

We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800

Open access books available

122,000

International authors and editors

135M

Downloads

Our authors are among the

154

Countries delivered to

TOP 1%

most cited scientists

12.2%

Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.

For more information visit www.intechopen.com



Using Algebraic Fractals in Steganography

Oleg Sheluhin and Dzhennet Magomedova

Abstract

Steganography is a technology for hiding watermarks inside media files, which is relevant in the field of copyright protection, secret communication, etc. The effectiveness of modern methods of digital image processing allows determining the presence of embedded watermarks in a stegoimage using the original image and its statistical characteristics, as well as a priori information about the method and algorithm of embedding. In contrast to the known approaches, it is proposed to use algebraic fractals for steganographic embedding of watermarks in color images. It is proposed to use algebraic fractals in the form of medium cover image acting as a secret key, which allows the embedding to be more resistant to computer attacks, including JPEG compression. The main advantage of such use of fractals is an increase in the level of secrecy in which the attacker must know the parameters of the fractal image. Without knowledge of these parameters, it will not be enough to have the original stegoimage and a priori information about the embedding method to extract secret data. This chapter analyzes the methods and provides examples of generating algebraic fractals in the form of the Julia set using the escape time algorithm.

Keywords: steganography, watermarks, color images, copyright protection, Julia set, escape time algorithm

1. Introduction

The digital form of audio, image, and video has become a commercial standard in the past decade. Digitized multimedia files can be easily created, copied, processed, saved, and distributed using commercial and free software. Unfortunately, the digitization of multimedia files leads to the fact that these files are subject to digital piracy: illegal copying, use, and distribution of copyrighted digital data. In order to combat digital piracy, various copyright protection mechanisms have been developed for many years. However, most of these mechanisms were found to be erroneous and unsafe.

One of the tools to protect multimedia data from copyright infringement is digital watermarks. Into digital watermarks, an imperceptible “mark” signal is embedded in the original image. This label uniquely identifies the owner. After embedding the watermark in the original image (container) there should be no noticeable distortion. Embedded watermarks should not be removed by an incomplete person and must be resistant to intentional and unintentional attacks.

One of the ways to protect copyright, which currently exists, is steganography, the art of hiding information in electronic media. Methods of steganography vary in their approach to hiding information. As a rule, the hidden data in electronic means will change some of their properties, which can lead to degradation or unusual characteristics.

The paper deals with the introduction of watermark into a still color image of JPEG format using steganography methods. A feature of the proposed method is the use of a fractal image as a secret key, in which a two-dimensional fractal of algebraic type in the form of a Julia set is used. As a result, an evil intentional person will not be able to generate an identical fractal image without the exact value of a certain complex number, which is agreed in advance between the sender and the recipient, which makes the proposed method resistant to attacks.

Algorithms for creating a fractal image are considered, into which a digital watermark is then embedded.

To embed the fractal image in the container, the algorithm that performs a three-level expansion using the Haar transform was chosen and implemented in the MATLAB. In order to optimize the quality of extraction and the amount of embedded information, an algorithm for calculating thresholds has been proposed, with the help of which the necessary number of wavelet decomposition coefficients suitable for embedding has been selected.

To assess the possibility of detecting the fact of embedding the watermark in the stegoimage, the method of estimating the fractal dimension (FR) can be used before and after the mark has been embedded. With numerous examples of parameters of algebraic fractals and the hidden algorithms used, it is shown that the introduction of a watermark slightly changes the fractal dimension of the stegoimage, which corresponds to the high stability of the method to possible steganographic attacks.

To assess the quality of the results obtained, the quality was estimated based on the following metrics: mean square error (MSE) and signal-to-noise ratio (SNR), expressed in decibels. The results of the original container and the container in which the watermark with the key in the form of a fractal are embedded, as well as the original and extracted watermark, are given.

Analysis of the results of the quality assessment of the original container and the container with the embedded watermark and the key in the form of a fractal, as well as the original and extracted watermark showed that the proposed algorithm provides a high quality of hiding confidential information.

2. Algorithm development and software implementation of the generation of a fractal image of Julia set

The fractal image of a Julia sets will form using an algorithm of time escape (Escape time algorithm) [1, 2]. The algorithm is based on the use of complex maps when one complex number $z_n = x_n + iy_n$ is matched by another complex number $z_{n+1} = x_{n+1} + iy_{n+1}$ according to the iterative rule $z_{n+1} = f(z_n)$, where $f(z)$ is some nonlinear function of the argument z , and n is the iteration number.

The algorithm uses a quadratic complex polynomial: $f(z) = z^2 + c$ where $c = x + iy$ is the starting point on the complex plane on the basis of which the Julia set is constructed.

If point c belongs to the Mandelbrot set, the Julia set constructed on it is connected [3]. If c does not belong to the Mandelbrot set, the constructed sets are scattered into an infinite number of isolated points (Fatou dust). If the point c lies near the boundary of the Mandelbrot set, such sets form fractal figures figuratively

called the “seahorse valley” in the vicinity of $z_0 = 0$, which have the property of self-similarity with chaotic dynamics.

Thus, the algorithm for constructing the Julia set takes the form of [4]:

1. Select point c to set polynomial $f(z) = z^2 + c$;
2. Calculate the radius R for a given polynomial $f(z) = z^2 + c$;
3. Select the maxstep parameter to indicate the maximum iteration. The higher it is, the higher the accuracy and the slower the algorithm;
4. Generate an array of colors from less bright to brighter. We will denote the color dependence of the removal of points from a variety of Julia sets;
5. For each point, we calculate whether it is part of the filled Julia set or not, as well as the iteration number at which the threshold has exceeded. If $|z| > R$ then use the first color, then use the color palette on which the iteration number was exceeded a threshold.

The implementation of the algorithm begins with the construction of the rectangle $L \times L$, within which the Julia set will be constructed. By changing its parameters, we can “bring closer” the concentrations of interest to a set with high accuracy. The resulting rectangle is divided into parts with coordinates $z_{i,j}$ depending on the number of pixels of the image $M \times M$.

Each point $z_{i,j}$ enters an iterative loop $f(z) = z^2 + c$. The number of iterations depends on the maxstep parameter. If after a given number of iterations a point does not go to infinity, then such a point belongs to the Julia set.

Depending on which area of the Julia set will be generated ($L \times L$), a certain number of points will go to infinity, without reaching the maximum iteration.

To optimize the algorithm and significantly reduce the generation time of the Julia set, the following approach was adopted: for a complex number, the radius

$R = \frac{1}{2} + \frac{\sqrt{1+4|c|}}{2}$ is calculated, and at each iteration, the test $z_{i,j} > R$ occurs, for points satisfying this condition, iterations are no longer performed.

The points $z_{i,j}$ extending beyond this radius will go to infinity through a certain number of iterations; since their attractor is an infinitely distant point (points $z_{i,j} > R$ enter the pool of the attractor $A(\infty)$). The basin of an attractor of a point z of a function f is a subset of points from a neighborhood of z (denoted as $A(z)$), that any trajectory starting at one of these points converges to a point z .

Such an approach can significantly reduce the number of calculations, especially with the general form of the Julia set (2×2). However, with an increase in the approximation to the level (0.005×0.005) the number of calculations will tend to the initial ones, as more and more points will belong to the Julia Set. The color in the implementation of the algorithm was chosen so that the absolute red color (with a brightness level of 255) has points (belonging to the Julia set) that do not go to infinity with the maximum number of iterations maxstep. The brightness of the “red component” of the rest of the check depends on the number of iterations that the point passed, going to infinity (points not belonging to the Julia set, but rather close to its boundary). For the green channel, an inverse relationship was used: the smaller the iterations, the greater the brightness of the green component up to the level of 255, if the point was originally outside the radius. The blue channel changes depending on the distance of the point from the origin and is minimal at $z = 0$.

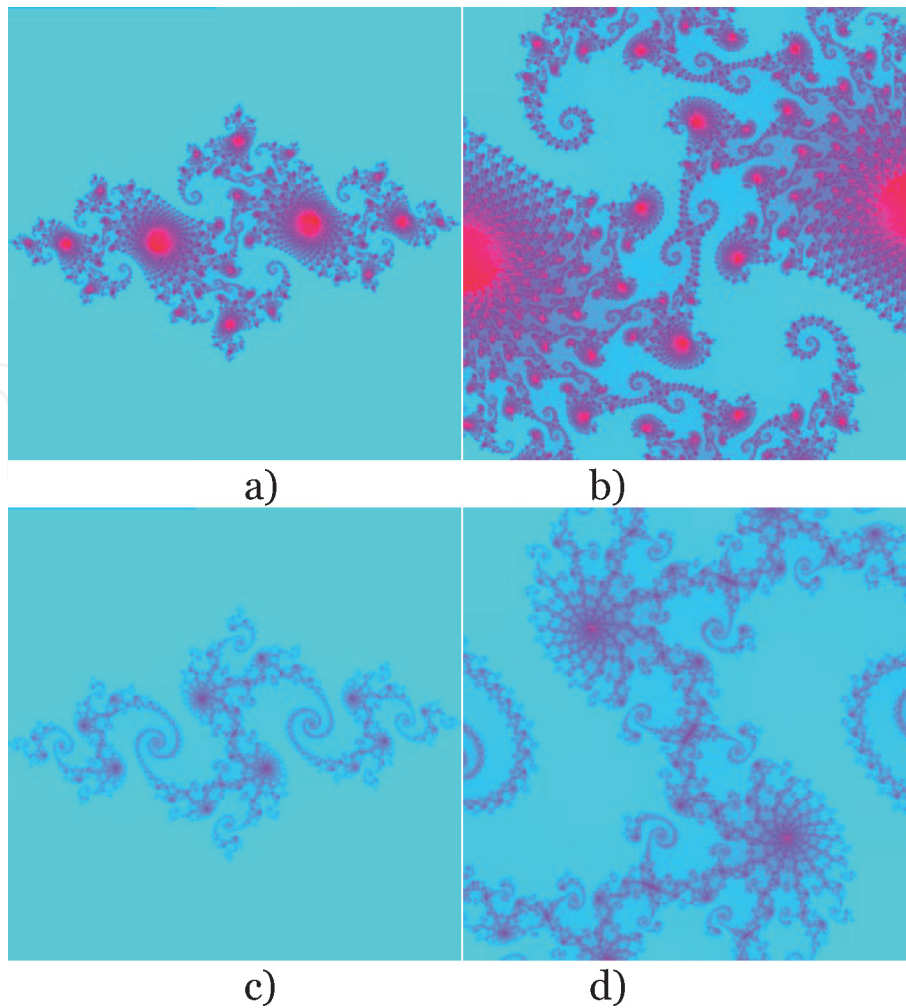


Figure 1. Generated fractals with various parameters: (a) $c = -0.74543 + 0.11301i$, $\text{maxstep} = 300$, $l = 1.5$, $m = 1024$; (b) $c = -0.74543 + 0.11301i$, $\text{maxstep} = 300$, $l = 0.5$, $m = 1024$; (c) $c = -0.77780781 + 0.13164510i$, $\text{maxstep} = 300$, $l = 1.5$, $m = 1024$; and (d) $c = -0.77780781 + 0.13164510i$, $\text{maxstep} = 300$, $l = 0.5$, $m = 1024$.

The implementation of the above method in the MATLAB math package is shown in Listing 1:

```
clear;
cx=0;
cy=0;
l=0.005;
maxstep=300;
m=600;
max=0;
C=-0.74543 + 0.11301i;
r=0.5+sqrt(1+4*abs(C))/2;

for i=1:m
for j=1:m
z(i,j)=-l+j*2*l/(m-1)+(-l+i*2*l/(m-1))*1i;
for k=1:maxstep
w((i-1)*m+j)=maxstep;
if (abs(z(i,j))>r)
w((i-1)*m+j)=k;
break;
end
z(i,j)=power(z(i,j),2)+C;
```

```

    if w((i-1)*m+j)>max
    max=w((i-1)*m+j);
    end
    end
    Red(i,j)=uint8(255*w((i-1)*m+j)/max);
    Green(i,j)=uint8(255-255*w((i-1)*m+j)/max);
    if (abs(z(i,j))/r)>1
    Blue(i,j)=255;
    else
    Blue(i,j)=uint8(255*(abs(z(i,j)/r)));
    end
    end
    end
    img=Red;
    img(:,,2)=Green;
    img(:,,3)=Blue;
    imwrite(img,'fractal.bmp');

```

User-defined initial conditions:
 maxstep - the maximum number of iterations, which determines the detailing of the Julia set;
 l - the dimension of the rectangle (determines the scale of the approximation);
 cx, cy- coordinates of the center of the rectangle (determine the coordinates of the approximation);
 m is the size of the container (the number of pixels of the image mxm).
 The results of the developed program module are presented in **Figure 1**.

3. Embedding a watermark in a stegoimage using a “fractal key”

The created fractal image based on Julia’s set can be used as an intermediate cover image (key) for embedding a digital watermark into it by one of the well-known methods [5–7]. So, for example, when using the least significant bit method when embedding, a fractal is converted into a one-dimensional binary array and every eighth bit of the given array (that is, every least significant bit of the next fractal byte) is replaced with a watermark bit.

After the watermark is fully integrated into the created fractal image, the step of embedding the filled fractal key into the original image begins [8–10].

As an example, consider embedding a watermark in a container using a 2D wavelet algorithm. To embed a watermark (in our case, a fractal image containing a watermark) in a cover image using 2D fiberboard, the algorithm [11–13] was used as a basis, which uses a three-level wavelet decomposition using the Haar transform. The decomposition of the container in the area of coefficients of detail of the first level LH1, HH1, HL1 is as follows:

A horizontal Haar transform is applied (across all rows) to the source container (Eqs. (1) and (2)).

$$C1[i,jj] = \frac{Cont[i,j] + Cont[i, j + 1]}{2} \quad (1)$$

$$C2[i,jj] = \frac{Cont[i,j] - Cont[i, j + 1]}{2} \quad (2)$$

where $i = 1, \dots, M; j = 1, \dots, N; jj = j, \dots, \frac{N}{2}$.

In Eqs. (1) and (2), the following notation is used: Cont is an array of pixels of the original image; C1 is the half-sum matrix of pixels in the original image; C2 is the half-difference matrix of the pixels of the original image; M is the number of rows of the original image; and N is the number of columns of the original image.

The resulting matrices C1 and C2 will have the same dimension $[i, jj]$, which means that the number of columns will be two times less relative to the Cont matrix of the original image. Then, using the values of the matrices C1 and C2, we calculate the coefficients of approximation (LL1) and details (HL1, LH1, HH1) for level 1 of the decomposition.

$$LL1[ii, jj] = C1[i, jj] + C1[i + 1, jj] \quad (3)$$

$$HL1[ii, jj] = C1[i, jj] - C1[i + 1, jj] \quad (4)$$

$$LH1[ii, jj] = C2[i, jj] + C2[i + 1, jj] \quad (5)$$

$$HH1[ii, jj] = C2[i, jj] - C2[i + 1, jj] \quad (6)$$

where $i = 1, \dots, M; j = 1, \dots, N; jj = j, \dots, \frac{N}{2}; ii = i, \dots, \frac{M}{2}$.

In Eqs. (3)–(6), the following notation is used: LL1 $[ii, jj]$ —an array of approximation coefficients; HL1 $[ii, jj]$ —an array of horizontal detail coefficients; LH1 $[ii, jj]$ —an array of vertical detail coefficients; and HH1 $[ii, jj]$ —an array of diagonal detail coefficients.

The arrays of coefficients of approximation and detailing of level 1 obtained using 2D DVP have the same dimension $[ii, jj]$, which means that each of these arrays is two times smaller than the array of the original image.

To get the arrays of coefficients for the second level of decomposition: LL2, HL2, LH2, HH2, you need to perform similar actions, while taking the array of approximation coefficients LL1 obtained at the first level of decomposition for the original image.

At each decomposition level, coefficients exceeding the threshold values are selected. The threshold T_i for the decomposition of level i depends on the maximum absolute coefficient C_i over all ranges of coefficients of level i , thus [14]:

$$T_i = 2^{[\log_2 C_i] - 1} \quad (7)$$

The watermark is embedded in all coefficients exceeding the threshold value T_i for the corresponding decomposition level obtained from Eq. (7). Used additive algorithm for embedding watermark:

$$v'_i(x, y) = v_i(x, y) + \alpha * v_i(x, y) * x_i(x, y) \quad (8)$$

where $v'_i(x, y)$ is a modified coefficient with coordinates (x, y) ; $v_i(x, y)$ is a coefficient chosen for implementation with coordinates (x, y) , for example, for level 1: LH1 (x, y) , HH1 (x, y) , HL1 (x, y) ; $x_i(x, y)$ —bit of watermark with coordinates (x, y) ; α —the scaling factor is adjusted for each level of decomposition. So for the range of coefficients of approximation of the third level $\alpha = 0.02$, since the coefficients of this range have large values. Scaling factors of 0.1, 0.2, and 0.4 are used for the third, second, and first levels of decomposition, respectively. The fractal image is first embedded in the wavelet coefficients of the third level, then, if the coefficients of the third level were not enough to fully embed the watermark, the coefficients of the second level are used, etc. To extract the watermark, the inverse of the implementation equation (Eq. (9)) is used, but the adaptive-level scale factor α is taken into account.

$$x'_i(x, y) = \frac{v'_i(x, y) - v_i(x, y)}{\alpha * v_i(x, y)} \quad (9)$$

A fractal image, in which a watermark is embedded, acts as a watermark.

4. Experimental results

Figure 2a shows the original unfilled image with a size of 6400×4096 pixels in JPEG format. The image containing the embedded key is shown in **Figure 2b**.

Figure 3 shows the results of extracting the secret fractal key from the filled image. **Figure 3a** shows the original fractal image, in **Figure 3b** is the image obtained after extraction.

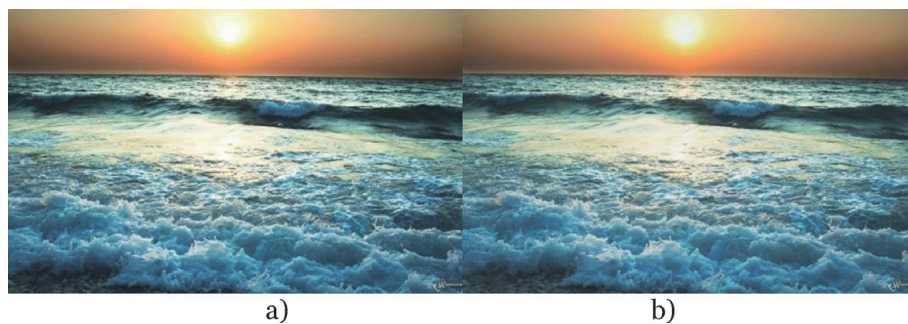


Figure 2.
(a) Original cover image; (b) stegoimage with embedded fractal key.

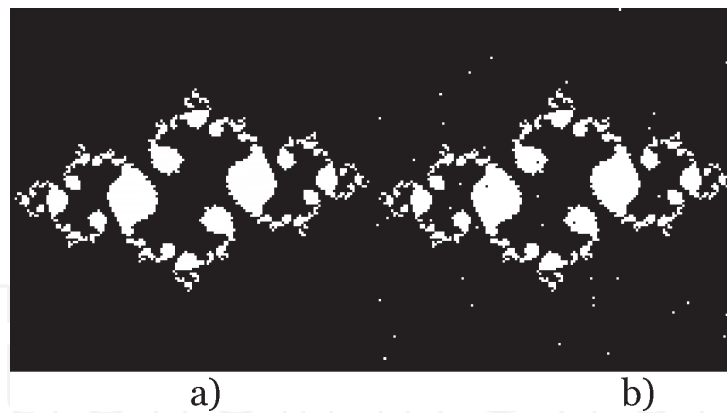


Figure 3.
(a) Generated fractal with watermark; (b) extracted from stegoimage fractal image with watermark.

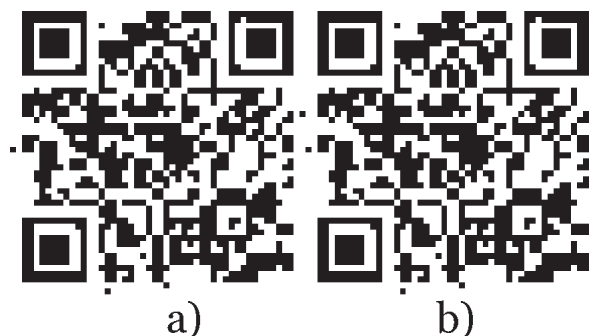


Figure 4.
(a) Original watermark; (b) watermark extracted from fractal key.

Figure 4 shows the results of extracting a digital watermark from the selected fractal. **Figure 4a** contains the original watermark and **Figure 4b** shows watermark extracted from the key.

As a result of the algorithm, a watermark was extracted with high quality. Despite the fact that you can observe small pixel distortions in the extracted secret key, the received watermark as a QR code is completely identical to the original one.

5. Evaluation of the quality of the algorithm

The quality assessment of the visual distortion of the fractal cover image was performed on the basis of the following metrics: normalized mean square error (NMSE) and peak signal-to-noise ratio (PSNR) (Eqs. (10) and (11)):

$$NMSE = \sum_{x,y} \frac{(C_{x,y} - S_{x,y})^2}{\sum_{x,y} (C_{x,y})^2} \quad (10)$$

$$PSNR = XY \cdot \max_{x,y} \frac{(C_{x,y})^2}{\sum_{x,y} (C_{x,y} - S_{x,y})^2} \quad (11)$$

In the relations presented, $C_{x,y}$ denotes a pixel with the coordinates (x, y) of the empty cover image, and $S_{x,y}$ denotes the corresponding pixel of the filled image.

The calculation of the metrics was carried out with different sizes of the implemented secret key and digital watermark. The results are presented in **Table 1**.

To assess the secrecy of the developed system, a situation was simulated in which the attacker took possession of a secret key with an embedded watermark, and he also knows some parameters of the generated fractal, namely, the size of the rectangle l , the maximum number of iterations k and the image size m . The only unknown parameter is the starting point c . As an example, four fractals were generated with different values of the parameter c (**Figure 5**) and the watermark was extracted using the obtained images and the secret key containing the watermark presented in **Figure 4b**.

The results of the extraction of watermark are shown in **Figure 6**.

As can be seen from the figures, all the images obtained as a result of the experiment are so significantly different from the original OT. As a result, the extraction of information from the QR code becomes impossible. From the experiment, it is clear that to obtain the information it is necessary to know the parameters of the fractal key with high accuracy. The developed method makes it possible to reduce the probability of the substitution or theft of secret information in similar steganographic systems to almost zero.

Size of watermark, pixels	Fractal key size, pixels	NMSE	PSNR
30 × 30	85 × 85	0.0021	31,1922
35 × 35	100 × 100	0.0031	29,4217
40 × 40	115 × 115	0.0056	26,7753
45 × 45	130 × 130	0.006	26,5334
50 × 50	150 × 150	0.0067	26,3538

Table 1.
Values of NMSE and PSNR with different sizes of embedded data.

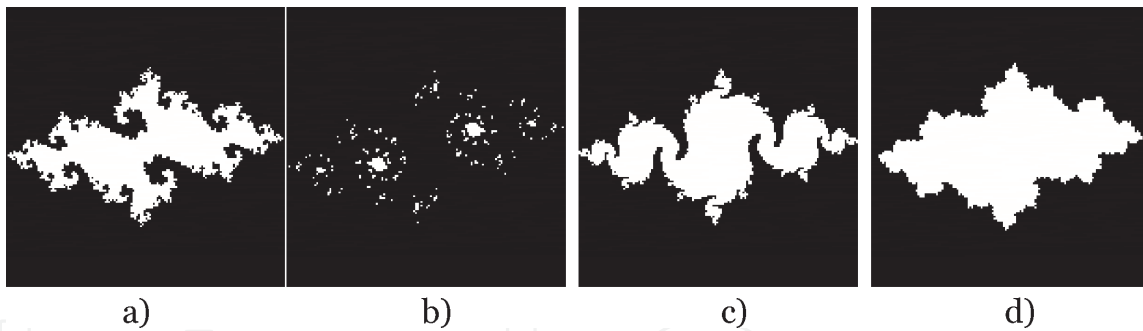


Figure 5.
 Fractal images with different starting point c . (a) $-0.73949 + 0.16498^*i$; (b) $-0.74549 + 0.37841^*i$; (c) $-0.80939 + 0.12388^*i$; (d) $-0.63949 + 0.19098^*i$.

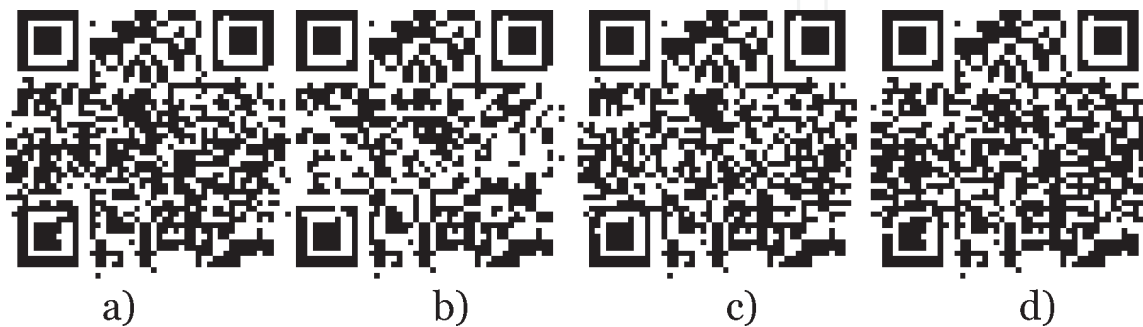


Figure 6.
 Watermarks obtained during extraction using fractals shown in Figure 5.

6. Analysis of the possibility of steganalysis by measuring the fractal dimension of the secret key before and after embedding

The importance of steganographic watermark embedding is their steganographic attacks resistance. In this case, the problem arises of detecting the very fact of the introduction of the watermark into the cover image. For this purpose, the method of estimating the fractal dimension of the image before and after embedding the watermark can be used. Consider the problem of measuring the fractal dimension of color or black and white images after embedding the watermark into them [15].

As is known, a fractal is defined as a collection of objects for which the Hausdorff dimension is strictly greater than the topological dimension. The concept of self-similarity is used to estimate the fractal dimension.

A bounded fractal set in a Euclidean n -space will be self-similar if it is a union of different N (disjoint) reduced copies that can be scaled using a special scaling factor r . In accordance with the entered scaling factor, the fractal dimension D of set A can be obtained using Eq. (12).

$$D = \frac{\log(N)}{\log(1/r)} \quad (12)$$

where N is the total number of boxes L needed to cover the fractal set; $1/r$ is the scaling factor of the box in relation to the image.

As a result, D is a dimension relative to the size of the box used to measure the fractal image. It can be said that the fractal dimension is a measure of how “complex” a self-similar figure is.

Let us consider the two most common methods for measuring dimension: differential box-counting and triangulation method.

The differential box-counting (DBC) method takes into account the difference between the maximum and minimum intensity values of the brightness of the image. Let an image be given with the size $M \times M$, which is divided into “boxes” with the size $L \times L \times L'$ (Figure 7). The height of the box L' divides the third coordinate of the image, which is the intensity value.

Space (x, y) of the image containing the values of the coordinates of the peak-mudflows is divided into cells of size $L \times L$, after which the maximum and minimum intensity values of (i, j) cell equal to l and k are found, respectively. The next step for each cell is the sum of the differences between the values found (Eq. (13)).

$$n_r(i,j) = (l - k + 1)/L' \tag{13}$$

where $r = L/M$ is the reduction factor.

After calculating the amount in all cells is the total amount of differences for the entire image (Eq. (14)).

$$N_r = \sum_{i,j} n_r(i,j) \tag{14}$$

The regression curve of log dependence $\log(\sum N_r)$ on $\log(1/R)$ is constructed on the basis of the calculations. Fractal dimension D is defined as the tangent of the slope angle of the curve.

When using the triangulation method (TM), the image is divided into identical cells ε of size $s \times s$. Four heights equal to the intensities of pixels in the corners of the cells (a–d) are considered. At the intersection of the diagonals of the cell, the point (e) is set, the value of which is equal to the arithmetic average of four heights. If we represent a cell in the form of a triangular prism, as shown in Figure 9, then it is necessary to calculate the area of the projected upper surface, shown in Figure 8.

First of all, the values of the sides of the four triangles, obtained by connecting the diagonals of the cell, are calculated (Eqs. (15) and (16)).

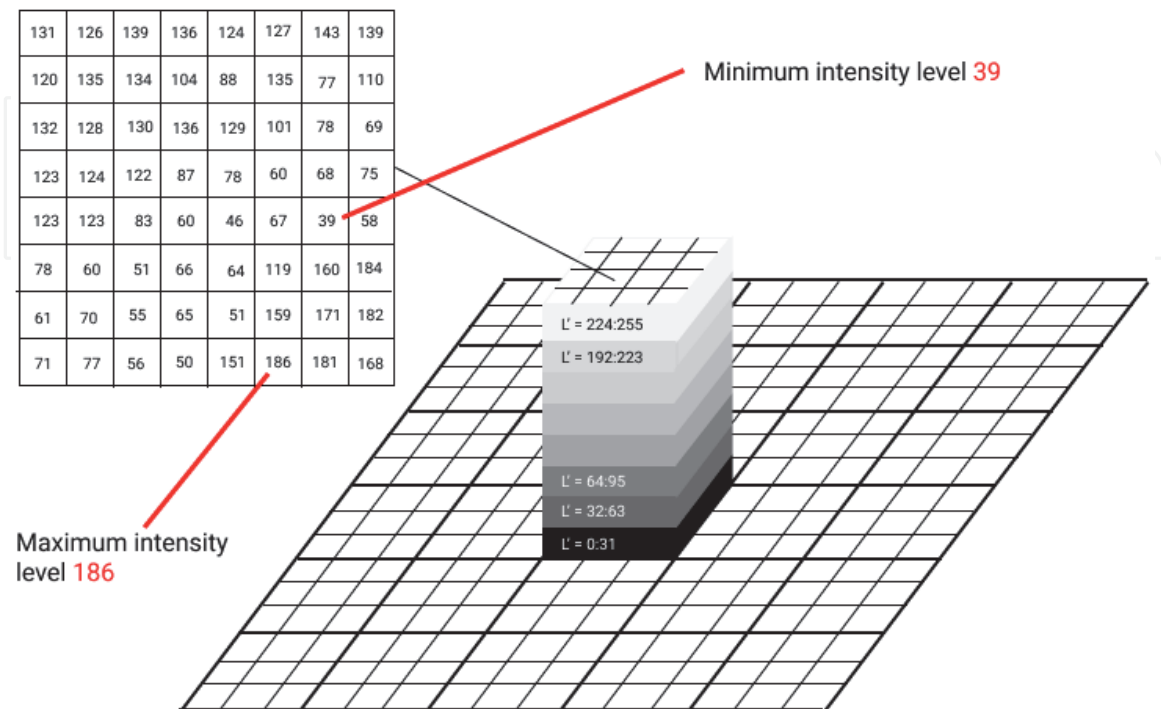


Figure 7. Three-dimensional representation of the image with its gray levels [15].

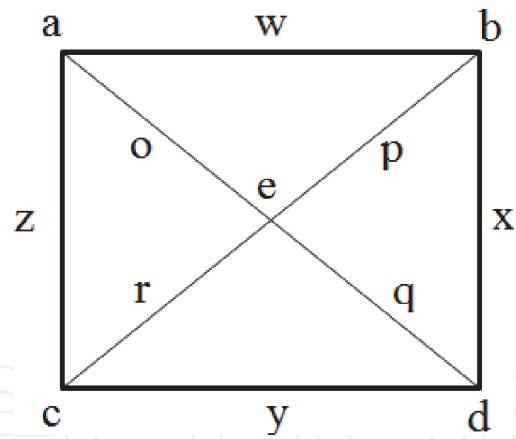


Figure 8.
 The projected top surface of a triangular prism.

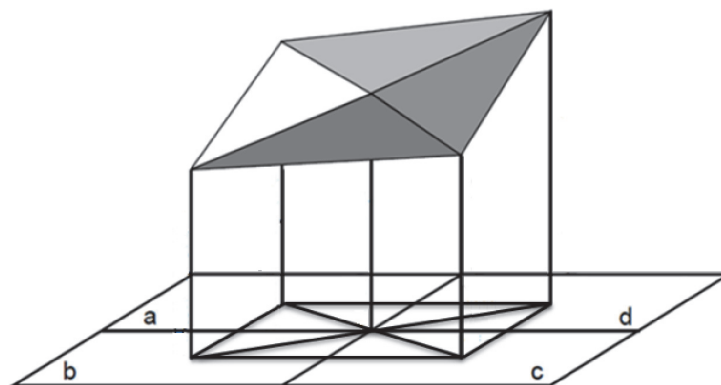


Figure 9.
 Representation of a cell of the image in the form of a triangular prism.

$$\begin{aligned}
 w &= \sqrt{(b-a)^2 + s^2}; x = \sqrt{(c-b)^2 + s^2}; \\
 y &= \sqrt{(d-c)^2 + s^2}; z = \sqrt{(a-d)^2 + s^2};
 \end{aligned}
 \tag{15}$$

$$\begin{aligned}
 o &= \sqrt{(a-e)^2 + \left(\frac{\sqrt{2}}{2}s\right)^2}; p = \sqrt{(b-e)^2 + \left(\frac{\sqrt{2}}{2}s\right)^2}; \\
 r &= \sqrt{(c-e)^2 + \left(\frac{\sqrt{2}}{2}s\right)^2}; q = \sqrt{(d-e)^2 + \left(\frac{\sqrt{2}}{2}s\right)^2}.
 \end{aligned}
 \tag{16}$$

Then, using the Heron formula, the semi-perimeters and areas of all triangles are calculated (Eqs. (17) and (18)).

$$\begin{aligned}
 A &= \sqrt{sa(sa-w)(sa-p)(sa-o)}; B = \sqrt{sb(sb-x)(sb-p)(sb-q)}; \\
 C &= \sqrt{sc(sc-y)(sc-q)(sc-r)}; D = \sqrt{sd(sd-z)(sd-o)(sd-r)}.
 \end{aligned}
 \tag{17}$$

$$\begin{aligned}
 sa &= \frac{1}{2}(w+p+o); sb = \frac{1}{2}(x+p+o); \\
 sc &= \frac{1}{2}(y+q+r); sd = \frac{1}{2}(z+o+r).
 \end{aligned}
 \tag{18}$$

The total surface area is equal to the sum of the areas of individual triangles (Eq. (19)).

$$S_{ABCD} = A + B + C + D \tag{19}$$

This procedure is repeated for all cell sizes. Then a regression line is constructed, which determines the dependence of the logarithm of the total area of all the triangles $\log(S)$ on the logarithm of the cell size $\log(\epsilon)$ (**Figure 10**).

To calculate the fractal dimension, it is necessary to find the tangent of the angle of inclination of the constructed curve B. It can be calculated using Eq. (20).

$$B = \frac{r * S_s}{S_\epsilon}$$

$$r = \frac{\text{cov}(\epsilon, S)}{S_\epsilon S_s}; \text{cov}(\epsilon, S) = \frac{\sum(\epsilon_i - \bar{\epsilon})(S_i - \bar{S})}{N};$$

$$S_\epsilon = \sqrt{\frac{\sum(\epsilon_i - \bar{\epsilon})^2}{N}}; S_s = \sqrt{\frac{\sum(S_i - \bar{S})^2}{N}}. \tag{20}$$

$\bar{\epsilon}, \bar{S}$ are the average values of the corresponding parameters.

The desired value of the fractal dimension D is calculated by Eq. (21).

$$D = 2 - B \tag{21}$$

In grayscale images, there is only one brightness level of each pixel of the image, while in a color image there are three color values (red, green, and blue) for each pixel of the image. As a result, to estimate the fractal dimension of color images, it is necessary to calculate the dimension for three different color levels. Using the estimation methods of calculating the fractal dimension for images in grayscale, one can calculate the dimension of a color image.

To calculate the dimension of a color image using the DBC method, the same method of calculating dimensions is used as in the grayscale images. In the color image, the dimension is calculated for the red R, green G, and blue B components. According to the DBC technique for each stage (red, green, and blue), you can apply the technique to images in shades of gray. After finding the results of each step, the results of different color levels are combined. As a result, you can get the dimension of the color image:

- R: $n_{rr}(i,j) = l-k + 1$ for red values;
- G: $n_{rg}(i,j) = l-k + 1$ for green values;
- B: $n_{rb}(i,j) = l-k + 1$ for blue values.

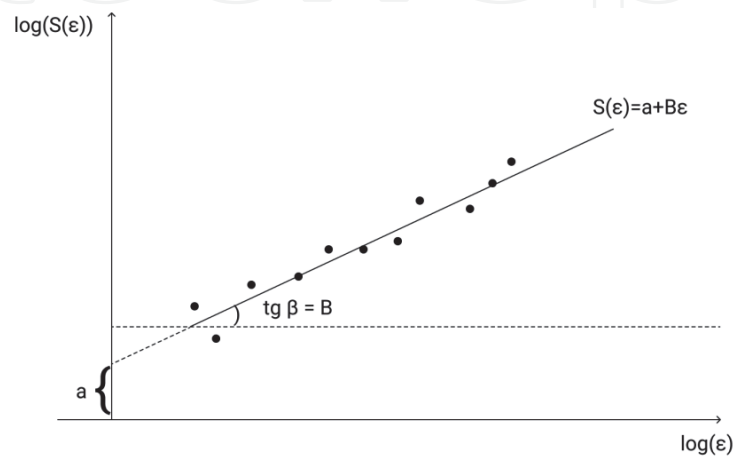


Figure 10.
Regression line.

According to the above, you can find the fractal dimension of the color image $N_{r\{r,g,b\}}$ (Eq. (22)).

$$N_{r\{r,g,b\}} = \frac{\sum_{i,j} n_{rr}(i,j) + \sum_{i,j} n_{rg}(i,j) + \sum_{i,j} n_{rb}(i,j)}{3} \quad (22)$$

When calculating the dimension of color images by the triangulation method, first of all, the particular values of the fractal dimension for each color component are determined, and then their average value is calculated.

7. Testing methods for estimating fractal dimension

To test the methods for evaluating the fractal dimension, two methods triangulation and DBC were considered. Testing was carried out with the help of fractal objects for which the dimension is known. **Table 2** presents the results of testing the above assessment methods using fractals with a known dimension value.

In **Table 2**, the following designations are made:

$\Delta_{TM}\% = \frac{\hat{D}_{TM}-D}{D} \cdot 100\%$ is the absolute error in estimating the dimension between the measured and the true value of the fractal dimension by the triangulation method.

$\Delta_{DBC}\% = \frac{\hat{D}_{DBC}-D}{D} \cdot 100\%$ is the absolute error in estimating the dimension between the measured and the true value of the fractal dimension by the DBC method.

$\Delta_{AV}\% = \frac{\bar{D}-D}{D} \cdot 100\%$ is the absolute error of the average estimate of the fractal dimension obtained by two methods.

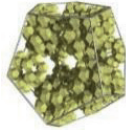
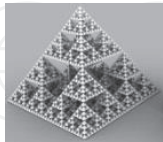


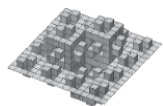
Name of fractal	Illustration	D	Testing results					
			TM	$\Delta_{TM},\%$	DBC	$\Delta_{DBC},\%$	\bar{D}	$\Delta_{AV},\%$
Dodecahedron fractal		2.3296	2.2276	4.378	2.4203	3.893	2.3239	0.243
Fractal pyramid		2.3219	2.2641	2.489	2.5524	9.03	2.4083	3.719
Jerusalem cube		2.529	2.3218	8.193	2.5830	2.135	2.4524	3.029
3D quadratic Koch surface		2.3347	2.2260	4.656	2.4068	3.088	2.3164	0.784
3D quadratic Koch surface (type 2)		2.5	2.3879	4.484	2.5705	2.82	2.4792	0.832

Table 2.
 Results of testing methods for measuring the fractal dimension.

The numerical data of the absolute errors of measurement of the dimension with the help of triangulation and DBC methods show that they give comparable results. The error of calculations for both methods does not exceed 5%. For practical use, it is advisable to use both methods, and the dimension values to find by averaging the results of both methods. In this case, the total error of calculations does not exceed 3%.

8. The measurement results of the fractal dimension of fractal keys

To illustrate the possibility of detecting an embedded watermark in images, we consider the experimental results of the evaluation of the fractal dimension after embedding the watermark in a fractal image based on Julia set.

Table 3 shows the results of measuring the fractal dimension of color fractal images in the form of a Julia set when embedding watermark by various known steganographic methods.

The results show that the steganographic introduction of data does not actually affect the value of the fractal dimension, which makes it impossible to illegally extract the watermark.

Starting point c	Number of iterations	Size of rectangle	Embedding method	Image size	D of original fractal	D of fractal with watermark
$-0.7778 + 0.1316i$	300	1.5	LSB	1024	2.318	2.318
			Block based		2.318	2.317
			Koch R		2.318	2.316
			Koch G		2.318	2.316
			Koch B		2.318	2.318
$-0.74543 + 0.11301i$	3000	1.5	LSB	1024	2.438	2.438
			Block based		2.438	2.437
			Koch R		2.438	2.436
			Koch G		2.438	2.436
			Koch B		2.438	2.438
$-0.74543 + 0.11301i$	3000	0.5	LSB	1024	2.109	2.109
			Block based		2.109	2.115
			Koch R		2.109	2.136
			Koch G		2.109	2.097
			Koch B		2.109	2.112
		0.5	LSB		2.192	2.192
			Block based		2.192	2.194
			Koch R		2.192	2.208
			Koch G		2.192	2.178
			Koch B		2.192	2.195
		0.0005	LSB		2.172	2.172
			Block based		2.172	2.179
			Koch R		2.172	2.186

Starting point c	Number of iterations	Size of rectangle	Embedding method	Image size	D of original fractal	D of fractal with watermark
			Koch G		2.172	2.165
			Koch B		2.172	2.359
	300	1.5	LSB	512	2.449	2.449
			Block based		2.449	2.442
			Koch R		2.449	2.427
			Koch G		2.449	2.425
			Koch B		2.449	2.450
		0.5	LSB		2.541	2.541
			Block based		2.541	2.546
			Koch R		2.541	2.523
			Koch G		2.541	2.523
			Koch B		2.541	2.537
		0.0005	LSB		2.396	2.396
			Block based		2.396	2.383
			Koch R		2.396	2.474
			Koch G		2.396	2.423
			Koch B		2.396	2.487
	3000		LSB		2.183	2.183
			Block based		2.183	2.183
			Koch R		2.183	2.221
			Koch G		2.183	2.323
			Koch B		2.183	2.122

Table 3.
 Changing the value of the fractal dimension of images during steganographic embedding.

9. Conclusions

The secrecy of steganographic systems is based on the assumption that the attacker is not aware of the fact of the information being introduced. In the event that this fact becomes publicly available, it will not be difficult for an unauthorized user to extract secret data or delete a given watermark. To solve this problem, it was proposed to use an additional container in the form of an algebraic fractal in the form of a Julia set.

Fractal generation is carried out using predefined secret parameters using the escape time algorithm (Escape time algorithm).

Embedding a digital watermark in a container was carried out in two stages. At the first stage, the watermark is added to the generated fractal. The resulting image is embedded in the original container in JPEG format. As a result, the original image has almost no visual distortion. The measurement of the values of the NMSE and PSNR metrics confirmed the high level of embedding quality and extraction of the watermark in the form of a QR code.

To confirm the high level of secrecy, an experiment was conducted, during which an attempt was made to replace the original secret key. The experiment

confirmed that it is impossible to extract the watermark without knowing the true parameters of the fractals.

To assess the possibility of detecting the fact of embedding a watermark on the basis of fractals, the fractal dimension of intermediate containers was measured on the basis of the Julia set. Measurement of the dimension value was carried out before and after steganographic embedding. The results showed that the fractal dimension varies slightly, within the limits of the method error, and cannot be a sign that characterizes the presence of an integrated watermark.

The proposed technique allows for steganographic embedding with a high level of visual quality, as well as resistance to various steganographic attacks. As a result, it is possible to increase the level of secrecy of the watermark embedding algorithms and significantly reduce the likelihood of data theft.


IntechOpen

Author details

Oleg Sheluhin and Dzhennet Magomedova*
Information Security Department, Moscow Technical University of
Communication and Informatics, Moscow, Russia

*Address all correspondence to: jimagomedova@gmail.com

IntechOpen

© 2020 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/3.0>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. 

References

- [1] Thamizhchelvy K, Geetha G. Design of digital signature algorithm by fractals and chaos theory. *International Journal of Computer Applications*. 2012;5:50-57. DOI: 10.5120/4608-6588
- [2] Aslan N, Saltan M, Demir B. A different construction of the classical fractals via the escape time algorithm. *Journal of Abstract and Computational Mathematics*. 2018;4:1-15
- [3] Rani M. Cubic superior Julia sets. In: *Proceedings of the European Computing Conference*; 2011. pp. 80-84. ISBN: 978-960-474-297-4
- [4] Xing Y, Tan J, Hong P. Quaternion Julia fractals. In: *IEEE Computer Society, The 9th International Conference for Young Computer Scientists*; 2008. pp. 797-802
- [5] Nori AS, Al-Qassab AM. Steganographic technique using fractal image. *International Journal of Information Technology and Business Management*. 2014;52-59. ISSN: 2304-0777
- [6] Sheluhin OI, Kanaev SD. Algorithms and software implementation. In: Sheluhin OI, editor. *Steganography*. Moscow: Goryachaya liniya – Telekom; 2017. p. 616. ISBN: 978-5-9912-0579-5
- [7] Sheluhin OI, Smychek MA. Combined graphic scale image watermarking method for secret communication. *Radiotekhnicheskie i Telekommu-Nikatsionnyie Sistem*. 2017;1:68-75. ISSN: 2221-2574
- [8] Abbas TA, Hamza HK. Steganography using fractal images technique. *IOSR Journal of Engineering (IOSRJEN)*. 2014;4:52-61. ISSN(e): 2250-3021
- [9] Thamizhchelvy K, Geetha G. Data hiding technique with fractal image generation method using chaos theory and watermarking. *Indian Journal of Science and Technology*. 2014;7: 1271-1278. ISSN(Online): 0974-5645
- [10] Hardikkumar Desai V, Desai Apurva A. Image Steganography Using Mandelbrot Fractal. *International Journal of Computer Science Engineering and Information Technology Research (IJCSEITR)*. 2014; 4:71-80. ISSN(E): 2249-7943
- [11] Wu Y, Noonan J. Image steganography scheme using chaos and fractals with the wavelet transform. *International Journal of Innovation, Management and Technology*. 2012;3: 285-289
- [12] Sheluhin OI, Kanaev SD. Hiding watermarks in color images using algebraic fractals by 2D wavelet transform methods. *T-Comm*. 2018;6: 46-50
- [13] Xia X, Boncelet C, Arce G. Wavelet transform based watermark for digital images. *Optics Express*. 1998;12:497-511. DOI: 10.1364/oe.3.000497
- [14] Sheluhin OI, Magomedova DI. Analysis of methods for calculating the fractal dimension of color and gray-scale images. *H&ES Research*. 2017;6:6-16. ISSN: 2412-1363 (e)
- [15] Padhy LN. Fractal dimension of gray scale & colour image. *International Journal of Advanced Research in Computer Science and Software Engineering*. 2015;7:1285-1297. ISSN: 2277 128X