УДК 003.26

А. Н. Николайчук[1], П. П. Урбанович[1,2]
[1]Белорусский государственный технологический университет
[2]Люблинский Католический университет Яна Павла II, Польша

СТЕГАНОГРАФИЧЕСКИЙ МЕТОД НА ОСНОВЕ ИСПОЛЬЗОВАНИЯ ОСОБЕННОСТЕЙ ОТОБРАЖЕНИЯ ЭЛЕМЕНТОВ В ФОРМАТЕ SVG

Рассмотрены важные особенности файлов векторной графики, уникальная структура SVG-файлов, а также особенности формирования элементов, составляющих графические объекты в данном цифровом формате. Проанализированы области просмотра файла SVG и их параметры. Содержимое SVG-файла описывается на бесконечном холсте любого размера. Однако при описании фигур необходимо учитывать области просмотра изображения. На основе анализа вышеперечисленных особенностей параметров отображения геометрических фигур в данной статье обоснован и описан новый стеганографический метод и реализующие его алгоритмы встраивания (извлечения) скрытых данных при использовании векторных изображений формата SVG в качестве стеганоконтейнера. Одним из важных типов элементов таких контейнеров являются кривые Безье. В качестве модифицируемых параметров контейнера используются координаты, описывающие ключевые точки кривых Безье. Внедрение тайной информации в файл-контейнер предполагает формирование координат ключевых точек кривой Безье и размещение последовательности этих точек на кривой. При этом кривая будет иметь значения координат, превышающих границы отображения (видимости) объектов изображения. Представлено авторское программное средство, реализующее разработанный метод. Показаны примеры использования приложения. Метод и программное средство могут использоваться для защиты электронного контента от несанкционированного применения.

Ключевые слова: стеганография, векторные изображения, SVG-формат, viewport, кривая Безье.

Для цитирования: Николайчук А. Н., Урбанович П. П. Стеганографический метод на основе использования особенностей отображения элементов в формате SVG // Труды БГТУ. Сер. 3, Физико-математические науки и информатика. 2023. № 1 (266). С. 64–70. DOI: 10.52065/2520-6141-2023-266-1-11.

A. N. Nikolaichuk[1], P. P. Urbanovich[1,2]
[1]Belarusian State Technological University
[2]The John Paul II Lublin Catholic University

A STEGANOGRAPHIC METHOD BASED ON THE USE OF THE FEATURES OF ELEMENTS DISPLAYING IN SVG FORMAT

The important features of vector graphics files, the unique structure of SVG files, as well as the features of the formation of elements that make up graphic objects in this digital format are considered. The viewing areas of the SVG file and their parameters are analyzed. The contents of the SVG file are described on an infinite canvas of any size. However, it is necessary to take into account the viewing areas of the image to describe shapes. Based on the analysis of the above-mentioned features of the parameters for displaying geometric shapes, this article substantiates and describes a new steganographic method and algorithms for embedding/extracting hidden data by using vector images of the SVG format as a steganocontainer. One of the important types of elements of such containers are Bezier curves. Coordinates, which set key points of Bezier curves are used as modifiable container parameters. Data hiding into a container file involves the creation of key points' coordinates of the Bezier curve and the insertion of a sequence of this curve points. In this case, the curve will have coordinates' values that exceed the area of the displaying (visibility) of the image objects. Special software application implementing the developed method is presented. Examples of using the application are shown. The method and software can be used to protect electronic content from unauthorized use.

Keywords: steganography, vector graphics, SVG, viewport, Bezier curve.

For citation: Nikolaichuk A. N., Urbanovich P. P. A steganographic method based on the use of the features of elements displaying in SVG format. *Proceedings of BSTU, issue 3, Physics and Mathematics. Informatics*, 2023, no. 1 (266), pp. 64–70. DOI: 10.52065/2520-6141-2023-266-1-11 (In Russian).

Введение. В настоящее время информация со всеми ее важными свойствами и особенностями использования стала важнейшим экономическим ресурсом. Обеспечение безопасности информационных ресурсов – одна из самых важных задач в ИТ-сфере. Эти ресурсы нуждаются в защите от несанкционированного доступа, кражи, изменения либо уничтожения.

Одним из способов решения анализируемой проблемы является применение методов стеганографии, которые позволяют скрыто передавать данные внутри некоторого информационного объекта – носителя информации или контейнера.

Применение методов стеганографии в цифровых объектах (цифровой стеганографии) предполагает преобразование контейнера (файла, в котором скрывают данные) таким образом, чтобы внесенные изменения не отразились на визуальном отображении модифицированного контейнера – стеганоконтейнера [1]. Одним из таких стеганоконтейнеров являются файлы векторной графики, которые в последнее время становятся объектом изучения специалистов в области стеганографии и являются активно используемыми при создании web-ресурсов.

Векторная графика представлена широким разнообразием форматов, например, PDF, AI, EPS, CDR, SVG. Уникальным среди форматов векторной графики является формат SVG (Scalable Vector Graphics), имеющий структуру XML-документа, который на самом деле является текстовым файлом. Следовательно, к таким файлам могут быть применены классические методы текстовой стеганографии, а также методы, применяемые к файлам разметки: методы подмены и перестановки атрибутов или метод замены регистра тегов [2–4].

SVG имеет много преимуществ перед web-приложением: например динамическое отображение данных и интерактивность. По мере расширения области использования графики SVG в коммерческих web-приложениях для обмена или публикации данных несанкционированное копирование и распространение данных на основе SVG становится проблемой защиты авторских прав для создателей и владельцев многих web-приложений [5–10].

В данной статье предлагается новый стеганографический метод внедрения (извлечения) данных с использованием векторных изображений, а также описываются алгоритмы его реализации.

Основная часть. Хотя структура SVG-формата позволяет оперировать текстом, ценность формата заключается в описании графических объектов. Объекты в SVG основаны на математическом представлении элементарных геометрических фигур, таких как линии, круги, окружности, эллипсы, многоугольники. Они размещаются в файле с помощью специальных тегов, которые описывают каждую фигуру, учитывая ее особенности. Например, для описания круга, необходимо указать название тега <*circle*> и поставить значения атрибутов *cx* – координата центра круга по оси *X*; *cy* – координата центра круга для оси *Y*; *r* – значение радиуса [3–5].

Для создания сложных графических объектов используется общий элемент <*path*>.Тег <*path*> определяется одним атрибутом – *d*, содержащим серию команд и параметров, используемых этими командами, которые определяют траекторию и направление линии фигуры. Команды обозначаются буквами. Например, команда *M* в качестве параметров принимает координаты точки, обозначающей начальное положение; команда *L* принимает координаты и рисует прямую линию от текущего положения к этой точке; команда *Z* используется для замыкания контура. Вместе с тем для создания плавных кривых линий существует несколько различных команд, среди которых – отрисовка кривых Безье [11]. С помощью элемента <*path*> можно описывать и примитивы, используя для этого соответствующие команды. Однако чаще всего этот тег используется для изображения фигур, состоящих из комбинации линий разного типа. Поэтому такие фигуры могут включать до нескольких сотен и даже тысяч значений координат, что позволяет тегу быть контейнером с большой емкостью [6–7].

Содержимое SVG-файла описывается на бесконечном холсте и может быть любого размера, однако при описании фигур необходимо учитывать его области просмотра. Их две: системная (*viewport*) и пользовательская. Начало координат области *viewport* располагается в левом верхнем углу экрана и не объявляется. Для того чтобы задать размеры этой области, необходимо установить значение атрибутов корневого тега <*svg*>: *height* – высота и *width* – ширина. Пользовательская область просмотра устанавливается с помощью атрибута *viewBox*, значение которого принимает четыре параметра. С помощью последних задаются размеры: *min-x* – начало оси координат *X*; *min-y* – начало оси координат *Y*; *width* – ширина; *height* – высота.

SVG позволяет управлять как размерами *viewport*, так и поведением содержимого относительно него: растягиваться (масштабироваться) с потерей пропорций, увеличиваться или уменьшаться в масштабе, и самое главное – обрезаться. Этим поведением можно управлять с помощью изменения параметров областей просмотра. Таким образом, можно описывать фигуры, которые по значениям координат находятся за пределами границ областей просмотра

содержимого, но при этом не будут видны в изображении. Эта особенность имеет принципиальное значение для разработки стеганографических методов. Мы также ее использовали в нашем методе.

Для наглядности и понимания сущности метода рассмотрим два изображения, в которых фигуры описаны одинаково, различие составляют лишь значения ширины и высоты.

На рис. 1 показано содержимое изображения с квадратами, которые определены с помощью команд для рисования кривой Безье. Так как значения ширины и высоты изображения не установлены, обе фигуры видны пользователю (рис. 2).

```
<svg xmlns="http://www.w3.org/2000/svg">
        <path d="M10 10 H 90 V 90 H 10 L 10 10" fill="red"/>
        <path d="M110 10 H 190 V 90 H 110 L 110 110" fill="black"/>
</svg>
```

Рис. 1. Содержимое файла SVG без установленных значений ширины и высоты области *viewport*

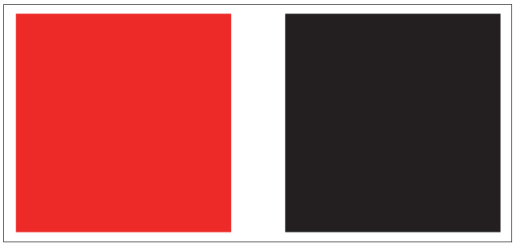

Рис. 2. Файл SVG без установленных значений ширины и высоты области *viewport*

Содержимое изображения на рис. 3 представлено точно такими же квадратами, как и на рис. 1. Однако во втором случае (рис. 3) границы видимости (того, что видит пользователь), то есть размер ширины и высоты изображения (указываются в атрибутах *width* и *height* тега <path>), имеют значения в 100 пикселей. Кривая Безье, отображающая квадрат черного цвета, имеет значения начальной точки с координатой 110 пикселей по оси *X* и 10 пикселей по оси *Y*.

```
<svg width="100" height="100"
        xmlns="http://www.w3.org/2000/svg">
        <path d="M10 10 H 90 V 90 H 10 L 10 10" fill="red"/>
        <path d="M110 10 H 190 V 90 H 110 L 110 110" fill="black"/>
</svg>
```

Рис. 3. Содержимое файла SVG с установленными значениями ширины и высоты изображения

Значение, установленное для начальной точки по оси *X*, превышает ширину изображения, установленную в атрибуте *width*, как и все последующие координаты по оси *X*, что делает фигуру невидимой для пользователя. Точно таким же образом можно описывать фигуры, превышающие значения высоты изображения.

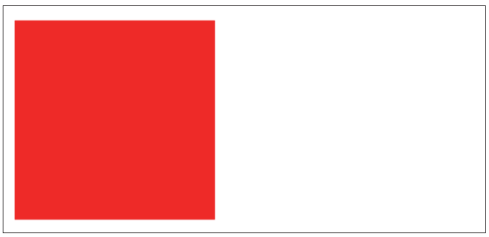

Рис. 4. Файл SVG с установленными значениями ширины и высоты изображения

С помощью этих условий фигуры, описанные в содержимом файла, не отображаются на экране (рис. 4): данная область является невидимой.

На основе использования вышеперечисленных особенностей параметров видимости геометрических фигур и предлагается новый метод встраивания данных в векторное изображение. В качестве модифицируемых параметров контейнера используются координаты, описывающие ключевые точки кривых Безье, так как графический объект такого типа может иметь наибольшее число значений, которыми можно описать фигуру.

Осаждение скрытой информации осуществляется путем создания кривой со значениями координат ключевых точек, вычисленных таким образом, чтобы кривая создавалась с координатами, превышающими значения ширины и высоты изображения. Необходимо хранить эти значения, но существуют также изображения, в которых они не установлены. В этом случае используются значения рассмотренного выше атрибута *viewBox*. Если же и он отсутствует, эти значения будут вычисляться путем удвоения самых больших значений координат, относящихся ко всем графическим объектам.

Чтобы однозначно определить пользователя (владельца ресурса – контейнера), для него генерируется уникальная псевдослучайная последовательность, состоящая из шести цифр. Обозначим ее *U*. Каждая из этих цифр будет использована для формирования кривой и размещения тайной информации в контейнере.

Для того чтобы рассматриваемые координаты превышали видимую область изображения, значения ширины и высоты перемножаются на некоторое число. Этим числом является первая цифра пользовательского кода: U_1 ($U_1 \in U$; $U_1 > 1$). Обозначим результаты этих вычислений так: *W* – для ширины и *H* – для высоты:

$$W = U_1 \cdot width; \quad (1)$$

$$H = U_1 \cdot height. \quad (2)$$

Четыре следующие цифры $U_2 – U_5$ (U_2, …, $U_5 \in U$) используются для внедрения сообщения. Вычисление координат кривой Безье осуществляется с помощью сообщения, представленного в бинарном виде, которое разбивается на пары, каждая пара при этом имеет порядковый номер $–j$. Каждая координата кривой определяет отдельную бинарную пару. Для вычисления этих координат используется одно из значений пользовательского кода, которое определяется следующим образом: если значением бинарной пары будет комбинация '00' – в формировании кривой используется U_2 (вторая цифра кода), аналогично для пары '01' – U_3, '10' – U_4, '11' – U_5. При вычислении координат кривой с внедренным сообщением используется произведение $j \cdot U_i$, для того чтобы параметры кривой не дублировались.

Так как кривая Безье создается с помощью ключевых точек, которые обозначаются координатами по осям x и y, то целесообразно вычислять для разных осей разные значения, так как ширина и высота изображения могут не совпадать.

Для вычисления координат предлагаются следующие формулы:

$$Kx = W \cdot (1 + j \cdot U_i / 10^{c+1}); \quad (3)$$

$$Ky = H \cdot (1 + j \cdot U_i / 10^{c+1}), \quad (4)$$

где $i = 2 – 5$; c – количество разрядов произведения $j \cdot U_i$.

Отметим также, что координаты в SVG-изображениях могут принимать самые различные значения, например, 1000 или 0,00001, поэтому в (3) и (4) используется выражение, которое даст возможность вычислять кривую, ориентируясь на размеры изображения, что позволит при одних и тех же значениях пользовательского кода и сообщения формировать разные кривые Безье.

Чтобы связать кривую с содержимым исходного файла, необходимо определить тег E. Для этого используется последняя цифра кода – U_6 следующим образом:

$$E = n \bmod U_6. \quad (5)$$

В зависимости от содержимого контейнера параметр n из (5) будет вычисляться по-разному:

1) если файл SVG содержит теги <*path*>, то n – количество таких тегов; в таком случае в E будет внедряться сообщение и его начальные координаты, Sx (для оси X) и Sy (для оси Y), будут использоваться при вычислении параметров кривой;

2) если файл не содержит теги <*path*>, то n – количество тегов, описывающих геометрические объекты. В таком случае ключевые координаты E (Sx и Sy) будут использоваться при вычислении параметров кривой, а для самой кривой создается новый тег, который будет размещен после вычисленного.

В результате для определения координат ключевых точек предлагается использовать следующие эмпирические выражения:

$$x = W \cdot (1 + j \cdot U_i / 10^{c+1}) + Sx; \quad (6)$$

$$y = H \cdot (1 + j \cdot U_i / 10^{c+1}) + Sy. \quad (7)$$

Использование начальных значений координат тега E в формулах (6) и (7) позволяет повысить устойчивость стеганоконтейнера к операциям сдвига изображения. Это осуществляется благодаря тому, что необходимая информация для внедрения секретного сообщения вычисляется по формулам (3) и (4), а начальные значения прибавляются в формулах (6) и (7) лишь для того, чтобы вычисленные значения оставались неизменными при условии, если все графические объекты будут сдвинуты. Такая операция сдвига может рассматриваться как тип несанкционированной модификации стеганоконтейнера.

Для рассматриваемого метода нет ограничения на использование определенных изображений, но в качестве рекомендации предлагается использовать файлы, содержащие кривые Безье.

Рассмотрим алгоритм стеганографического внедрения тайной информации на основе предлагаемого метода. Этот алгоритм состоит в выполнении следующих операций.

1. Сгенерировать пользовательский код U.

2. Сформировать внедряемое сообщение M.

3. Преобразовать сообщение в бинарную последовательность (обозначим ее B).

4. Разбить B на пары, присваивая каждой паре порядковый номер j.

5. Выбрать файл-контейнер C.

6. Если значения границ областей просмотра *viewport* (*width* и *height*) и *viewBox* существуют, перейти к п. 8, иначе – к п. 7.

7. Определить максимальные значения координат по осям x и y и установить эти удвоенные значения в атрибуты *width* и *height* тега <*svg*> соответственно.

8. Вычислить значения параметров W и H, используя (1) и (2).

9. Подсчитать общее количество фигур (n) файла-контейнера и вычислить остаток от деления E на последнюю цифру пользовательского кода U_6 с помощью (5).

10. Получить значения координат начальной точки элемента E (Sx и Sy).

11. Вычислить координаты ключевых точек с помощью формул (6) и (7).

12. Сформировать кривую Безье, состоящую из команд и соответствующих им координат (на основе вычисленных значений ключевых точек).

13. Если тег E является элементом <*path*>, перейти к п. 15, иначе – к п. 14.

14. Создать тег <*path*> с атрибутом *d* и задать ему значение сформированной кривой, перейти к п. 16

15. Добавить к имеющемуся значению атрибута *d* значение сформированной кривой, перейти к п. 16.

16. Сформировать SVG-файл.

Для извлечения сообщения *M* из стеганоконтейнера необходимо использовать следующий алгоритм.

1. Определить пользовательский код *U*.

2. Получить файл-стегоконтейнер *C*.

3. Получить значения границ областей просмотра *C*: *viewport* (*width* и *height*) и *viewBox*.

4. Вычислить значения постоянных *W* и *H*, перемножив значения атрибутов *width* и *height* тега с первой цифрой пользовательского кода U_1, используя (1) – для значений *width* и (2) – для значений *height.*

5. Подсчитать общее количество элементов *n* файла-контейнера и вычислить остаток от деления *E* на последнюю цифру пользовательского кода (U_6), используя (5). С учетом особенностей внедрения сообщения из-за содержимого файла *n* для извлечения будет определяться по-другому:

5.1) если файл-контейнер содержит больше одного элемента <*path*>, то *n* – количество элементов <*path*>;

5.2) если элемент <*path*> один и значения всех ключевых точек больше значений *width* и *height*, то *n* – количество всех графических объектов – 1;

5.3) если элемент <*path*> один и не все значения ключевых точек больше значений *width* и *height*, то *n* – количество элементов <*path*>.

6. Вычислить тег со значениями кривой, превышающими *width* и *height.*

Если файл-контейнер содержит больше одного элемента <*path*> и вычисленный тег совпадает с *E*, значит, содержимое исходного сообщения *M* не изменялось.

Если элемент <*path*> один и значения всех ключевых точек больше значений *width* и *height*, и вычисленный тег совпадает с тегом, следующим после *E*, то содержимое *M* не изменялось.

Если элемент <*path*> один и не все значения ключевых точек больше значений *width* и *height*, и вычисленный тег совпадает с *E*, значит, содержимое *M* не изменялось.

7. Получить значения координат начальных точек тега *E*: *Sx* и *Sy*.

8. Получить координаты ключевых точек кривой Безье, которые размещаются за границами областей просмотра, присваивая каждой координате порядковый номер *j*.

9. Последовательно вычислить значения с помощью следующих формул:

$$Mx = ((x - Sx) / W - 1) / j; \quad (8)$$

$$My = ((y - Sy) / H - 1) / j. \quad (9)$$

10. Из-за того, что при внедрении учитывается количество разрядов произведения $j \cdot U_i$, необходимо рекурсивно умножать значения *Mx* и *My*, пока оно не будет больше единицы. Для полученных параметров сопоставить значения бинарных пар в соответствии с пользовательским кодом следующим образом: если значение *M*_ будет равно U_2, то бинарной парой будет '00', аналогично для U_3 – '01', U_4 – '10', U_5 – '11'.

11. Из бинарных пар п. 10 сформировать бинарную последовательность.

12. Преобразовать сообщение из бинарной последовательности п. 11 в символьную строку.

Для демонстрации предложенного метода нами разработано приложение. В качестве технологии для создания приложения была выбрана ASP.NET MVC для удобного отображения SVG-изображений и выполнения необходимых манипуляций с элементами разметки за счет реализации представления с помощью файлов *cshtml*. Кроме того, использовалась библиотека *System.Xml.XmlDocument* для работы с XML-файлами, а также были созданы классы, аналогичные фигурам в SVG, что упрощает работу со значениями их атрибутов. Были созданы классы *Embedder* и *Extracter* для реализации функций внедрения и извлечения сообщения.

Для инициализации приложения необходимо зарегистрироваться, чтобы для пользователя была вычислена псевдослучайная последовательность, которая будет являться пользовательским кодом (*U*). Затем выбирается файл-контейнер *C*.

SVG-файл отображается в двух вариантах: непосредственно изображение и его содержимое.

После этого вводится сообщение *M*. При нажатии кнопки «ОК» сообщение внедряется в SVG-файл, затем, при успешном внедрении, изменение можно наблюдать в окне *Содержимое*, при этом само изображение остается неизменным. И осажденный файл *M* будет доступен для скачивания (рис. 5).

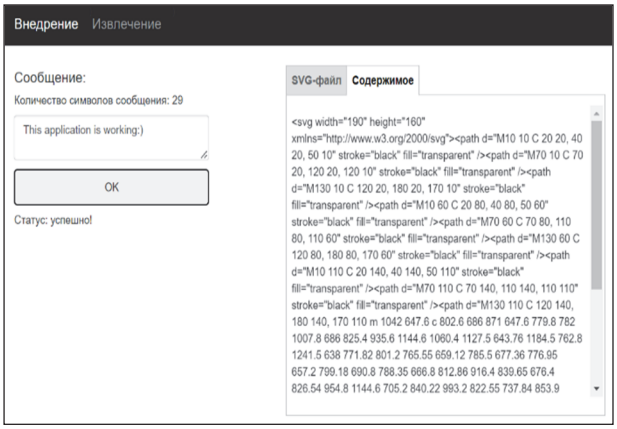

Рис. 5. Вид окна приложения после внедрения сообщения в контейнер

Для извлечения сообщения необходимо авторизоваться и выбрать файл с внедренным сообщением.

Если элементы содержимого файла не подвергались модификациям, то сообщение успешно извлекается из контейнера (рис. 6).

Рис. 6. Вид окна приложения после извлечения сообщения из контейнера

Заключение. Цифровые изображения, содержащие большое число геометрических фигур, в особенности элементов с тегом <*path*>, являются хорошими стеганографическими контейнерами с точки зрения объема внедряемой тайной информации и устойчивости стеганоконтейнеров к случайным или преднамеренным модификациям.

SVG позволяет управлять как размерами *viewport*, так и поведением содержимого относительно него: растягиваться (масштабироваться) с потерей пропорций, увеличиваться или уменьшаться в масштабе, и самое главное – обрезаться. Этим поведением можно управлять с помощью изменения параметров областей просмотра. Таким образом, можно описывать фигуры, которые по значениям координат находятся за пределами границ областей просмотра содержимого, но при этом не будут видны в изображении. На этой основе разработан описанный в статье стеганографический метод. Ввиду особенности формирования кривой Безье метод позволяет минимизировать увеличение размера исходного файла, так как сообщение внедряется в содержимое файла, которое не отображается на экране вовсе, не приводя фигуры к искажению и используя все контрольные точки. В этом состоит отличие предлагаемого метода от известных методов данного класса. Кроме того, при реализации предложенного метода нет ограничений на длину внедряемого в файл-контейнер тайного сообщения, так как координаты отображаемых фигур не задействованы в необходимых вычислениях. Такие вычисления координат повышают устойчивость стеганоконтейнера к атакам на основе аффинных преобразований.

Представляет научный и практический интерес дальнейшее расширение исследований в данной предметной области, связанное с оценкой пропускной способности канала на основе предложенных стеганографических преобразований и уточнением уровня стеганографическй стойкости метода к различным типам атак.

Список литературы

1. Урбанович П. П. Защита информации методами криптографии, стеганографии и обфускации. Минск: БГТУ, 2016. 220 с.

2. Blinova E., Shutko N. The use of steganographic methods in SVG format graphic files // Proc. of the 10th Intern. Conf. New Electrical and Electronic Technologies and their Industrial Implementation. Lublin, 2015. P. 45.

3. Text Steganography utilizing XML, HTML And XHTML Markup Languages / S. Imran [et al.] // International Journal of Computational Geometry & Applications. 2017. № 3. P. 99–116.

4. Almutairi A. A Comparative Study on Steganography Digital Images: A Case Study of Scalable Vector Graphics (SVG) and Portable Network Graphics (PNG) Images Formats // (IJACSA) International Journal of Advanced Computer Science and Applications. 2018. Vol. 9, no. 1. P. 170–175.

5. Блинова Е. А., Урбанович П. П. Стеганографический метод на основе встраивания дополнительных значений координат в изображения формата SVG // Труды БГТУ. Сер. 3, Физико-математические науки и информатика. 2018. № 1. С. 104–109.

6. Николайчук А. Н., Урбанович П. П. Анализ стеганографических методов на основе контейнеров SVG-формата // Информационные технологии: материалы 86-й науч.-техн. конф. проф.-препод. состава, научных сотрудников и аспирантов, Минск, 31 января – 12 февраля 2022 г. Минск, 2022. С. 49–51.

7. Николайчук А. Н., Урбанович П. П. Стеганография в векторных изображениях // 73-я науч.-техн. конф. учащихся, студентов и магистрантов: сб. науч. работ, Минск, 18–23 апреля 2022 г. Минск: БГТУ, 2022. С. 947–949.

8. Zhou X. and Pan. X. Watermark-Based Scheme to Protect Copyright of SVG Data, 2006 International Conference on Computational Intelligence and Security. 2006. P. 1199–1202. DOI: 10.1109/ICCIAS.2006.295245.

9. Badr Almutairi. A New Steganography Method for Scalable Vector Graphics (SVG) Images Based On An Improved LSB Algorithm // International Journal of Computer Science and Network Security. 2019. Vol. 19, no. 10. P. 99–104.

10. Kuznetsov A., & Kononchenko G. Steganographic methods in vector graphics // Radiotekhnika. 2021. № 2 (205). P. 32–41. DOI: 10.30837/rt.2021.2.205.03.

11. Blinova E. A, Urbanovich P. P. Steganographic method based on hidden messages embedding into Bezier curves of SVG images // Journal of the Belarusian State University. Mathematics and Informatics. 2021. No. 3. P. 68–83. DOI: https://doi.org/10.33581/2520-6508-2021-3-68-83.

References

1. Urbanovich P. P. *Zashchita informatsii metodami kriptografii, steganografii i obfuskatsii* [Information protection by cryptography, steganography and obfuscation methods]. Minsk, BGTU Publ., 2016. 220 p. (In Russian).

2. Blinova E., Shutko N. The use of steganographic methods in SVG format graphic files. *Proc. of the 10th Intern. Conf.* "New Electrical and Electronic Technologies and their Industrial Implementation". Lublin, 2015, p. 45.

3. Imran S. [et al.]. Text Steganography utilizing XML, HTML And XHTML Markup Languages. *International Journal of Computational Geometry & Applications*, 2017, no 3, pp. 99–116.

4. Almutairi A. A Comparative Study on Steganography Digital Images: A Case Study of Scalable Vector Graphics (SVG) and Portable Network Graphics (PNG) Images Formats. *International Journal of Advanced Computer Science and Applications*, 2018, vol. 9, no. 1, pp. 170–175.

5. Blinova E. A., Urbanovich P. P. A steganographic method based on the embeddingof additional coordinates into images of SVG format. *Trudy BGTU* [Proceedings of BSTU], issue 3, Physics and Mathematics. Informatics, 2018, no. 1, pp. 104–109 (In Russian).

6. Nikolaichuk A. N., Urbanovich P. P. Analysis of steganographic methods based on SVG-format containers. *Informatsionnyye tekhnologii: materialy 86-y nauchno-tekhnicheskoy konferentsii professorsko-prepodavatel'skogo sostava, nauchnykh sotrudnikov i aspirantov* [Information technologies: materials of the 86th Scientific and Technical Conference of the teaching staff, researchers and postgraduates]. Minsk, 2022, pp. 49–51 (In Russian).

7. Nikolaichuk A. N., Urbanovich P. P. Steganography in vector images. *73-ya nauchno-tekhnicheskaya konferentsiya uchashchikhsya, studentov i magistrantov: sbornik nauchnykh rabot* [73rd Scientific and Technical Conference of students, undergraduates and undergraduates: collection of scientific papers]. Minsk, 2022, pp. 947–949 (In Russian).

8. Zhou X. and Pan X. Watermark-Based Scheme to Protect Copyright of SVG Data. *International Conference on Computational Intelligence and Security*, 2006, pp. 1199–1202. DOI: 10.1109/ICCIAS.2006.295245.

9. Badr Almutairi. A New Steganography Method for Scalable Vector Graphics (SVG) Images Based On An Improved LSB Algorithm. *International Journal of Computer Science and Network Security*, 2019, vol. 19, no. 10, pp. 99–104.

10. Kuznetsov A., & Kononchenko G. Steganographic methods in vector graphics. *Radiotekhnika*, 2021, no. 2 (205), pp. 32–41. DOI: 10.30837/rt.2021.2.205.03.

11. Blinova E. A., Urbanovich P. P. Steganographic method based on hidden messages embedding into Bezier curves of SVG images. *Journal of the Belarusian State University. Mathematics and Informatics*, 2021, no. 3, pp. 68–83. DOI: 10.33581/2520-6508-2021-3-68-83.

Информация об авторах

Николайчук Александра Николаевна – магистрант кафедры информационных систем и технологий. Белорусский государственный технологический университет (220006, г. Минск, ул. Свердлова, 13а, Республика Беларусь). E-mail: nikolaichukalexandra@gmail.com

Урбанович Павел Павлович – доктор технических наук, профессор, профессор кафедры информационных систем и технологий. Белорусский государственный технологический университет (220006, г. Минск, ул. Свердлова, 13а, Республика Беларусь). E-mail: p.urbanovich@belstu.by

Information about the authors

Nikolaichuk Aleksandra Nikolaevna – Master's degree student, the Department of Information Systems and Technologies. Belarusian State Technological University (13a, Sverdlova str., 220006, Minsk, Republic of Belarus). E-mail: nikolaichukalexandra@gmail.com

Urbanovich Pavel Pavlovich – DSc (Engineering), Professor, Professor, the Department of Information Systems and Technologies. Belarusian State Technological University (13a, Sverdlova str., 220006, Minsk, Republic of Belarus). E-mail: p.urbanovich@belstu.by

Поступила 30.12.2022