УДК 004.056:004.42

СТЕГАНОГРАФИЧЕСКИЙ МЕТОД НА ОСНОВЕ ВСТРАИВАНИЯ СКРЫТЫХ СООБЩЕНИЙ В КРИВЫЕ БЕЗЬЕ ИЗОБРАЖЕНИЙ ФОРМАТА SVG

Е. А. БЛИНОВА[1], ***П. П. УРБАНОВИЧ***[1), 2)]

[1]*Белорусский государственный технологический университет, ул. Свердлова, 13а, 220006, г. Минск, Беларусь*
[2]*Люблинский католический университет им. Иоанна Павла II, ал. Рацлавицке, 14, 20-950, г. Люблин, Польша*

Приведено описание стеганографического метода встраивания цифрового водяного знака в файлы векторных изображений формата SVG. Векторные изображения формата SVG могут включать в себя элементы на основе кривых Безье. Предлагаемый стеганографический метод базируется на разбиении кубических кривых Безье. Встраивание скрытой информации предусматривает разбиение кубических кривых Безье в соответствии со значениями последовательности, формирующей цифровую метку. Рассмотрены алгоритмы прямого и обратного

Образец цитирования:
Блинова ЕА, Урбанович ПП. Стеганографический метод на основе встраивания скрытых сообщений в кривые Безье изображений формата SVG. *Журнал Белорусского государственного университета. Математика. Информатика.* 2021; 3:68–83 (на англ.).
https://doi.org/10.33581/2520-6508-2021-3-68-83

For citation:
Blinova EA, Urbanovich PP. Steganographic method based on hidden messages embedding into Bezier curves of SVG images. *Journal of the Belarusian State University. Mathematics and Informatics.* 2021;3:68–83.
https://doi.org/10.33581/2520-6508-2021-3-68-83

Авторы:
Евгения Александровна Блинова – старший преподаватель кафедры информационных систем и технологий факультета информационных технологий.
Павел Павлович Урбанович – доктор технических наук, профессор; профессор кафедры информационных систем и технологий факультета информационных технологий[1)], профессор[2)].

Authors:
Evgeniya A. Blinova, senior lecturer at the department of information systems and technology, faculty of information technology.
evgenia.blinova@belstu.by
https://orcid.org/0000-0001-7245-8721
Pavel P. Urbanovich, doctor of science (engineering), full professor; professor at the department of information systems and technology, faculty of information technology[a] and professor[b].
p.urbanovich@belstu.by
https://orcid.org/0000-0003-2825-4777

стеганографического преобразования для доказательства подлинности и целостности цифрового векторного изображения. Создана библиотека StegoSVG для выполнения прямого и обратного стеганографического преобразования. Кратко охарактеризовано разработанное настольное приложение, реализующее метод.

Ключевые слова: стеганография; авторское право; цифровой водяной знак; векторная графика; формат SVG.

STEGANOGRAPHIC METHOD BASED ON HIDDEN MESSAGES EMBEDDING INTO BEZIER CURVES OF SVG IMAGES

E. A. BLINOVA[a]***, P. P. URBANOVICH***[a, b]

[a]*Belarusian State Technological University, 13a Sviardlova Street, Minsk 220006, Belarus*
[b]*John Paul II Catholic University of Lublin, 14 Racławickie Alley, Lublin 20-950, Poland*
Corresponding author: E. A. Blinova (evgenia.blinova@belstu.by)

The description of the steganographic method for embedding the digital watermark into image vector files of the SVG format is given. Vector images in SVG format can include elements based on Bezier curves. The proposed steganographic method is based on the splitting of cubic Bezier curves. Embedding hidden information involves splitting cubic Bezier curves according to the digital watermark given as numerical sequence. Algorithms of direct and reverse steganographic transformation are considered for proving the authenticity and integrity of a digital vector image. The StegoSVG library has been developed to implement forward and reverse steganographic transformations. The developed desktop application that implements the method is briefly described.

Keywords: steganography; copyright; digital watermark; vector graphics; SVG format.

Introduction

Technologies make it possible to easily create, store and transmit various types of data, such as images, texts and sounds, but these advantages allow other actions to be done easily as well: illegal copying, distribution, use, intentional distortion or destruction of information. Nowadays, the protection of digital content is gaining popularity. In this regard, the problem of developing and using methods and tools for protection of intellectual property rights, one of the areas of which is digital watermark technologies based on steganographic methods, is becoming more and more acute [1].

Due to the specific features and properties of digital document formats that are protected by a copyright, specific protection methods are developed for various formats to achieve specific goals. This is how two main directions in the subject area were formed: text steganography [1–8] and image steganography [1; 9; 10]. However, at present, for some types of containers, methods can be considered that are a combination of different approaches. For example, an electronic text document can be considered as a text or as a graphic image or as a set of fields containing meta-information and as a container with a specific structure [5–8], which allows to combine the classical approaches of both text and image steganography.

Regardless of the type of a container, using steganographic methods solves two main classes of problems: data hiding and copyright protection. The first implies the inconspicuous transmission of information through open channels, as well as undeclared and hidden storage of information. The second is implemented using digital watermarks and digital fingerprints. Digital fingerprints, sometimes referred to as digital tags, imply different steganographic message tags that are unique to each copy of the media. Digital watermarking usually means having the same watermark for every copy of a container. Digital prints and watermarks can be used to protect copyright on each copy of content, and to confirm the accuracy and integrity of the information transmitted. The main requirements for digital marks or watermarks are reliability and resistance to distortion or conversions [1; 3; 9; 10].

Research usually focuses on raster image formats. A feature of such files is their ubiquitous distribution, a relatively large volume, and, as a result, ample opportunities for watermark embedding. The most popular are image formats like BMP and JPG, for which many methods for embedding labels have been developed, including steganographic ones. Typically, images use the classical least significant bit (LSB) method or its modification [1; 3; 9; 10]. In connection with the growing popularity of vector images, it is of interest to develop and study steganographic methods for images of this type.

Theoretical justification of the steganographic method

This paper proposes a new steganographic method for vector image files in scalable vector graphics (SVG) format and describes an algorithm for its implementation. The method is based on the functional features of the main graphic primitives, based on which two-dimensional images are formed. Modification of some of the parameters of these primitives allows to embed secret information into an SVG file used as a container.

SVG vector image format and usage of its features in steganography. SVG files are vector graphics files used to describe two-dimensional vector and mixed vector and raster graphics in XML format. The SVG image format is the main vector graphics format on the Internet. Such files are used as pictures on buttons and other elements of a web application, for representation of graphs, maps, diagrams in speeches, reports, presentations. This is justified because the SVG file is scalable and looks the same at all screen resolutions. Features of this format are small file size, scalability, integration with HTML documents, the ability to embed bitmap graphics, the ability to edit in text editors and support by most modern browsers such as Google Chrome, Internet Explorer, Mozilla Firefox and Safari. In addition, the popular office suite Microsoft Office 2016 supports direct import of such files.

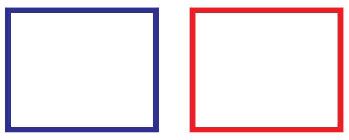
An SVG graphic file is a collection of XML tags that describe graphic elements. The first line is a standard XML header indicating the XML version and character encoding. Then the DOCTYPE header comes, that defines the SVG document type. Further is the root tag of the document <SVG> indicating the namespace, and it contains graphic and text elements. The document ends by closing the root tag </SVG>. An SVG file includes three types of objects: shapes, images, and text. All elements are described with XML subset tags. Text elements can be described as text or converted into curves.

An SVG file is a text, has a structure corresponding to markup files, geometric elements are described with appropriate tags. Tags have attributes whose values are enclosed in quotes. Shapes are described using RGB colour space, although CMYK is also supported.

Classical methods of text steganography, such as the insertion of trailing spaces and tabs method, as well as methods typical for markup files: the method of replacing the tags case, methods of substituting and rearranging of attributes [4] can be applied to files in the SVG format. Using the RGB colour model allows you to hide information in changing colour parameters by analogy with the methods of LSB.

The authors of [11; 12] substantiate the possibility of using additional vertices in the description of geometric shapes in an SVG file to embed a unique digital watermark to confirm copyright for images or for transfer a hidden message, and also propose a steganographic method that allows to embed a hidden message in placing additional vertices. Geometric shapes are specified by the coordinates of the vertices, and additional vertices can be placed on the line connecting these coordinates. Moreover, when viewing the image, these additional points are not displayed. Additional points in geometric shapes are set in a certain relation from the starting point of some file element, and a message is hidden in the value of this relation. This method allows to control the integrity of the image or to embed a hidden message to solve the problem of protecting the intellectual copyright to the image.

Figure 1 shows examples of rendering of the SVG file elements in the browser, as well as their description.



```
<polygon style = "stroke:blue; stroke-width:2; fill:none;"
points = "10,10  10,50  50,50  50,10"/>
<polygon style = "stroke:red; stroke-width:2; fill:none;"
points = "60,10  60,30  60,50  100,50  100,10"/>
```

Fig. 1. A rendering of the SVG file in the browser and its description

An additional vertex (60, 30) has been added in the second square, and since this element is in line with vertices (60, 10) and (60, 50), the vertex is not displayed.

Features of displaying Bezier curves in SVG files. SVG files support Bezier elements. To understand the essence of the proposed method, it is advisable to give a brief description of the parameters of the Bezier curves that form the basis of the method.

It is known that Bezier curves are a special case of polynomial plane curves with one parameter. They are often used in computer graphics to produce smooth curves, including fonts, vector images and CSS animations [13].

Bezier curves are generally defined by the expression

$$B(t) = \sum_{t}^{n} P_i b_{i,n}(t),\ 0 \le t \le 1, \quad (1)$$

where P_i is a function of the components of the vectors of the anchor points; $b_{i,n}(t)$ are the basic functions of the Bezier curve given by the expression

$$b_{i,n}(t) = \frac{n}{i} t^i (1-t)^{n-i}, \tag{2}$$

where $\frac{n}{i} = \frac{n!}{i!(n-i)!}$ is a number of combinations from n to i, where n is the degree of the polynomial and i is the degree of the anchor point.

In the context of the method under consideration, two types of Bezier curves are of interest: a quadratic and cubic curve. From relations (1) and (2) we can obtain parametric equation of a quadratic Bezier curve in the following form:

$$Q = (1-t)^2 P_1 + 2(1-t)tP_2 + t^2 P_3,\ t \in [0, 1].$$

It can be seen that three points are required to uniquely determine the quadratic curve: P_1, P_2 and P_3. The starting point P_1 and the ending point P_3 are called the anchor point of the curve, and the point P_2 is the control point. The curve starts at point P_1, ends at point P_3, and point P_2 defines the direction and magnitude of the curve bend.

The parametric equation of cubic Bezier curve is as follows:

$$C = (1-t)^3 P_1 + 3(1-t)^2 tP_2 + 3(1-t)t^2 P_3 + t^3 P_4,\ t \in [0, 1]. \tag{3}$$

To uniquely define this curve, four points are required: P_1, P_2, P_3 and P_4. The starting point P_1 and the ending point P_4 are the anchor points, and the points P_2 and P_3 are the control points. The curve starts at point P_1, ends at point P_4, and points P_2 and P_3 define the direction and magnitude of the curve bend.

To describe curves in SVG format, is used a special tag *path*. The SVG file parser parses the contents of the *path* tag and displays the corresponding shape. The *path* tag allows to sequentially set the coordinates of the vertices of a line, polygon and other geometric shapes. The *path* tag has one attribute *d*, which can contain a series of commands and parameters used by those commands. Each command is identified with a special letter. All commands come in two variants: a command with a uppercase letter indicates absolute coordinates on the page and a command with a lowercase letter indicates relative coordinates. Coordinates in the *d* attribute are always specified without units and in a custom coordinate system. Typically, custom coordinates are specified in pixels. For example, *M* is a command that tells the SVG parser to move to the specified point and to start the next command from there, and the *L* command draws a straight line from the current point to the specified one.

SVG files use quadratic and cubic Bezier curves. Commands *Q* and *T* are used to display quadratic curves, commands *C* and *S* are used to display curves of the third order. In this case, the commands *T* and *S* allow to set an additional segment of the curve without specifying a control point and are parts of the commands *Q* and *C*. These commands can be found in one element and used multiple times. If the parser cannot execute the statement due to an invalid command, the element is displayed as long as possible, and the rest is not displayed.

To reduce the number of elements of the SVG file, and, accordingly, the size of the image file, the combination of several curves in one geometric element can be used. The Bezier curve in this case consists of several segments. Adding segments can be achieved in two ways: either by mirroring the control point relatively to the ending anchor point, or by adding additional segments after the existing curve segment.

Figure 2 shows the cubic Bezier curve. The curve consists of three segments, the first of which is marked with additional red lines, the second segment is marked with orange lines, and the third is marked with green lines. Lines are drawn to the control points for clarity, to demonstrate how the location of the control points affects the appearance of the curve.

Figure 3 shows the part of the SVG file that is responsible for displaying such a curve.

The listing (see fig. 3) shows four blocks of *path* commands, at that the execution of the first one displays the entire quadratic curve, and the rest are needed only to demonstrate the location of the control points.

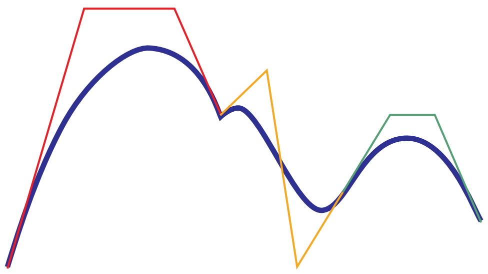


Fig. 2. A cubic Bezier curve consisting of three segments

```
<path d = ″M 10,200 C 60,30 120,30 150,100 180,70 200,200 230,150
S 290,100 320,170″ style = ″stroke:blue; stroke-width:4; fill:none;″/>

<path d = ″M 10,200 L 60,30″ style = ″stroke:red; stroke-width:1; fill:none;″/>
<path d = ″M 60,30 L 120,30″ style = ″stroke:red; stroke-width:1; fill:none;″/>
<path d = ″M 120,30 L 150,100″ style = ″stroke:red; stroke-width:1; fill:none;″/>

<path d = ″M 150,100 L 180,70″ style = ″stroke:orange; stroke-width:1; fill:none;″/>
<path d = ″M 180,70 L 200,200″ style = ″stroke:orange; stroke-width:1; fill:none;″/>
<path d = ″M 200,200 L 230,150″ style = ″stroke:orange; stroke-width:1; fill:none;″/>

<path d = ″M 230,150 L 260,100″ style = ″stroke:green; stroke-width:1; fill:none;″/>
<path d = ″M 260,100 L 290,100″ style = ″stroke:green; stroke-width:1; fill:none;″/>
<path d = ″M 290,100 L 320,170″ style = ″stroke:green; stroke-width:1; fill:none;″/>
```

Fig. 3. Description of a cubic Bezier curve, consisting of three segments, in SVG format

Let's consider the first path command. The curve starts at point (10, 200), the *M* command indicates to start displaying from this point. The *C* command then tells the SVG parser to display a third-order Bezier curve. Thus, point (60, 30) is perceived by parser as the first control point, point (120, 30) is perceived as the second control point, and point (150, 100) is perceived as the end point of the curve. If we look at the second, third and fourth commands of the listing, we will see that straight lines of red colour were drawn from the start and end points to the control points of the first segment using the *L* command. Since the command is not finished at point (150, 100), this point is perceived by the SVG parser as the starting point of the next segment. Thus, point (180, 70) is perceived as the first control point of the second curve segment, point (200, 200) is perceived as the second control point of the second curve segment, and point (230, 150) is perceived as the end point of the second curve segment. The next three lines of the listing draw additional orange lines to these points from the ends of the current segment of the curve. Since point (230, 150) is followed by an *S* operator, the SVG parser must display another segment of the curve starting at point (230, 150). Usually three points are required to display a segment which are two control points and an end anchor point, but in this case the first control point is calculated as a mirroring of the second control point of the previous segment at coordinates (200, 200) relative to the start point of the current segment at coordinates (230, 150). The coordinates of the new control point are calculated as follows: abscissa is calculated as

$$(230 - 200) + 230 = 260,$$

ordinate is calculated as

$$150 - (200 - 150) = 100.$$

So the operator displaying curves of the second and third order can be presented as follows:

```
path d = ″M x_1, y_1 Q x_2, y_2 … T x_i, y_i … C x_i, y_i … S x_k, y_k …″.
```

Some commands can be omitted here. If the command *Q* is omitted, then the command *T* is not used; if the command *C* is omitted, then the command *S* is not used. Any command can be followed by any number of points, but the command will be executed correctly provided that for command *Q* and *S* the number of points is 2*k*, and for command *C* is respectively 3*k*, where *k* is any non-negative integer.

Applying the de Casteljau dividing method to embed a hidden message. According to the method of dividing Bezier curves proposed by de Casteljau [13], any curve can be divided into two parts at some ratio. The result curves (which are segments of the original curve) will also be Bezier curves. Dividing the curve into segments can be used to embed a hidden message. If we divide the Bezier curve into segments at some ratio, then the value of this ratio can hide an element of the secret sequence, which is a digital watermark or digital fingerprint. Since in the general case one additional segment of the Bezier curve is required to embed one element of the sequence, it is rational to use a combined approach: dividing the Bezier curves into segments at a certain ratio and an additional method that will on the one hand increase the number of possible hidden characters of the sequence, and on the other hand it will control the integrity of the hidden data.

Let us show how de Casteljau's method works for a cubic Bezier curve, since for other orders the method is used in a similar way.

Let the cubic Bezier curve ***B*** (fig. 4) be described by four points: $P_1(x_1, y_1)$, $P_2(x_2, y_2)$, $P_3(x_3, y_3)$, $P_4(x_4, y_4)$, at that $P_1(x_1, y_1)$ is the starting point of the curve ***B***, $P_4(x_4, y_4)$ is the ending point, and $P_2(x_2, y_2)$ and $P_3(x_3, y_3)$ are control ones, point $P(x, y)$ divides this curve at some ratio *t*.

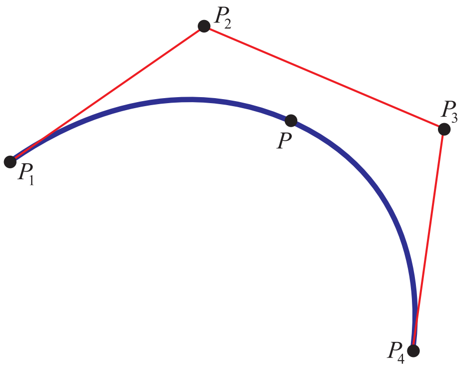


Fig. 4. An original Bezier curve ***B*** and intended dividing point *P*

Red (auxiliary) lines show the location of the original control points P_2 and P_3.

It is necessary to find the location of the anchor and control points of the two resulting Bezier curve segments. These segments, each of which is a cubic Bezier curve, will be denoted as $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$.

Let segment $\boldsymbol{B}_1$ be defined by points $P_{11}(x_{11}, y_{11})$, $P_{12}(x_{12}, y_{12})$, $P_{13}(x_{13}, y_{13})$, $P_{14}(x_{14}, y_{14})$, and segment $\boldsymbol{B}_2$ be defined by points $P_{21}(x_{21}, y_{21})$, $P_{22}(x_{22}, y_{22})$, $P_{23}(x_{23}, y_{23})$, $P_{24}(x_{24}, y_{24})$. Obviously, the starting point of the first curve segment $P_{11}(x_{11}, y_{11})$, is the same as the starting point of the entire curve $P_1(x_1, y_1)$, the ending point of the first curve segment $P_{14}(x_{14}, y_{14})$, and the starting point of the second curve segment $P_{21}(x_{21}, y_{21})$, coincide with the dividing point of the curve $P(x, y)$. The ending point of the second curve segment $P_{24}(x_{24}, y_{24})$, coincides with the ending point of the original curve $P_4(x_4, y_4)$.

The split ratio (t) of the Bezier curve is preserved for the lines connecting the anchor and control points. De Casteljau's method is to recursively divide such segments in the original ratio. The segment P_1P_2 is divided at the ratio t, just like segments P_3P_4 and P_2P_3. Figure 5, *a*, shows points separating the segments P_1P_2, P_3P_4 and P_2P_3 at the t ratio. Then the obtained segments are divided recursively in the same ratio, and the resulting line passes through the separation point P, as shown at fig. 5, *b*.

The intersection points of the line segments become new control points. Thus, we get two Bezier curves: $\boldsymbol{B}_1$ и $\boldsymbol{B}_2$, as indicated at fig. 6.

The coordinates of the starting, ending, and anchor points of the two curve segments can be calculated from the ratio (3). Let us denote as follows:

$$t_0 = 1 - t,\ t_1 = t_0^3,\ t_2 = 3t_0^2 t,\ t_3 = 3t_0 t^2,\ t_4 = t^3.$$

From the ratio (3) we obtain the coordinates of the point $P(x, y)$:

$$x = x_1t_1 + x_2t_2 + x_3t_3 + x_4t_4, y = y_1t_1 + y_2t_2 + y_3t_3 + y_4t_4. \qquad (4)$$

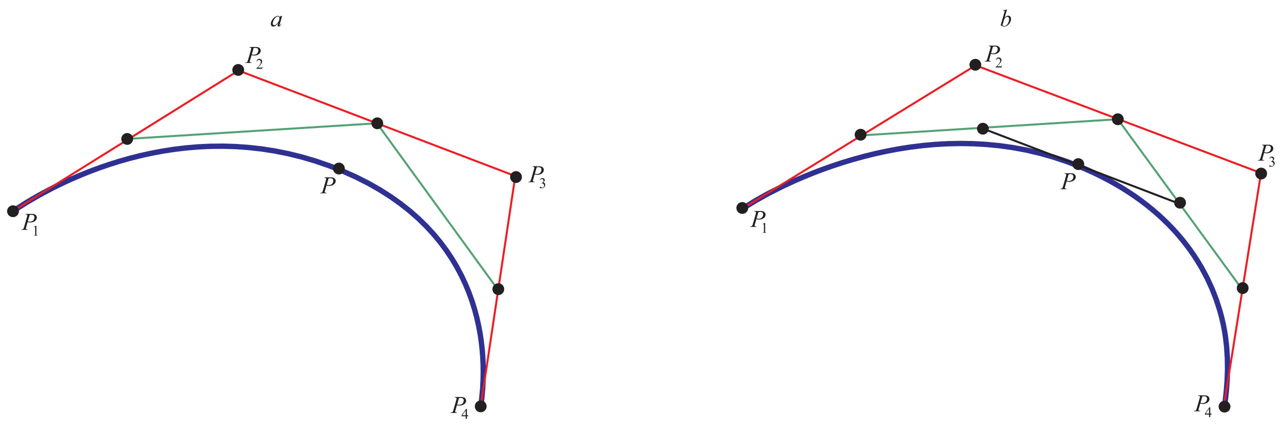


Fig. 5. A subdividing of a cubic Bezier curve using the de Casteljau's method

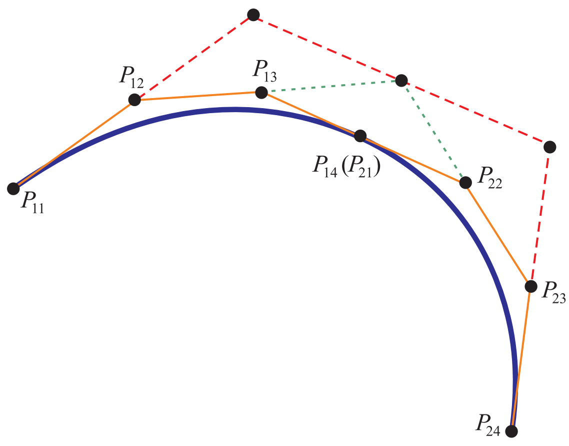


Fig. 6. Bezier curves $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$

Then the coordinates of points $P_{11}(x_{11}, y_{11})$, $P_{12}(x_{12}, y_{12})$, $P_{13}(x_{13}, y_{13})$, $P_{14}(x_{14}, y_{14})$ of the curve $\boldsymbol{B}_1$ can be calculated using the following formulas:

$$x_{11} = x_1,\ y_{11} = y_1;\ x_{12} = x_1t_0 + x_2t,\ y_{12} = y_1t_0 + y_2t;$$

$$x_{13} = x_{12}t_0 + (x_2t_0 + x_3t)t,\ y_{13} = y_{12}t_0 + (y_2t_0 + y_3t)t;\ x_{14} = x,\ y_{14} = y. \tag{5}$$

The coordinates of the points $P_{21}(x_{21}, y_{21})$, $P_{22}(x_{22}, y_{22})$, $P_{23}(x_{23}, y_{23})$, $P_{24}(x_{24}, y_{24})$ of the segment $\boldsymbol{B}_2$ we calculate in a similar way:

$$x_{21} = x_{14},\ y_{21} = y_{14};\ x_{23} = x_3t_0 + x_4t,\ y_{23} = y_3t_0 + y_4t;$$

$$x_{22} = x_{23}t + (x_2t_0 + x_3t)t_0,\ y_{22} = y_{23}t + (y_2t_0 + y_3t)t_0;\ x_{24} = x_4,\ y_{24} = y_4. \tag{6}$$

Thus, the original cubic Bezier curve $\boldsymbol{B}$ can be represented as two segments: $\boldsymbol{B}_1$ и $\boldsymbol{B}_2$, and for the display by the SVG parser it is written in one attribute *d* of the element *path* in general as follows:

path d = "M x_{11}, y_{12} C x_{12}, y_{12} x_{13}, y_{13} x_{14}, y_{14} x_{22}, y_{22} x_{23}, y_{23} x_{24}, y_{24}".

Both curve segments are exactly the same as the original Bezier curve, and the presence of a dividing point is not displayed. Such a divided curve can be written both as separate geometric elements and as a curve of several segments. Further on we will call a curve of several segments a *polycurve.*

Steganographic algorithms for embedding and extracting the hidden message

The summary of the proposed steganographic method is that hidden information is located at the point of dividing the curve into segments.

Let us suppose the hidden message is a binary sequence. In case of dividing the Bezier curve into two segments in half, one symbol of the message can be hidden (the curve is not divided – 0, the curve is divided – 1; can be vice versa). If we choose a division ratio, then we can thereby hide a symbol of a message in a natural language, setting, for example, its own division ratio for each letter. The message will be extracted as follows: we check two successive curves, and if they form a single curve, then we calculate the division ratio, and so we find the hidden symbol.

It is assumed that in general the user of the application will be able to generate values for division on his own, which will be the part of his own steganographic key. Depending on the division ratio, the division point of the curve into segments is calculated and the control points of both new curves are found.

However, firstly, dividing the curve entails the creation of new anchor and control points which increases the file size, and secondly, it is not always possible to accurately calculate the division ratio from coordinates, which leads to the loss of a hidden message. Therefore, we propose to hide only a part of the hidden message in division ratio and to use the emerging anchor and control points for the rest, similar to the LSB method. When receiving new anchor and control points we will change the minor digits in them so that on the one hand this change was visually invisible, but a hidden message was deposited in them, and on the other hand so that during extraction it was possible to check whether the division ratio was extracted correctly.

The hidden message is converted into a binary form, after which it is divided into binary pairs, i. e. the message 01111000 is divided into pairs: 01, 11, 10, 00. For each Bezier curve and each two binary pairs, two actions are performed: the division of the curve at a certain ratio into two segments and embedding of data of two binary pairs into anchor points of segments. Thus, two binary pairs are embedded in one curve.

The method is designed for cubic Bezier curves. For additional embedding of the message, the second anchor point of the first segment (indicated at fig. 6 as P_{13}) and the first control point of the second segment of the curve (indicated at fig. 6 as P_{22}) are used. These points are chosen to embed the message because their coordinates do not affect the restoration of the original curve (which follows from the ratio (5) and (6) and will be demonstrated below with the example). To embed the message, LSB of the anchor point coordinate is used, in this case the sixth (0.00000x). This bit is chosen because of the visual invisibility of changes in coordinates in a vector drawing.

The following two parameters are suggested to be used as key information:

1) division ratio t;

2) values, that correspond the binary pairs V_i, $i = \{1, 2, 3, 4\}$.

Description of the steganographic method algorithm for hidden message embedding. Let us describe an algorithm for the steganographic method of embedding a hidden message in an SVG file, based on the division of cubic Bezier curves at a certain ratio.

The key information is the one of the division ratio t, as well as the values corresponding to the binary pairs V_i, $i = \{1, 2, 3, 4\}$.

The principle of random selection is used to obtain the value of t. It is assumed that this value will be generated on a case-by-case basis and is similar to key generation in symmetric cryptography. The value of t can range from 0.01 to 0.99, assuming that numbers with two decimal places are used.

The principle of random selection is also used to obtain integer values corresponding to binary pairs. For example, you can set the limits shown in table 1, assuming that the limits are numbers from 1 to 99.

Table 1

Correspondence of limits of values written in coordinates to binary pairs

Binary pair value	Limits
00	From 75 to 99
01	From 50 to 74
10	From 25 to 49
11	From 1 to 24

No effect was found for selected values corresponding to binary pairs on the display of a Bezier curve with an embedded message.

First, the SVG file is analysed and the number of cubic Bezier curves N is calculated. The length of the hidden message L is calculated. The maximum length of the hidden message must be less than one half the number of Bezier curves, because one message character is written in two curves, and one more character is needed to indicate the end of the message.

The hidden message is converted into a binary form, after which it is divided into binary pairs Q_k, $k = \{1, \ldots, 4L\}$. Each cubic Bezier curve $\boldsymbol{B}_j$, $j \in [1, N]$, will be successively associated with two binary pairs Q_k. If the length of the embedded message M is less than the number of curves N, then the curves which are not associated with binary pairs remain unchanged.

For each cubic curve $\boldsymbol{B}_j$ and each odd binary pair Q_k, we will perform as long as necessary the division at ratio t if the first symbol of the binary pair is 0, and $1 - t$ if the first symbol of the current binary pair is 1, in accordance with ratio (5) and (6). Thus we obtain a cubic Bezier curve $\boldsymbol{B}'_j$, $j \in [1, N]$, consisting of two segments.

Now, for each segment of the obtained curve $\boldsymbol{B}'_j$ we will embed the data of two binary pairs Q_k into the control points of the segments. The first binary pair is written to the second control point of the first segment of the obtained curve $P_{j13}\left(x_{j13}, y_{j13}\right)$: the first digit of the value is written to LSB of the x_{j13} coordinate, the second digit of the value is written to LSB of the y_{j13} coordinate. Similarly, the second binary pair is written to the coordinates of the first control point of the second segment of the resulting curve $P_{j22}\left(x_{j22}, y_{j22}\right)$. Thus, two binary pairs of the hidden message are written into one original Bezier curve.

The algorithm continues until all binary pairs have been embedded. After that, a new SVG file is generated, consisting of cubic Bezier curves divided into segments, and the original elements. The block diagram of the embedding algorithm is shown in fig. 7.

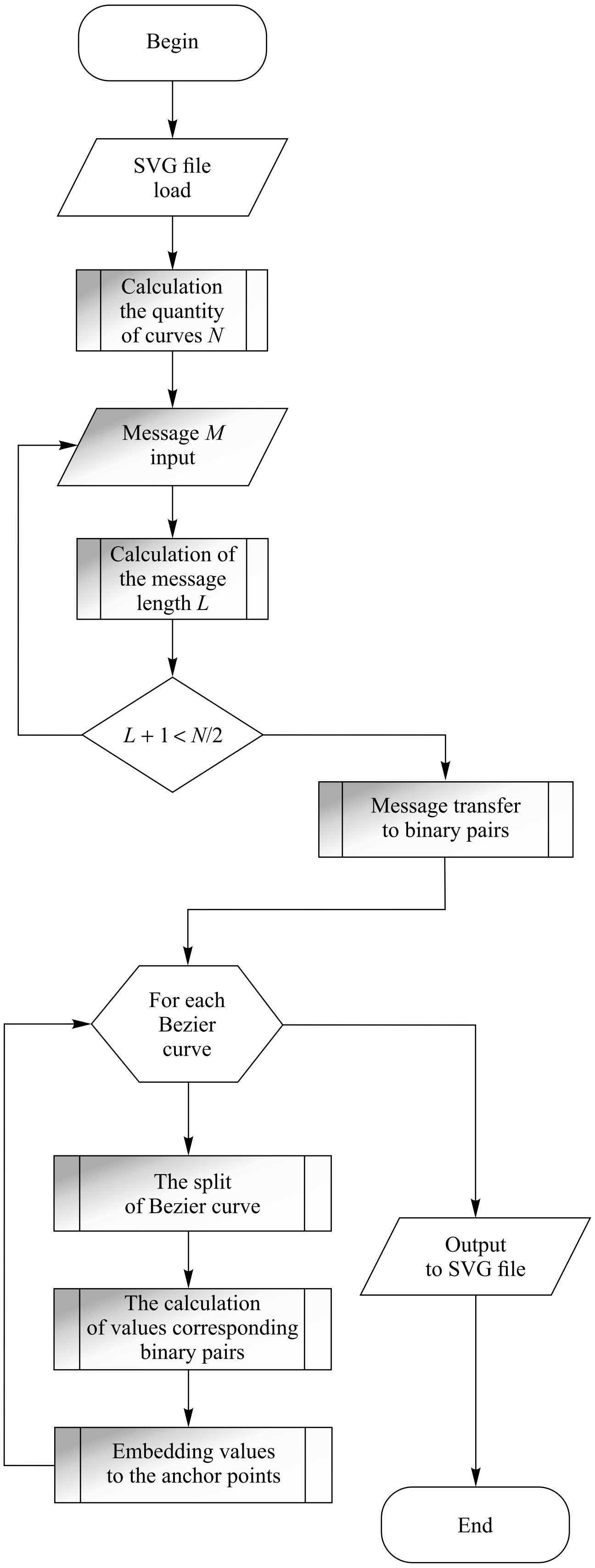

Begin
SVG file load
Calculation the quantity of curves N
Message M input
Calculation of the message length L
$L + 1 < N/2$
Message transfer to binary pairs
For each Bezier curve
The split of Bezier curve
The calculation of values corresponding binary pairs
Embedding values to the anchor points
Output to SVG file
End

Fig. 7. Block diagram of the algorithm of hidden message embedding

Let us consider an example of how the method works. First we will describe the custom key. Let us take the division ratio as $t = 0.25$. The division of the curve into segments at the ratio is as follows: if the first character of a binary pair is 1, then the division ratio is taken as 0.25, if the first character of a binary pair is 0, then the division ratio is taken as 0.75.

Let us take a bijective mapping of a binary pair into the value V_i, $i = \{1, 2, 3, 4\}$, as given in table 2.

Table 2

Correspondence of values of binary pairs to values written in coordinates

Binary pairs value	Value for embedding
00	87
01	64
10	37
11	12

Figure 8 shows the part of the SVG file that is displaying two cubic Bezier curves without an embedded message.

```
<path d = "M 10,200 C 120,30 170,70 220,100"
style = "stroke:blue; stroke-width:2; fill:none;"/>
<path d = "M 10,200 C 180,70 200,200 230,150"
style = "stroke:red; stroke-width:2; fill:none;"/>
```

Fig. 8. Description of two cubic Bezier curves without hidden message

Assume the message to be embedded in this SVG file is 11000110. Let us divide the message into binary pairs: 11, 00, 01, 10. To embed such a message, we need two curves. The first binary pair 11 starts with 1, so the division ratio of the first curve will be 0.25. Let us divide the first curve at the ratio 0.25 and find the coordinates of the anchor and control points according to ratio (5) and (6). The coordinates of the original curve and two new curves are presented in table 3.

Table 3

Coordinates of the Bezier anchor and control points of the original first curve and curves obtained by its division

Curve	x_1	y_1	x_2	y_2	x_3	y_3	x_4	y_4
Original	10	200	120	30	170	70	220	100
1st segment	10	200	37.5	157.5	61.25	128.125	82.1875	108.4375
2nd segment	82.1875	108.4375	145	49.375	182.5	77.5	220	100

Thus, we can rewrite the display of the first curve as

```
path d = "M 10,200 C 37.5,157.5 61.25,128.125
82.1875,108.4375 145,49.375 182.5,77.5 220,100".
```

Now it is necessary to add hidden information to the control points. The first two binary pairs are 11 and 00. They correspond to the values 12 and 87. Therefore, in LSB (0.00000x) of the second control point of the first segment we write **1** and **2**, and in LSB of the first control point of the second segment we write **8** and **7**. Thus, we can rewrite the display of the first curve as

```
path d = "M 10,200 C 37.5,157.5 61.250001,128.125002
82.1875,108.4375 145.000008,49.375007 182.5,77.5 220,100".
```

Let us do the same for the second curve. The third binary pair 01 starts with 0, so the division ratio of the first curve will be 0.75. Let us divide the first Bezier curve at the ratio 0.75 and find the coordinates of the anchor and control points according to ratio (5) and (6). The coordinates of the original curve and two new curves are presented in table 4.

Table 4

Coordinates of the anchor and control points of the original second curve and curves obtained by its division

Curve	x_1	y_1	x_2	y_2	x_3	y_3	x_4	y_4
Original	10	200	180	70	200	200	230	150
1st segment	10	200	137.5	102.5	180.625	151.25	206.875	160.625
2nd segment	206.875	160.625	215.625	163.75	222.5	162.5	230	150

Thus, we can rewrite the display of the second curve as

```
path d = "M 10,200 C 137.5,102.5 180.625,151.25
206.875,160.625 215.625,163.75 222.5,162.5 230,150".
```

Now it is necessary to add hidden information to the control points. The second two binary pairs are 01 and 10. They correspond to the values **64** and **37**. Therefore, in LSB (0.00000x) of the second control point of the first segment we write 6 and 4, and in LSB of the first control point of the second segment we write 3 and 7. Thus, we can rewrite the display of the first curve as

```
path d = "M 10,200 C 137.5,102.5 180.625006,151.250004
206.875,160.625 215.625003,163.750007 222.5,162.5 230,150".
```

The description of the SVG file containing the hidden message is shown in fig. 9.

```
<path d = "M 10,200 C 37.5,157.5 61.250001,128.125002 82.1875,108.4375
145.000008,49.375007 182.5,77.5 220, 100"
style = "stroke:blue; stroke-width:1; fill:none;"/>
<path d = "M 10,200 C 137.5, 102.5 180.625006, 151.250004 206.875, 160.625
215.625003, 163.750007 222.5, 162.5 230, 150"
style = "stroke:red; stroke-width:1; fill:none;"/>
```

Fig. 9. Description of the SVG file of two cubic Bezier curves containing a hidden message

The description of the algorithm for extracting hidden messages. Next we will analyse the algorithm for extracting a hidden message from an SVG file (stego container), based on the division of third-order Bezier curves at a certain ratio. The key information is the one about the division ratio *t* and the values corresponding to binary pairs V_i, $i = \{1, 2, 3, 4\}$.

When extracting information, the file is analysed and the number of cubic Bezier curves *N* in the used container file is calculated. Then a sequential analysis of the curves $\boldsymbol{B}'_j$, $j \in [1, N]$, is performed. If the curve consists of two segments ($\boldsymbol{B}'_{j1}$ и $\boldsymbol{B}'_{j2}$), a check is performed: whether these segments form a single curve $\boldsymbol{B}_j$, and what is the division ratio *t*. The check is carried out in two stages: at the first stage, an assumption is made that the segments form a single Bezier curve, the coordinates of the control points of this curve are calculated; at the second stage it is checked whether the division point of the segments belongs to the assumed curve. If the division point belongs to a curve, then we can conclude that these curves are a single Bezier curve.

Let the segment $\boldsymbol{B}_{j1}$ be defined by the points $P_{j11}(x_{j11}, y_{j11})$, $P_{j12}(x_{j12}, y_{j12})$, $P_{j13}(x_{j13}, y_{j13})$, $P_{j14}(x_{j14}, y_{j14})$, and the segment $\boldsymbol{B}_{j2}$ by the points $P_{j21}(x_{j21}, y_{j21})$, $P_{j22}(x_{j22}, y_{j22})$, $P_{j23}(x_{j23}, y_{j23})$, $P_{j24}(x_{j24}, y_{j24})$. Let find the control and anchor points of the assumed Bezier curve $\boldsymbol{B}_j$. The starting point of the first segment of the curve $P_{j11}(x_{j11}, y_{j11})$ coincides with the starting point of the assumed curve $P_{j1}(x_{j1}, y_{j1})$, the ending point of the second segment of the curve $P_{j24}(x_{j24}, y_{j24})$ coincides with the ending point of the original curve $P_{j4}(x_{j4}, y_{j4})$.

From formula (5) and (6) find the coordinates of the control points of the assumed curve:

$$x_{j2} = \frac{x_{j12} - x_{j1}t_0}{t}, \quad y_{j2} = \frac{y_{j12} - y_{j1}t_0}{t}, \quad x_{j3} = \frac{x_{j23} - x_{j4}t}{t_0}, \quad y_{j3} = \frac{y_{j23} - y_{j4}t}{t_0}.$$

Having determined the parameters of all points of the assumed curve, we check whether the division point of this curve corresponds to the parameter *t*. If this is the case, then the next step of the algorithm is performed;

otherwise, the division ratio $1 - t$ is additionally checked. If in this ratio the curve is divided by a point, then the next step of the algorithm is performed; otherwise, it is assumed that the message is not hidden in this curve.

At the next step of the algorithm, for each segment of the curve $\boldsymbol{B}'_j$ the data of two binary pairs Q_k is extracted from the control points of the segments. From the second control point of the first segment of the original curve $P_{j13}\left(x_{j13}, y_{j13}\right)$ the value corresponding to the first binary pair is extracted: the first digit of the value is extracted from LSB of the coordinate x_{j13}, and the second digit of the value is extracted from LSB of the coordinate y_{j13}. Similarly, the value corresponding to the second binary pair is extracted from the coordinates of the first control point of the second segment of the obtained curve $P_{j22}\left(x_{j22}, y_{j22}\right)$.

From the obtained values according to table 1 the binary pairs are formed, which are combined into a single message. The binary message is converted to text and analysed. If an end-of-message character is found in the recovered string, the message has been recovered.

The block diagram of the extraction algorithm is shown in fig. 10.

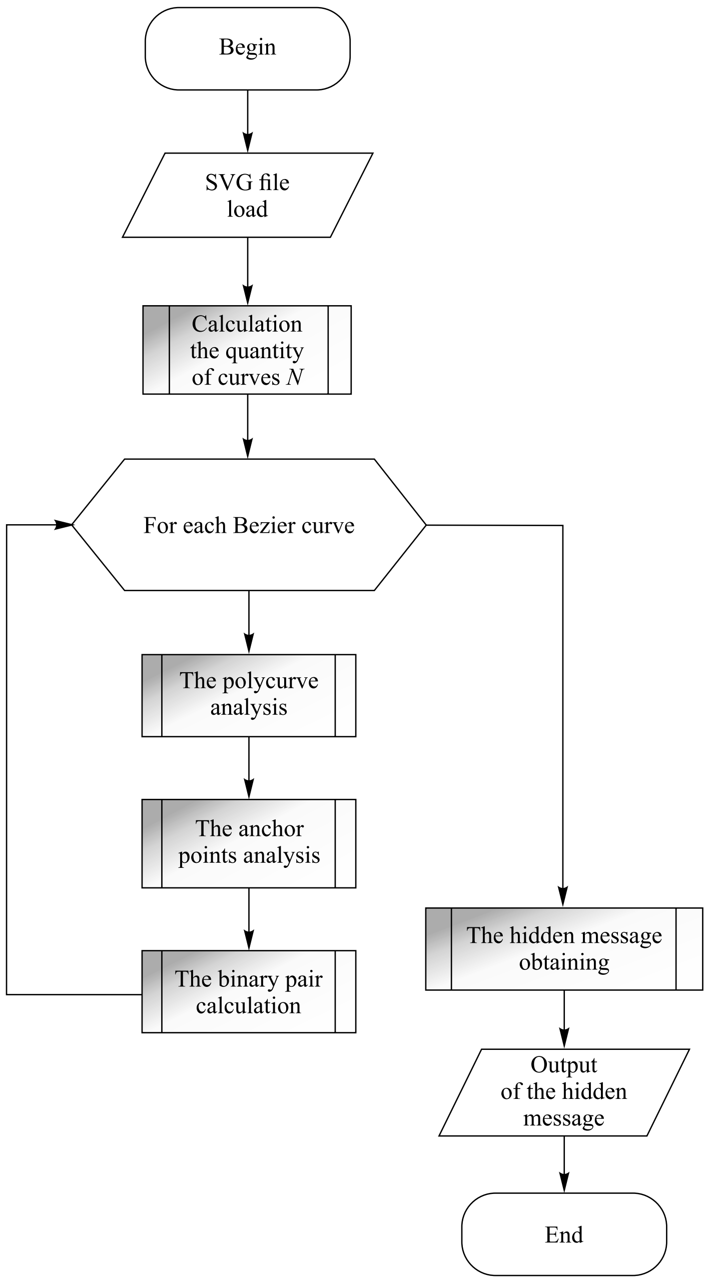

Begin
SVG file load
Calculation the quantity of curves N
For each Bezier curve
The polycurve analysis
The anchor points analysis
The binary pair calculation
The hidden message obtaining
Output of the hidden message
End

Fig. 10. Block diagram of the algorithm of a message extraction from the stego container

Let's continue with an example of how the method works. Assume we have a drawing in SVG format, the description of which is shown in fig. 9.

There are two of third-order Bezier curves in this image. Consider the first curve. It consists of two segments. We will check two stages: at the first stage we assume that the segments form a single Bezier curve and calculate the coordinates of the control points of this curve; at the second stage we check whether the division point of the segments belongs to the assumed curve. If the division point belongs to a curve, then we can conclude that this curve is a single Bezier curve.

From formulas (5) and (6), knowing x_1, x_{12}, t and t_0, it is possible to obtain x_2. Similarly, knowing y_1, y_{12}, t and t_0, it is possible to obtain y_2. Just the same, knowing x_4, x_{23}, t and t_0, it is possible to obtain x_3, and knowing y_4, y_{23}, t and t_0, it is possible to obtain y_3. We know that the division ratio t can be either 0.25 or 0.75, depending on the first symbol of the binary pair, if the binary pair is hidden in this curve. Assume that $t = 0.25$. Based on this, we calculate the coordinates of the assumed Bezier curve: $P_1(10, 200)$, $P_2(120, 30)$, $P_3(170, 70)$, $P_4(220, 100)$. Let's make sure that the point $(82.1875, 108.4375)$ belongs to the assumed curve. From formula (4) we obtain the coordinates of the division point of the assumed curve at the ratio 0.25, this is the point $(82.1875, 108.4375)$, therefore, the point $(82.1875, 108.4375)$ belongs to the curve, so the curve is a single Bezier curve. Consider points $P_{13}(61.250001, 128.125002)$ and $P_{22}(145.000008, 49.375007)$. At these points we expect to find an embedded message. Let's extract the values from the least significant digit of the coordinates: from the first coordinate 61.250001 we extract the last significant digit, i. e. 1, from the second coordinate 128.125002 we extract the last significant digit, i. e. 2, we get the first value 12. Similarly, we get 87 from the second control point. According to the table 2 we get binary pairs 11 and 00.

Let us do the same for the second curve. It also has two segments. It is necessary to check if these segments are a single Bezier curve and what is the division ratio. Suppose that the division ratio is $t = 0.25$, and from formulas (5) and (6) we calculate the coordinates of the assumed curve. We obtain the coordinates of the assumed Bezier curve: $P_1(10, 200)$, $P_2(520, -190)$, $P_3(220, 166.6667)$, $P_4(230, 150)$. Let us check if the point $(206.875, 160.625)$ belongs to the obtained curve. From formula (4) we obtain the coordinates of the division point of the assumed curve at the ratio 0.25, this is the point $(258.125, 30)$, therefore, the division ratio is not 0.25. Assume that the division ratio is $t = 0.75$, and from formulas (5) and (6) we calculate the coordinates of the assumed curve. We obtain the coordinates of the assumed Bezier curve: $P_1(10, 200)$, $P_2(180, 70)$, $P_3(200, 200)$, $P_4(230, 150)$. Let us check if the point $(206.875, 160.625)$ belongs to the obtained curve. From formula (5) we obtain the coordinates of the division point of the assumed curve at the ratio of 0.25, this is the point $(206.875, 160.625)$. Therefore, the point $(206.875, 160.625)$ belongs to the curve, this curve is a single Bezier curve. Consider points $P_{13}(180.625006, 151.250004)$ and $P_{22}(215.625003, 163.750007)$. At these point, we expect to find an embedded message. Let's extract the values from the least significant digits of the coordinates: 64 and 37, respectively. According to the table 1 we get binary pairs 01 and 10.

Thus, the extracted message will look like 11000110. The message has been recovered correctly.

Results and discussion

To implement the described steganographic method, the DLL StegoSVG library was developed. The library contains classes that implement the analysis of a file in SVG format, counting the number of cubic Bezier curves, dividing a message into binary pairs, dividing cubic Bezier curves into segments, both algorithms for embedding and extracting messages.

A desktop application StegoSVG Demo has been created to demonstrate the method. The application interface can be roughly divided into three areas: the configuration area, the analysis area, and the working area. The analysis area contains information obtained during the analysis of the file, such as the time of the beginning and the end of the analysis, the number of third-order Bezier curves found in the file, the maximum possible message length, errors that occur when the library or application is running. The working area, depending on the operating mode, provides the ability to add or extract a message. Figure 11 shows a general view of the application in the hidden message embedding mode. This is a small SVG file of 100 kilobytes.

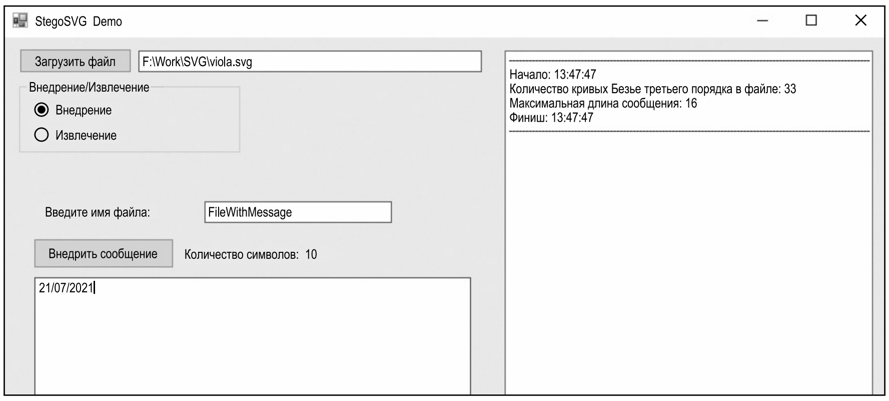
To test the operation of the program SVG images were taken from the site *www.freesvg.org*. The results of the program are presented in table 5.

Not all images were suitable for analysis as they did not contain cubic Bezier curves. Ten SVG files containing cubic Bezier curves and suitable for the implementation of the watermark were analysed. The hidden message consisted of 25 characters (Stegano Message 21/07/2021), with the exception of file No. 5 where a hidden message of 10 characters was embedded (21/07/2021).

Table 5

Results of hidden message embedding into SVG files

No.	Source file size (V_1), bytes	The number of curves	The size of stego container (V_2), bytes	The difference ($V_1 - V_2$), bytes
1	5 115 022	17 882	5 126 306	11 284
2	5 028 074	17 878	5 042 548	14 474
3	429 566	472	435 923	6357
4	1 426 801	13 048	1 432 974	6173
5	45 826	40	48 340	2514
6	136 111	52	140 410	4299
7	117 750	297	122 281	4531
8	108 998	144	113 419	4421
9	83 261	219	87 666	4405
10	122 579	229	127 187	4608



F:\Work\SVG\viola.svg

[● Внедрение] ○ Извлечение

Введите имя файла: FileWithMessage

Количество символов: 10

21/07/2021

```
Начало: 13:47:47
Количество кривых Безье третьего порядка в файле: 33
Максимальная длина сообщения: 16
Финиш: 13:47:47
```

Fig. 11. StegoSVG Demo application interface

File sizes are in bytes. We also note that the difference in size between the original and the file and the corresponding steganographic file is not constant. The increase in the size of the latter, of course, depends on the length of the message, but also depends on the original image. Since the hidden message is embedded in the new control points, and the calculation accuracy is no less than 0.000001, then, depending on the initial coordinates of the anchor and control points of the Bezier curve, the number of characters in the file description can increase when fractional coordinates appear.

Visually, the original file and the steganographic file look the same. When viewed in specialised software (Corel DRAW v.20), the number of objects and curves is the same, but the number of points and file size are different.

When changing a file with a watermark with specialised software, the sign can either be preserved or destroyed, depending on the type of changes. When converting a steganographic file using Corel DRAW v.20 to PNG and TIFF raster formats and vice versa, the mark is completely lost, because from raster formats the file is restored close to the original, but not exactly the same. When converting a steganographic file to EPS vector format and vice versa, the mark is also lost. In this case the file also does not return to its original state, all coordinates change. When archiving a file with an embedded message using ZIP and RAR archivers, the watermark remains after unpacking.

Conclusion

A new steganographic method is considered for embedding and extracting hidden messages when using SVG files as stego containers. The method is based on modifying the parameters of cubic Bezier curves. In particular, a combined approach is used to embed digital watermarks or digital labels in additional points of Bezier curves, as well as in additional lower digits of control points. The combined use of several methods of introducing a secret message allows to increase the length of the hidden sequence, as well as to control the integrity of the message being embedded.

To implement the method, the StegoSVG library was developed, which was used in the development of the author's desktop application (StegoSVG Demo). Several files of the SVG format taken from open sources have been analysed for the possibility of placing steganographic DWMs in them.

The method can be used to apply digital watermarks or digital labels on graphic images in SVG format, in which cubic Bezier curves are present. With some modifications the method can be applied to SVG images with quadratic Bezier curves. Two Bezier curves are required to embed one message symbol, so this method is also suitable for relatively small images. Two parameters are the steganographic key: the ratio of dividing the original curve into segments and the correspondence of the values of binary pairs to the values written in LSB of coordinates.

Due to the specificity of the path tag used to embed hidden information into an image, the method cannot be directly used in other kinds of XML containers such as electronic documents and electronic maps, however, an approach that connects the separation of an object in some way and a record of hidden information in LSB of emerging points seems to be quite promising.

Since the SVG format provides web designers with tremendous opportunities in the implementation of static and interactive animated images for sites, the proposed method can be used to protect sites from fakes. Phishing protection can also be developed on this basis.

Further research in this direction is of interest from the point of view of ensuring a given level of steganographic resistance of the method, for example, based on the development and use of the corresponding key information.

Библиографические ссылки

1. Грибунин ВГ, Оков ИН, Туринцев ИВ. *Цифровая стеганография.* Москва: Солон-пресс; 2009. 264 с.

2. Urbanovich P, Chourikov K, Rimorev A, Urbanovich N. Text steganography application for protection and transfer of the information. *Przeglad elektrotechniczny.* 2010;7:95–97.

3. Agarwal M. Text steganographic approaches: a comparison. *International Journal of Network Security & Its Applications.* 2013; 1(5):91–106. DOI: 10.5121/ijnsa.2013.5107.

4. Blinova E, Shutko N. The use of steganographic methods in SVG format graphic files. In: *New electrical and electronic technologies and their industrial implementation. Proceedings of the 10th International conference; 2017 June 23–26; Zakopane, Poland.* Lublin: Lublin University of Technology; 2017. p. 45.

5. Шутько НП. Защита и передача текстовой информации на основе изменения кернинга. *Труды БГТУ. Серия 3. Физико-математические науки и информатика.* 2017;2:92–95.

6. Shutko N, Urbanovich P, Zukowski P. A method of syntactic text steganography based on modification of the document-container aprosh. *Przegląd elektrotechniczny.* 2018;6:82–85. DOI: 10.15199/48.2018.06.15.

7. Блинова ЕА, Сущеня АА. Применение нескольких стеганографических методов для осаждения скрытых данных в электронных текстовых документах. *Системный анализ и прикладная информатика.* 2019;2:32–38. DOI: 10.21122/2309-4923- 2019-2-32-38.

8. Урбанович ПП, Юрашевич ДЭ. Использование системных свойств и параметров текстовых файлов в стеганографических приложениях. В: Харин ЮС, Чернявский АФ, Берник ВИ, Кучинский ПВ, Курбацкий АН, Агиевич СВ, редакторы. *Теоретическая и прикладная криптография. Материалы Международной научной конференции; 20–21 октября 2020 г.; Минск, Беларусь.* Минск: БГУ; 2020. с. 68–73.

9. Kaur D, Verma HK, Singh RK. Image steganography: hiding secrets in random LSB pixels. In: Pant M, Sharma TK, Verma OP, Singla R, Sikander A, editors. *Soft computing: theories and applications.* Singapore: Springer; 2020. p. 331–341 (AISC; volume 1053). DOI: 10.1007/978-981-15-0751-9_31.

10. Subramanian N, Elharrouss O, Al-Maadeed S, Bouridane A. Image steganography: a review of the recent advances. *IEEE Access.* 2021;9:23409–23423. DOI: 10.1109/access.2021.3053998.

11. Блинова ЕА, Урбанович ПП. Стеганографический метод на основе встраивания дополнительных значений координат в изображения формата SVG. *Труды БГТУ. Серия 3. Физико-математические науки и информатика.* 2018;2:104–109.

12. Блинова ЕА, Голик АА. Модификация стеганографического метода на основе встраивания дополнительных значений координат в изображения формата SVG. В: Тузиков АВ, Григянец РБ, Венгеров ВН, редакторы. *Развитие информатизации и государственной системы научно-технической информации (РИНТИ-2018). Доклады XVII Международной конференции; 20 сентября 2018 г.; Минск, Беларусь.* Минск: ОИПИ НАН Беларуси; 2018. с. 130–133.

13. Farin GE, Hansford D. *The essentials of CAGD.* 1st edition. Natick: A. K. Peters Ltd.; 2000. 242 p.

References

1. Gribunin VG, Okov IN, Turintsev IV. *Tsifrovaya steganografiya* [Digital steganography]. Moscow: Solon-press; 2009. 264 p. Russian.

2. Urbanovich P, Chourikov K, Rimorev A, Urbanovich N. Text steganography application for protection and transfer of the information. *Przeglad elektrotechniczny.* 2010;7:95–97.

3. Agarwal M. Text steganographic approaches: a comparison. *International Journal of Network Security & Its Applications.* 2013; 1(5):91–106. DOI: 10.5121/ijnsa.2013.5107.

4. Blinova E, Shutko N. The use of steganographic methods in SVG format graphic files. In: *New electrical and electronic technologies and their industrial implementation. Proceedings of the 10th International conference; 2017 June 23–26; Zakopane, Poland.* Lublin: Lublin University of Technology; 2017. p. 45.

5. Shutko NP. Protection and transfer of text information on the basis of kerning changing. *Trudy BGTU. Seriya 3. Fiziko-matematicheskie nauki i informatika.* 2017;2:92–95. Russian.

6. Shutko N, Urbanovich P, Zukowski P. A method of syntactic text steganography based on modification of the document-container aprosh. *Przegląd elektrotechniczny.* 2018;6:82–85. DOI: 10.15199/48.2018.06.15.

7. Blinova EA, Sushchenia AA. Several steganographic methods using for embedding of hidden data in electronic text documents. *Sistemnyi analiz i prikladnaya informatika.* 2019;2:32–38. Russian. DOI: 10.21122/2309-4923-2019-2-32-38.

8. Urbanovich PP, Yurashevich DE. [Using system properties and text file settings in steganographic applications]. In: Kharin YuS, Chernyavskii AF, Bernik VI, Kuchinskii PV, Kurbatskii AN, Agievich SV, editors. *Teoreticheskaya i prikladnaya kriptografiya. Materialy Mezhdunarodnoi nauchnoi konferentsii; 20–21 oktyabrya 2020 g.; Minsk, Belarus'* [Theoretical and applied cryptography. Materials of an International scientific conference; 2020 October 20–21; Minsk, Belarus]. Minsk: Belarusian State University; 2020. p. 68–73. Russian.

9. Kaur D, Verma HK, Singh RK. Image steganography: hiding secrets in random LSB pixels. In: Pant M, Sharma TK, Verma OP, Singla R, Sikander A, editors. *Soft computing: theories and applications.* Singapore: Springer; 2020. p. 331–341 (AISC; volume 1053). DOI: 10.1007/978-981-15-0751-9_31.

10. Subramanian N, Elharrouss O, Al-Maadeed S, Bouridane A. Image steganography: a review of the recent advances. *IEEE Access.* 2021;9:23409–23423. DOI: 10.1109/access.2021.3053998.

11. Blinova EA, Urbanovich PP. A steganographic method based on the embedding of additional coordinates into images of SVG format. *Trudy BGTU. Seriya 3. Fiziko-matematicheskie nauki i informatika.* 2018;2:104–109. Russian.

12. Blinova EA, Golik AA. [The modification of the steganographic method based on the embedding of additional coordinates into images in SVG format]. In: Tuzikov AV, Grigyanets RB, Vengerov VN, editors. *Razvitie informatizatsii i gosudarstvennoi sistemy nauchno-tekhnicheskoi informatsii (RINTI-2018). Doklady XVII Mezhdunarodnoi konferentsii; 20 sentyabrya 2018 g.; Minsk, Belarus'* [Development of informatisation and the state system of scientific and technical information (RINTI-2018). Reports of the 17th International conference; 2018 September 20; Minsk, Belarus]. Minsk: Joint Institute for Informatics Problems of the National Academy of Sciences of Belarus; 2018. p. 130–133. Russian.

13. Farin GE, Hansford D. *The essentials of CAGD.* 1st edition. Natick: A. K. Peters Ltd.; 2000. 242 p.

Received 02.09.2021 / revised 12.10.2021 / accepted 09.11.2021.