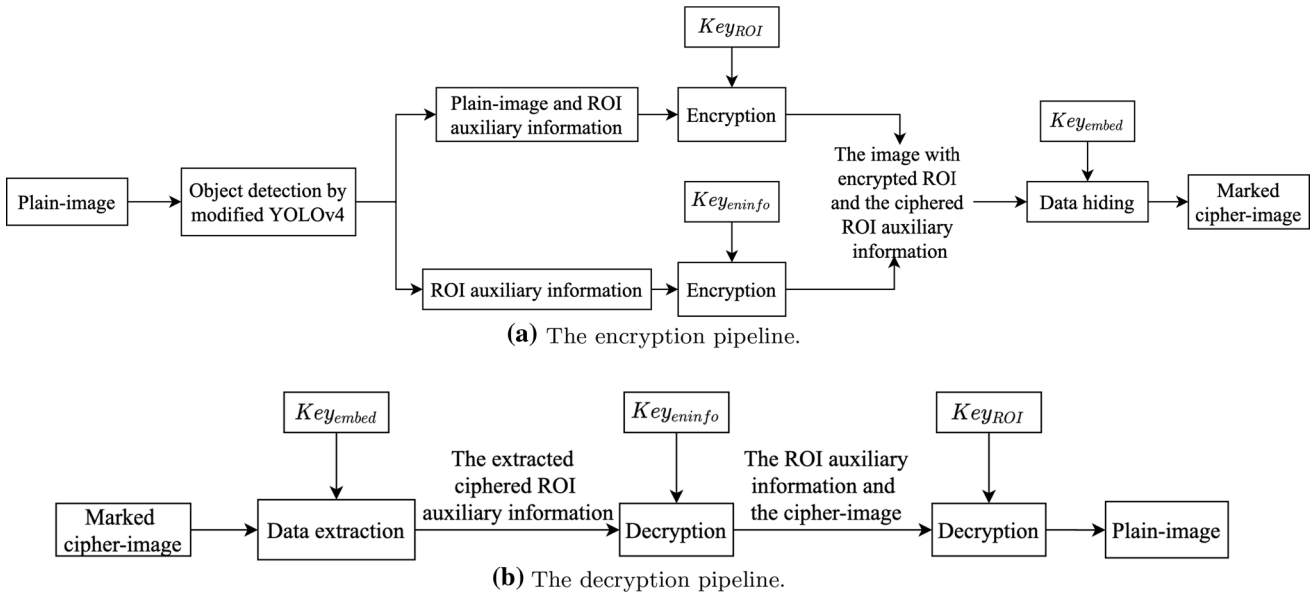

Fig. 2 The working pipeline of proposed algorithm

3.1 The modified YOLOv4

Let's take Fig. 3 as an example. Figure 3a–d shows the plain-image, YOLOv3's output, YOLOv4's output, and our modified YOLOv4's output, respectively. The outputs of YOLO-based object detection model are three bounding boxes, each of which contains five predictions. They involve the center point coordinates, the width/height of bounding box, and confidence scores, respectively. Here, confidence scores indicate the probability of containing an object in a box.

The three bounding boxes are selected from all the bounding boxes predicted by the model. The selection uses Non-Maximum Suppression (NMS). Suppose that there are nc object classes, $C = \{c_0, c_1, ..., c_{nc-1}\}$ and nb bounding boxes $B = \{box_0, box_1, ..., box_{nb-1}\}$. For object class c_0, the model sorts these boxes in descending order according to the probability that the object class contained in each box is c_0. Then, we get $BSorted = \{boxSorted_0, boxSorted_1, ..., boxSorted_{nb-1}\}$ Next, we calculate the IOU between $boxSorted_0$ and other boxes. If the IOU is larger than a user-defined threshold, the model determines that the two boxes predict the same object at the same location. The smaller of two predicted probability is set to 0, which means the corresponding box is deprecated. For the rest of boxes, $\{boxSorted_1, boxSorted_2, ..., boxSorted_{nb-1}\}$, repeat the above process to select the best box that predicts the same object. And for the other classes, $\{c_1, c_2, ..., c_{nc-1}\}$, repeat the above process to get the best boxes that predict a specific object with a specific class. As shown in Fig. 3b and c, the three bounding boxes selected by YOLOv3 and YOLOv4 are the best boxes that contain dog, person, and horse.

The detection results of Fig. 3b and c are acceptable in terms of accuracy and speed. However, there is a risk of leaking information of edge areas. The reason is that some bounding boxes with the edge areas of object are deprecated when performing NMS. To solve the security problems, the NMS process is modified to obtain the bounding box that contains entire object. We name this kind of bounding box as a greedy box.

The detailed process is described in Alg. 1. Here, nc, nb, $thresh$, and $detboxes$ represent the number of object classes, the number of detected bounding boxes, the value of user-defined threshold and the array of detected bounding boxes, respectively. The data type of each element in $detboxes$ is declared as a structure including two members, a bounding box with (x, y, w, h) and the probabilities of detected objects belonging to all classes. The process in the lines of 2-7 is the same as the second paragraph in this subsection. The core idea is to integrate the bounding boxes that predict the same object in the same position. For a specific object in a specific position, the gB is the greedy box. At first, the gB is initialized as the same size as the bounding box, which predicts the object best. Then we traverse the other bounding boxes. When gB meets another bounding box, b, it compares its boundary with b to get a smaller left boundary, a larger right boundary, a smaller top boundary, and a larger bottom

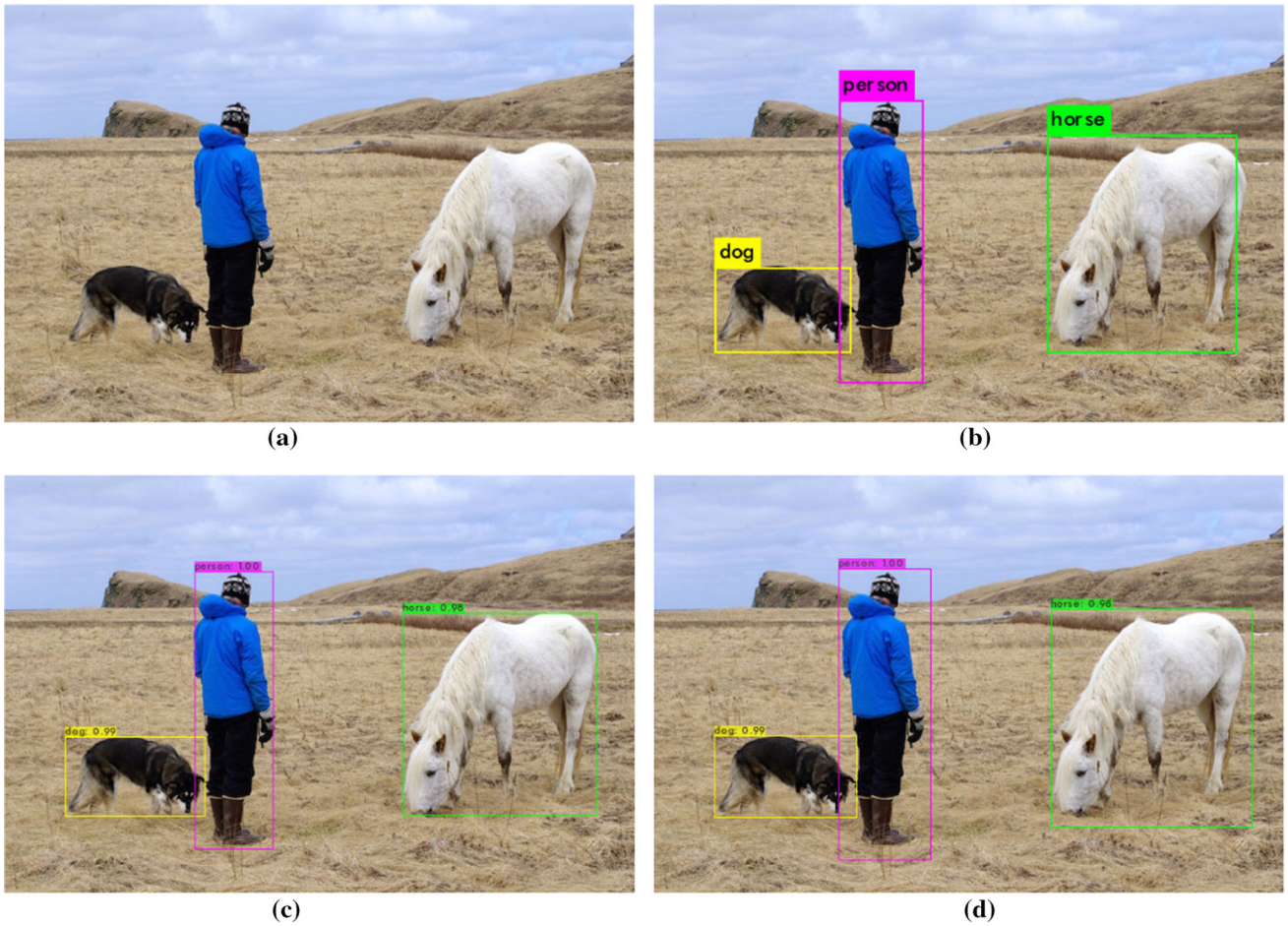

Fig. 3 The results of object detection. **a** The person image, **b** The detection result of YOLOv3, **c** The detection result of YOLOv4, **d** The detection result of our modified YOLOv4

boundary between itself and b. Note that the coordinate counter starts at 0 from left to right and top to bottom, so the coordinates values on the left and top are smaller. According to the new boundary values, gB calculate its new values of (x, y, w, h). Then, b is deprecated. After the traversal is over, the model outputs the greedy bounding box, gB, that contains the entire object. The detection result of our greedy detection algorithm is shown in Fig. 3d. Compared with YOLOv3 and YOLOv4, our detection algorithm successfully contains the entire object, so the experimental result is acceptable from a security perspective.

3.2 The encryption of multiple objects

The encryption process comprises two stages, permutation and diffusion. In the previous works, most encryption structures are designed for encrypting a single object. If we still utilize the existing algorithm to encrypt multiple objects, for each single object, the encryption influence is confined within itself and does not spread to other objects. As a result, the algorithm does not have the avalanche effect. In particular, tiny changes in a plaintext or the key should greatly impact the ciphertext. Another problem is that if we encrypt objects one by one, overlapping regions of multiple objects are repeatedly encrypted. This problem results in that the encryption strength of overlapping regions is different from that of non-overlapping regions.

To solve the above problems, we design an encryption algorithm for protecting multiple objects. Our permutation strategy can swap a pixel position with another pixel in any regions, achieving the total shuffling of pixels in all bounding boxes. And in diffusion stage, the encryption influence of one region can be spread to other regions .

Algorithm 1 The process of generating greedy bounding boxes.

Input: $nc, nb, thresh, detboxes$
Output: The greedy bounding boxes.

```
1: function GETGREEDYBOXES(nc, nb, thresh, detboxes)
2:     for k = 0 to nc − 1 do
3:         Sort the detected boxes detboxes
4:         for i = 0 to nb − 1 do
5:             if detboxes[i].prob[k] == 0 then
6:                 continue
7:             end if
8:             Bbox gB ← detboxes[i].box
9:             for j = i + 1 to nb − 1 do
10:                Bbox b = detboxes[j].box
11:                gbLx ← gB.x − gB.w/2
12:                gbRx ← gB.x + gB.w/2
13:                gbTp ← gB.y − gB.h/2
14:                gbBt ← gB.y + gB.h/2
15:                if IOU(detboxes[i].box, b) > thresh then
16:                    minLx ← min(gbLx, b.x − b.w/2)
17:                    maxRx ← max(gbRx, b.x + b.w/2)
18:                    minTp ← min(gbTp, b.y − b.h/2)
19:                    maxBt ← max(gbBt, b.y + b.h/2)
20:                    gB.x ← (minLx + maxRx)/2
21:                    gB.y ← (minTp + maxBt)/2
22:                    gB.w ← (maxRx − minLx)/2
23:                    gB.h ← (maxBt − minTp)/2
24:                    detBoxes[j].prob[k] ← 0
25:                end if
26:            end for
27:            detBoxes[i].box = gB
28:        end for
29:    end for
30: end function
```

The encryption scheme is depicted in Fig. 4 and the detailed encryption process is described in Alg. 2. Here *numROI* represents the number of ROI, *roiBoxes*. *ImgDat* is the 2-D image data and (x_0, y_0, z_0, u_0) is the initial condition of Jia system. To simplify the calculation of the coordinates during the encryption process, we first extract the pixels in each bounding box to a 1-D array, *rdt*. To avoid the pixels in overlapping regions being processed repeatedly, a flag for each pixel is set to record whether it has ever been read when extracting pixels. The above process corresponds to the lines 2-11 of Alg. 2. During encryption process, Jia system [40] is iterated to generate the keystreams for permutation and diffusion. Mathematically, the system is defined by,

$$\begin{cases} \dfrac{dx}{dt} = -a(x-y) + u, \\ \dfrac{dy}{dt} = -xz + rx - y, \\ \dfrac{dz}{dt} = xy - bz, \\ \dfrac{du}{dt} = -xz + du, \end{cases} \tag{1}$$

where a, r, b are the system parameters, and d is the control parameter. When $a = 10, r = 28, b = 8/3$ and $0.85 < d < 1.3$, the system exhibits chaotic behavior. And Runge–Kutta fourth-order method is used to solve Eq.(1), and the step length is 0.0005.

The whole permutation process is shown in the line 12-20. Jia system is pre-iterated for T_0 times to avoid the harmful effect of the transitional procedure, where T_0 is a user-defined value. Then, in the 1-D array, *rdt*, we perform pixel swapping strategy, which means that each pixel swaps positions with the a random pixel behind it. The permutation coordinates are extracted from generated chaotic sequences. The diffusion process is shown in the line 21-28. cdt_1 is a user-defined value, here is 128. L is the gray level, for a 24-bit RGB color image, $L = 256$.

The above process can be applied for several rounds to obtain a satisfactory encryption effect. The encryption effect is shown in Fig. 5, it can be seen that all parts of the objects are protected. Figure 6 can more intuitively depict how our encryption algorithm spreads the encryption influence of a certain region to other regions.

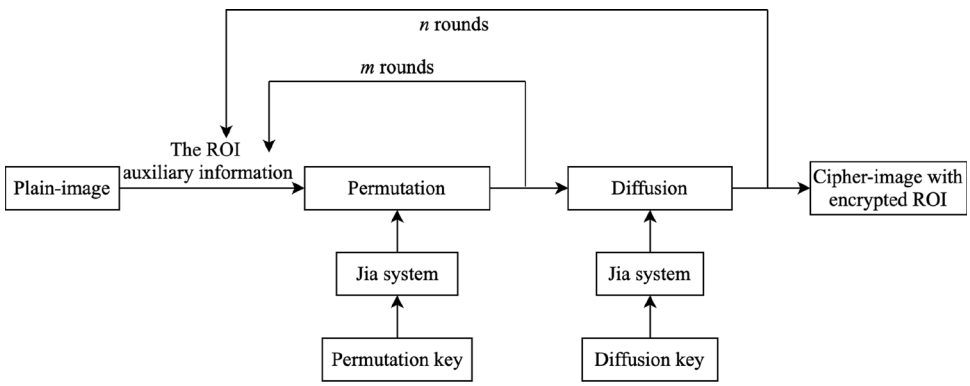

Fig. 4 The encryption structure

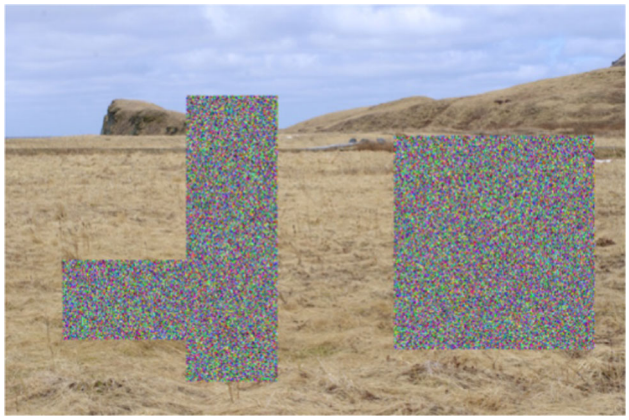

Fig. 5 The cipher-image with encrypted ROI

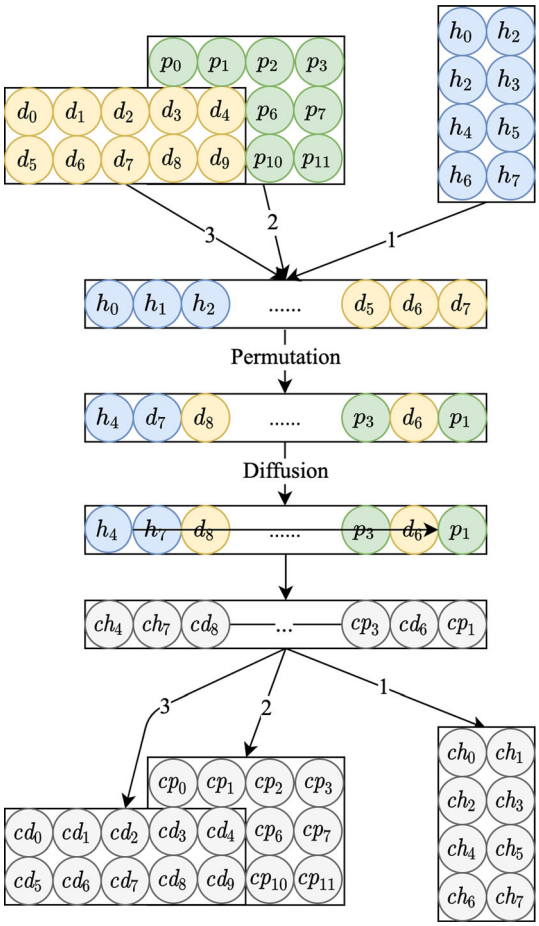

Fig. 6 The process of encrypting multiple objects

Algorithm 2 The proposed encryption algorithm for multiple objects.

Input: $numROI, roiBoxes, ImgDat, x_0, y_0, z_0, u_0$
Output: The cipher-image with encrypted ROI.

function ROIENCRYPTION($numROI, roiBoxes, ImgDat, x_0, y_0, z_0, u_0$)
  Set the *rdFlg* of all the pixels in *imgDat* to 0.
  for $i = 0$ to $numROI - 1$ **do**
    Check the *rdFlg* of each pixel *pix* in $roiBoxes[i]$
    if $rdFlg_{pix} == 0$ **then**
      Load it into the 1-D array *rdt*
      $rdFlg_{pix} \leftarrow 1$
    else
      continue
    end if
  end for
  Iterate Jia system for $len_{rdt} - 1$ times to generate chaotic sequences, *seqPm*
  for $i = 0$ to $len_{rdt} - 2$ **do**
    while $|seqPm_i| > 1$ **do**
      $seqPm_i \ /= 10$
    end while
    $Pos_i \leftarrow i + (1 + (int64)\,|seqPm_i| \times 10^{15}) \mod ((len_{rdt} - 1) - i)$
  end for
  $Pos_{i+1} \leftarrow len_{rdt} - 1$
  Traverse the array, *rdt*, and for each pixel in position *px*, swap the value of $rdt[px]$ and $rdt[Pos_{px}]$, and get shuffled data, *rsfdt*.
  Iterate Jia system for len_{roiDat} times to generate chaotic sequence, *seqDf*
  for $i = 0$ to $len_{rdt} - 1$ **do**
    while $|seqDf_i| > 1$ **do**
      $seqDf_i \ /= 10$
    end while
    $ksdf_i \leftarrow (int64)(|seqDf_i| \times 10^{15}) \mod L)$
    $cdt_i = ksdf_i \oplus \{[rsfdt_i + ksdf_i] \mod L\} \oplus cdt_{i-1}$
  end for
  Write *cdt* back to the image according to the ROI coordinates information
end function

3.3 The protection of ROI auxiliary information using reversible data hiding

In this subsection, we first give the basic concept of the reversible data hiding using DE, and present the idea of using DE to embed/extract the ROI auxiliary information.

The core idea of difference expansion is the integer Haar wavelet transform. Assume there is a pixel pair (x, y), we define the values of integer average, *intAver* and the value of difference, *diffV* by

$$intAver = \lfloor \frac{x+y}{2} \rfloor, diffV = x - y. \quad (2)$$

The inverse of Eq. (2) is

$$x = intAver + \lfloor \frac{diffV + 1}{2} \rfloor, y = intAver - \lfloor \frac{diffV}{2} \rfloor. \quad (3)$$

For an 8-bit gray image, the gray scale $L = 256$, so the range of pixel value is [0, 255], we have

$$\begin{aligned} 0 \le intAver + \lfloor \frac{diffV + 1}{2} \rfloor \le 255, \\ 0 \le intAver - \lfloor \frac{diffV}{2} \rfloor \le 255. \end{aligned} \quad (4)$$

And Eq. (4) is equivalent to

$$\begin{cases} |diffV| \le 2(255 - intAver), if\ 128 \le intAver \le 255 \\ |diffV| \le 2intAver + 1, if\ 0 \le intAver \le 127. \end{cases} \quad (5)$$

Next we embed a bit b into $diffV$, and we get new $diffV_{new} = 2 \times diffV + b$. According to Eq. (5), if

$$|diffV_{new}| \le min(2(255 - intAver), 2intAver + 1),$$

then we call such $diffV$ is expandable, and we can embed ciphered ROI auxiliary information into the difference value of such pixel pairs. The new values of pixel pair are

$$x' = intAver + \lfloor \frac{diffV_{new+1}}{2} \rfloor, y' = intAver - \lfloor \frac{diffV_{new}}{2} \rfloor. \quad (6)$$

During the extraction process, we extract the bit from the new difference value, then we get the original difference value. And we can use Eq.(7) to restore the original pixel values,

$$x = x' - \lfloor \frac{diffV + 1}{2} \rfloor, y' = intAver + \lfloor \frac{diffV + 1}{2} \rfloor. \quad (7)$$

The process of embedding ciphered ROI auxiliary information is described in Alg 3. Here, *roiInfo* represents the data of plain ROI auxiliary information, *cimgRDat* is the cipher-image with encrypted ROI. H and W are the height and width of image. The length of image is len_{img}, whose value is $3 \times H \times W$. lgx_0 is the initial value of logistic map [41], which is defined by

$$x_{n+1} = \mu x_n(4 - x_n), x_n \in (0, 1) \quad (8)$$

where x_n is the state variable, and μ is control parameter whose range is (0, 4]. When $\mu = 4$, the logistic map has the best pseudo-randomness.

The data type of each ROI coordinate value is declared as integer, which needs 64 bits to represent. They are stored in *roiInfo* in bytes. Next, a one-bit bitmap of the image with ciphered ROI is generated. If the difference value of a pixel pair meets the requirement for embedding a bit, the value in its corresponding position of the bitmap is set to 1, otherwise 0. After the encryption of *roiInfo*, the ciphered data are embedded into the image bit by bit according to the bitmap. The embedding position is determined by the current values of logistic map and the embedding interval.

Algorithm 3 The embedding process of encrypted ROI information.

Input: $roiInfo, cimgRDat, len_{img}, W, lgx_0$
Output: The marked cipher-image with embedded ROI auxiliary information.

1: **function** EMDINFO($roiInfo, cimgRDat, len_{img}, W, lgx_0$)
2: Use "Permutation-Diffusion" network and logistic map to encrypt *roiInfo*, and get *cipherInfo*
3: $emdInter = len_{img}/len_{info}$
4: **for** $i = 0$ to $len_{info} - 1$ **do**
5: **for** $j = 0$ to 7 **do**
6: $emdBit = (cipherInfo[i] >> j)\&0x01$
7: Iterate logistic map to get a new value, lgx_{new}
8: $emdPos = i \times W + (int64)(lgx_{new} \times 10^{15})$ mod $emdInter$
9: **while** $bitMap[emdPos]! = 1$ **do**
10: $emdPos{+}{+}$
11: **end while**
12: Embed the $emdBit$ into the current pixel pair using DE
13: **end for**
14: **end for**
15: **end function**

3.4 The decryption process

The decryption process is roughly the reverse of encryption process as shown in Fig. 2b. The decryption result is shown in Fig. 7. Particularly, the reverse of line 27 in Alg. 3 is given by,

$$rsfdt_i = \{ksdf_i \oplus cdt_i \oplus cdt_{i-1} + L - ksdf_i\} \text{ mod } L. \quad (9)$$

Fig. 7 The decryption result of our proposed algorithm

4 Experimental results and security analysis

4.1 Key space analysis

For the proposed algorithm, the key space comprises four initial conditions of Jia system. Their data type is all declared as double precision, which needs 53 bits to represent. So the key space is $2^{53\times4} = 2^{212}$. It can be considered secure to resist the brute force attack as the key space is larger than 2^{100} [42].

4.2 Statistical attack

4.2.1 Histogram analysis

From a qualitative perspective, we carry out histogram analysis to evaluate the frequency distributions of pixel values in three plain-regions and their corresponding cipher-regions. Figure 8a–c and d–f depicts the 3D histograms of three plain-regions and their corresponding cipher-regions. We can see that, compared with plain-regions, the frequency distributions of the cipher-regions are almost uniform. It means that our algorithm has good performance in masking the pixel distribution.

4.2.2 Information entropy analysis

From a quantitative perspective, information entropy is used to measure the randomness and unpredictability of three plain-regions and their corresponding cipher-regions. Mathematically, it is defined by

$$H(inS) = -\sum_{i=0}^{N-1} P(inS_i)\log_2^{P(inS_i)}, \tag{10}$$

where inS represents an information source which contains N possible values $\{inS_0, inS_1, ..., inS_{N-1}\}$ and the probability of inS_i is $P(inS_i)$. If inS is a random information source, its information entropy is $\log_2^N$. The gray level of the test image is 256, so the information entropy of three cipher-regions should be close to 8. Table 1 lists the information entropy of three plain-regions and their corresponding cipher-regions. From this table, we can see that the information entropy of three cipher-regions are very close to 8, indicating that the pixel distributions of three plain-regions are successfully hidden.

4.2.3 Correlation of adjacent pixels analysis

Pixels usually have similar values with their neighbors, this is a sign of strong correlations among them. An effective encryption algorithm must eliminate the correlation, otherwise the attacker can easily predict the pixel values of certain region by some simple predictor, such as MED predictor and GAP predictor. The strength of correlation can be measured by calculating the correlation coefficient among adjacent pixels. The calculation method is defined by

$$r_{xy} = \frac{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{(\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2)(\frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2)}}, \tag{11}$$

where x_i, y_i represent the values of two adjacent pixels, $\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$, $\bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i$, and N is the number of

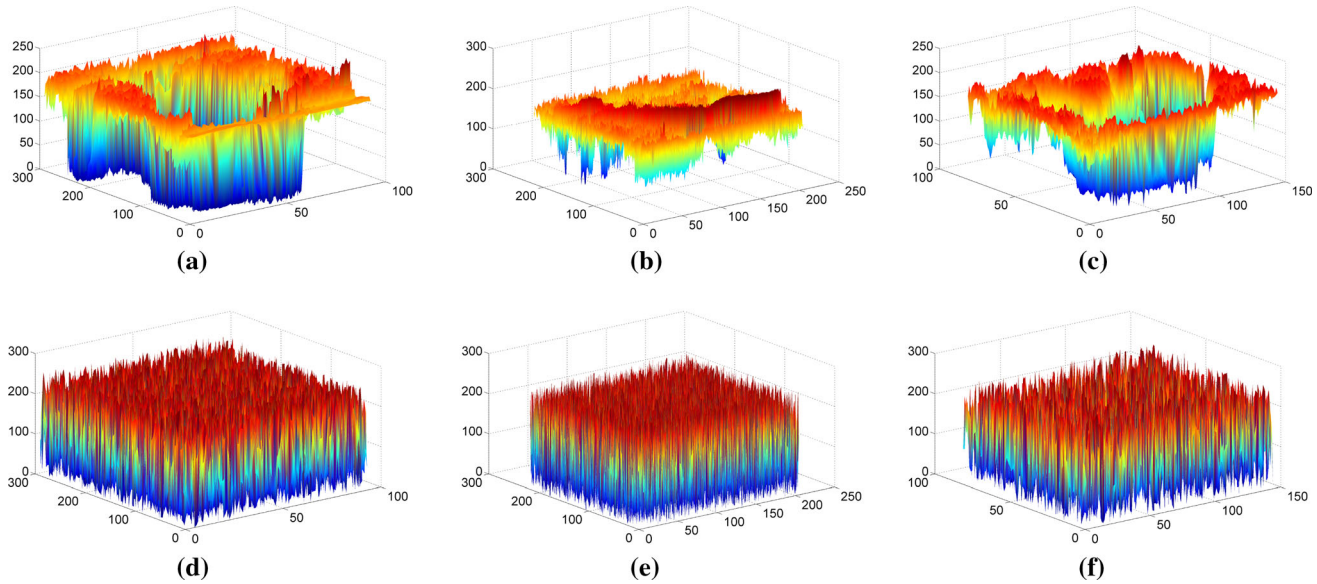

Fig. 8 The histograms of three plain-regions and their corresponding cipher-regions. **a–c** are the histograms of the plain-person, plain-horse and plain-dog, respectively. **d–f** are the histograms of the cipher-person, cipher-horse and cipher-dog, respectively

Table 1 The information entropy of the three lain-regions and their corresponding cipher-regions

Object Name	Plain-region	Cipher-region
Person	7.464343	7.991828
Horse	7.205766	7.995847
Dog	7.389903	7.983977

sampled pixel pairs. In the three color channels of each plain-region and its corresponding cipher-region, 5000 pairs of neighboring pixels are sampled in horizontal, vertical, and diagonal directions. And Table 2 reports the test results. From this table, we can see that the correlation coefficients are close to 1 in each plain-region, while those of each cipher-region are close to 0. It implies that our encryption algorithm successfully decorrelates the strong correlation in each plain-region.

Scatter diagram is usually used to analyze the correlation among adjacent pixels from a qualitative perspective. These pixel pairs sampled from the red channel of each region are plotted to 3D scatter diagrams, as depicted in Fig. 9. On the X-axis of each scatter diagram, $x = 0, 1, 2$ represent the horizontal, vertical, and diagonal directions, respectively. Then, each sampled pixel pair (x_i, y_i) is plotted as a point in the Y-Z plane. The values of x_i, and y_i determine the positions on the Y-axis and Z-axis, respectively. Figure 9a, c, and e depicts the scatter diagrams of plain-person, plain-horse and plain-dog, respectively. We can see that, in each Y-Z plane, most points lie along the diagonal line, showing a strong correlation among neighboring pixels in plain-regions. Figure 9b, d, and f depict the scatter diagrams of their corresponding cipher-regions. We can see that the distribution of these points is evenly cover the entire Y-Z plane. Similar results can be obtained for the other two color channels in each region. This phenomenon shows the weak correlation among adjacent pixels in three cipher-regions.

The test results in Sects. 4.2.1– 4.2.3 show that our ROI encryption algorithm can resist statistical attack.

4.3 Differential attack

To resist differential attack, if we input two plain-images with only 1 bit difference in one of three ROI, the corresponding cipher-regions in two output cipher-images should be completely different.

There are two criteria, *NPCR* (the number of pixel change rate) and *UACI* (the unified average changing intensity), for measuring the degree of difference between two images/ROI with same size. *NPCR* is defined by

$$NPCR = \frac{\sum_{i=1}^{W}\sum_{j=1}^{H} Dffe(i,j)}{W \times H} \times 100\%, \tag{12}$$

where

$$Dffe(i,j) = \begin{cases} 0 & if\ R_1(i,j) = R_2(i,j), \\ 1 & if\ R_1(i,j) \neq R_2(i,j). \end{cases} \tag{13}$$

UACI is defined by

$$UACI = \frac{\left[\sum_{i=1}^{W}\sum_{j=1}^{H} \frac{|R_1(i,j) - R_2(i,j)|}{2^L - 1}\right]}{W \times H} \times 100\%, \tag{14}$$

For two random images/ROI (gray level $L = 256$), the theoretical values of *NPCR* and *UACI* are 99.609% and 33.464%, respectively.

We first use secret key to encrypt the three plain-regions, and get their corresponding cipher-regions. Then, we randomly select a pixel whose coordinates are (501, 70) in plain-person, and modify the value of its red channel from 129 to 130. Next, we still use the same key to encrypt three plain-regions and get their corresponding cipher-regions. Finally, we calculate *NPCR* and *UACI* between two sets of

Table 2 The correlation coefficients of three plain-regions and the corresponding cipher-regions

Object name	Direction	Plain-region			Cipher-region		
		R	G	B	R	G	B
Person	Horizontal	0.973155	0.952966	0.960383	−0.004518	−0.002726	0.013430
	Vertical	0.962879	0.943667	0.967607	−0.021939	0.002643	0.031336
	Diagonal	0.948099	0.917047	0.945104	−0.016485	−0.000733	−0.005509
Horse	Horizontal	0.925748	0.942939	0.959856	−0.000272	0.007523	−0.025988
	Vertical	0.921298	0.935037	0.954211	−0.005799	0.014574	0.000293
	Diagonal	0.887695	0.904766	0.930582	−0.011644	−0.001528	0.014441
Dog	Horizontal	0.962245	0.957855	0.939610	0.028987	−0.019884	−0.006741
	Vertical	0.967799	0.963566	0.945931	0.019779	−0.050795	0.031677
	Diagonal	0.945600	0.939755	0.913779	0.003471	−0.002232	0.014529

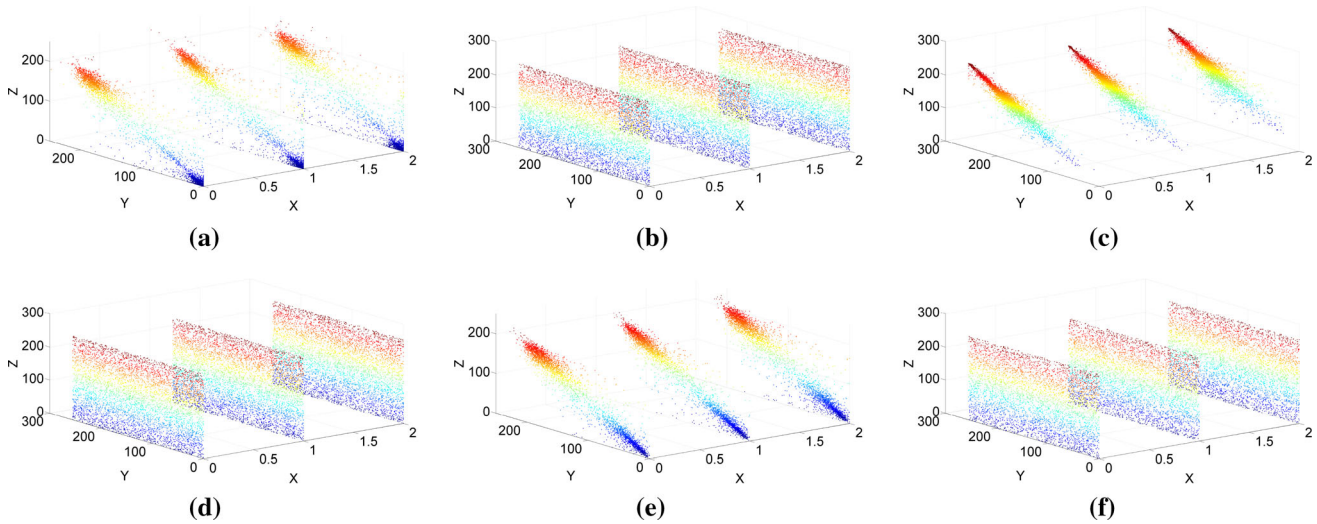

Fig. 9 The scatter diagrams of three detected regions in red channel. **a** and **b** are the scatter diagrams of plain-person and cipher-person. **c** and **d** are the scatter diagrams of plain-horse and cipher-horse. **e** and **f** are the scatter diagrams of plain-dog and cipher-dog

cipher-regions. The test results are reported in Table 3, from which we can see that the values of two criteria are very close to theoretical values. It implies that the two sets of cipher-regions are completely different, demonstrating the strong ability of resisting differential attack. Three rounds of encryption are used to achieve the encryption effect.

4.4 Key sensitivity analysis

A well-designed encryption algorithm should be extremely sensitive to the secret key. The key sensitivity is tested using the most extreme case. For each test case, only the least significant bit is changed in a key component, and the other three key components remain unchanged. Then, each modified secret key is used to encrypt/decrypt the plain-text/ciphertext produced by the original key. If the proposed algorithm has good key sensitivity in encryption process, the output cipher-regions corresponding to different keys should be completely different. And the corresponding case in decryption process is that the cipher-regions cannot be correctly decrypted with the wrong key.

4.4.1 Encryption key sensitivity analysis

The initial values of encryption key are (6.13455323257449, −6.76623087823196, 7.52223762673178, 6.2204540358 4687). We use each modified key to encrypt three plain-regions and calculate *NPCR* and *UACI* between the output cipher-regions and the cipher-regions ciphered by the original key. The modified key values and the corresponding test results of *NPCR* and *UACI* are reported in Table 4. From this table, we can see that the values of *NPCR* and *UACI* are very close to theoretical values, showing the strong encryption key sensitivity of our algorithm.

4.4.2 Decryption key sensitivity analysis

In decryption process, the modified keys listed in Table 4 are used to decrypt the cipher-regions encrypted by the original key. The decryption results are shown in Fig. 10, we can see that the cipher-regions cannot be decrypted correctly, showing strong decryption key sensitivity.

It should be noted that the decryption process depends on the decryption key and data hiding key. The receiver cannot decrypt the cipher-region without the correct data hiding key.

4.5 Comparison of proposed work and state-of-the-art algorithms

We compare our work with some state-of-the-art similar algorithms [18–20, 22, 23], which are reviewed in Sect. 2.1. The automatic detection, the protection of objects, edge area, and ROI information are the key points of comparison. Table 5 lists the comparison results. It can be seen that our algorithm outperform others in the protection of ROI coordinates information. And the object detection model is modified to protect all the areas of detected object, instead of using existing models without change.

Table 3 The results of *NPCR* and *UACI* test

Object Name	*NPCR*	*UACI*
Person	0.994085	0.334527
Horse	0.994117	0.334208
Dog	0.993551	0.332660

4.6 Limitation and discussion

Although the Jia system provides a large key space, its iteration is time consuming when there are many images that need to be encrypted. Recently, some 2D discrete chaotic systems with simple structures and continuous chaotic ranges have been developed [43, 44]. For example, the authors proposed a 2D modular chaotification system in [44], and proved that it significantly improves the chaos complexity and enlarges the chaotic ranges of existing 2D chaotic maps. And it's very convenient to use such 2D chaotic systems to encrypt a large number of images.

Table 4 Key sensitivity analysis of encryption process

Key component	New value	Person		Horse		Dog	
		NPCR	*UACI*	*NPCR*	*UACI*	*NPCR*	*UACI*
x_0	6.13455323257448	0.99394	0.33405	0.99414	0.33378	0.99360	0.33334
y_0	−6.76623087823195	0.99357	0.33365	0.99464	0.33466	0.99417	0.33428
z_0	7.52223762673177	0.99347	0.33445	0.99348	0.33375	0.99394	0.33519
u_0	6.22045403584687	0.99358	0.33332	0.99468	0.33307	0.99366	0.33393

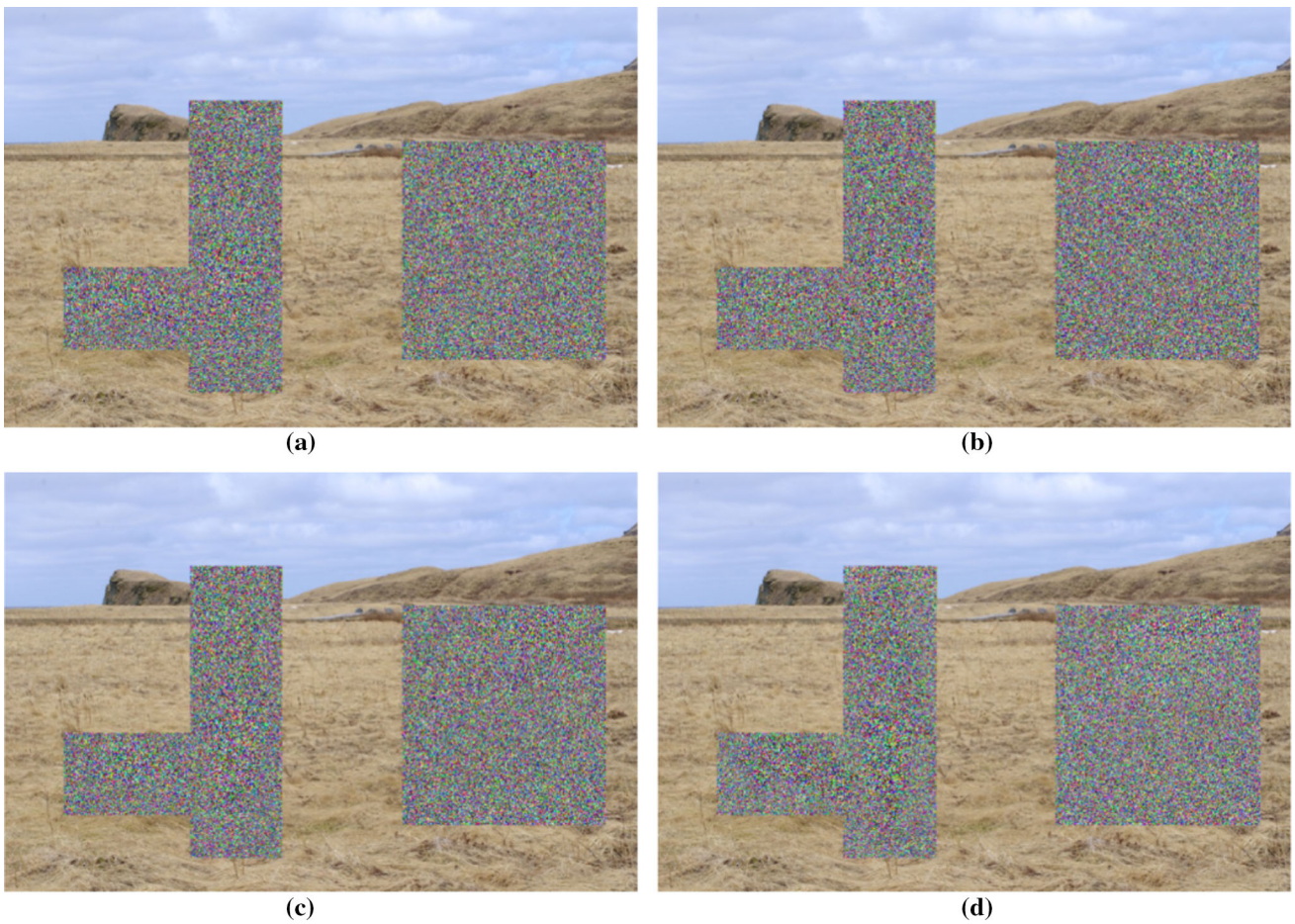

Fig. 10 The key sensitivity analysis in decryption process. **A–D** are the decrypted results using modified keys

Table 5 Comparison of proposed work and state-of-the-art algorithms

Algorithm	Automatic detection	Edge area protection	Target protection	ROI coordinates protection
Our work	✓	✓	✓	✓
Ref. [18]	✓	✓	✓	–
Ref. [19]	–	✓	✓	–
Ref. [20]	✓	–	✓	–
Ref. [22]	✓	–	✓	–
Ref. [23]	✓	–	✓	–

5 Conclusion and future work

In this paper, we protect the image ROI using chaos-based encryption and DCNN-based object detection. We first analyze the security problems in the object detection process, as the existing object detection algorithm fails to contain all parts of detected objects, we modify the detection process to make the output bounding boxes contain the whole objects. Then we propose an encryption algorithm for protecting multiple detected objects, ensuring that the encryption effect of each cipher-region can meet the security requirements. After that, we encrypt the ROI auxiliary information and embed them into the whole image using reversible data hiding. The encryption and embedding of ROI auxiliary information can be seen as a second layer of ROI protection and also save the trouble of its distribution. The experimental results and security analyses show that our proposed encryption scheme is secure and very suitable for ROI encryption.

In the future work [1], we would explore the use of access control encryption (ACE) in object detection. ACE [45, 46] decides not only what users are allowed to read but also what users are allowed to write. It can be constructed by attribute-based encryption, in a way that senders of correct knowledge (e.g., secret key) can transmit data to a restricted recipient with particular attributes. Object detection supports picking regions of interests/importance/sensitivity that should be limited in both reading and writing. Extending ACE usage to practical life would be promising in ROI scenarios.

Acknowledgements This work was supported by the National Natural Science Foundation of China (No. 61773068), the National Key R&D Program of China (No. 2021YFF0306405), and the Fundamental Research Funds for the Central Universities (No. N2024005-1).

[1] In response to an anonymous reviewer: Authors at CUHK and NEU thanks him/her for motivating their discussion on potentially-further works, i.e., protecting detectable-and-sensitive objects with reasonable access control and fast 2D chaotic encryption.

Declarations

Conflicts of interest The authors declare that they have no conflict of interest.

References

1. Zhang W, Yu H, Yl Zhao, Zl Zhu (2016) Image encryption based on three-dimensional bit matrix permutation. Sign Process 118:36–50
2. Fridrich J (1998) Symmetric ciphers based on two-dimensional chaotic maps. Int J Bifurc chaos 8(06):1259–1284
3. Murillo-Escobar MA, Cruz-Hernández C, Abundiz-Pérez F, López-Gutiérrez RM, Del Campo OA (2015) A rgb image encryption algorithm based on total plain image characteristics and chaos. Sign Process 109:119–131
4. Chen J, Zhang Y, Qi L, Fu C, Xu L (2018a) Exploiting chaos-based compressed sensing and cryptographic algorithm for image encryption and compression. Optics Laser Technol 99:238–248
5. Chen J, Zhu Z, Zhang L, Zhang Y, Yang B (2018b) Exploiting self-adaptive permutation-diffusion and DNA random encoding for secure and efficient image encryption. Sign Process 142:340–353
6. Alawida M, Teh JS, Samsudin A, Alshoura WH (2019) An image encryption scheme based on hybridizing digital chaos and finite state machine. Sign Process 164:249–266
7. Xingyuan W, Suo G (2020) Image encryption algorithm for synchronously updating boolean networks based on matrix semi-tensor product theory. Inf Sci 507:16–36
8. Song W, Zheng Y, Fu C, Shan P (2020) A novel batch image encryption algorithm using parallel computing. Inf Sci 518:211–224
9. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. Adv Neural Inf Process Syst 25:1097–1105
10. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M et al (2015) Imagenet large scale visual recognition challenge. Int J Comput Vision 115(3):211–252
11. Szegedy C, Toshev A, Erhan D (2013) Deep neural networks for object detection. In: Adv Neural Inf Process Syst, 2553–2561
12. Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 580–587
13. Girshick R (2015) Fast r-cnn. In: Proceedings of the IEEE international conference on computer vision, pp 1440–1448
14. Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: unified, real-time object detection. In: Proceedings of

the IEEE conference on computer vision and pattern recognition, pp 779–788
15. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, Berg AC (2016) Ssd: single shot multibox detector. In: European conference on computer vision, Springer, pp 21–37
16. He K, Gkioxari G, Dollár P, Girshick R (2017) Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision, pp 2961–2969
17. Wen W, Zhang Y, Fang Z, Jx Chen (2015) Infrared target-based selective encryption by chaotic maps. Optics Commun 341:131–139
18. Kanso A, Ghebleh M (2015) An efficient and robust image encryption scheme for medical applications. Commun Nonlinear Sci Numer Simul 24(1–3):98–116
19. Xiao D, Fu Q, Xiang T, Zhang Y (2016) Chaotic image encryption of regions of interest. Int J Bifurc Chaos 26(11):1650193
20. Sun J, Liao X, Chen X, Guo S (2017) Privacy-aware image encryption based on logistic map and data hiding. Int J Bifurc Chaos 27(05):1750073
21. Xue Hw DuJ, Sl Li, Wj Ma (2018) Region of interest encryption for color images based on a hyperchaotic system with three positive lyapunov exponets. Optics Laser Technol 106:506–516
22. Liu Y, Zhang J, Han D, Wu P, Sun Y, Moon YS (2020) A multidimensional chaotic image encryption algorithm based on the region of interest. Multimed Tools Appl 79:1–37
23. Asgari-Chenaghlu M, Feizi-Derakhshi MR, Nikzad-Khasmakhi N, Feizi-Derakhshi AR, Ramezani M, Jahanbakhsh-Nagadeh Z, Rahkar-Farshi T, Zafarani-Moattar I, (2021) Cy: chaotic yolo for user intended image encryption and sharing in social media. Inf Sci 542:212–227
24. Bochkovskiy A, Wang CY, Liao HYM (2020) Yolov4: optimal speed and accuracy of object detection. arXiv preprint arXiv:200410934
25. Zhu W, Liang S, Wei Y, Sun J (2014) Saliency optimization from robust background detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2814–2821
26. Redmon J, Farhadi A (2018) Yolov3: an incremental improvement. arXiv preprint arXiv:180402767
27. Ronneberger O, Fischer P, Brox T (2015) U-net: convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention, Springer, pp 234–241
28. Redmon J, Farhadi A (2017) Yolo9000: better, faster, stronger. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 7263–7271
29. Ali W, Abdelkarim S, Zidan M, Zahran M, El Sallab A (2018) Yolo3d: end-to-end real-time 3d oriented object bounding box detection from lidar point cloud. In: Proceedings of the European Conference on Computer Vision (ECCV) Workshops
30. Huang R, Pedoeem J, Chen C (2018) Yolo-lite: a real-time object detection algorithm optimized for non-gpu computers. In: 2018 IEEE International Conference on Big Data (Big Data), IEEE, pp 2503–2510
31. Ni Z, Shi YQ, Ansari N, Su W (2006) Reversible data hiding. IEEE Trans Circuits Syst Video Technol 16(3):354–362
32. Ma K, Zhang W, Zhao X, Yu N, Li F (2013) Reversible data hiding in encrypted images by reserving room before encryption. IEEE Trans Inf Forensics Secur 8(3):553–562
33. Cao X, Du L, Wei X, Meng D, Guo X (2015) High capacity reversible data hiding in encrypted images by patch-level sparse representation. IEEE Trans Cybernet 46(5):1132–1143
34. Puteaux P, Puech W (2018) An efficient msb prediction-based method for high-capacity reversible data hiding in encrypted images. IEEE Trans Inf Forensics Secur 13(7):1670–1681
35. Puyang Y, Yin Z, Qian Z (2018) Reversible data hiding in encrypted images with two-msb prediction. In: 2018 IEEE International Workshop on Information Forensics and Security (WIFS), IEEE, pp 1–7
36. Yi S, Zhou Y (2018) Separable and reversible data hiding in encrypted images using parametric binary tree labeling. IEEE Trans Multimed 21(1):51–64
37. Yin Z, Xiang Y, Zhang X (2019) Reversible data hiding in encrypted images based on multi-msb prediction and huffman coding. IEEE Trans Multimed 22(4):874–884
38. Wu Y, Xiang Y, Guo Y, Tang J, Yin Z (2019) An improved reversible data hiding in encrypted images using parametric binary tree labeling. IEEE Trans Multimed 22(8):1929–1938
39. Tian J (2003) Reversible data embedding using a difference expansion. IEEE Trans Circuits Syst Video Technol 13(8):890–896
40. Jia Q (2007) Hyperchaos generated from the lorenz chaotic system and its control. Phys Lett A 366(3):217–222
41. Robert Matthews (1989) On the derivation of a chaotic encryption algorithm. Cryptologia 8(1):29–41
42. Alvarez G, Li S (2006) Some basic cryptographic requirements for chaos-based cryptosystems. Int J Bifurc Chaos 16(08):2129–2151
43. Hua Z, Jin F, Xu B, Huang H (2018) 2d logistic-sine-coupling map for image encryption. Signal Process 149:148–161
44. Hua Z, Zhang Y, Zhou Y (2020) Two-dimensional modular chaotification system for improving chaos complexity. IEEE Trans Signal Process 68:1937–1949
45. Han J, Bei M, Chen L, Xiang Y, Cao J, Guo F, Meng W (2019) Attribute-based information flow control. Comput J 62(8):1214–1231
46. Wang X, Chow SS (2021) Cross-domain access control encryption: arbitrary-policy, constant-size, efficient. In: IEEE Symposium on Security and Privacy (S&P), pp 388–401