УДК 681.3.06

ОПТИМИЗАЦИЯ СТЕГАНОГРАФИЧЕСКОГО АЛГОРИТМА ОСАЖДЕНИЯ ИНФОРМАЦИИ В КОНТЕЙНЕР ФОРМАТА EPUB НА ОСНОВЕ ПРЕДВАРИТЕЛЬНОГО СЖАТИЯ МОДИФИЦИРУЕМОГО ИЗОБРАЖЕНИЯ

Сущеня А.А.

Белорусский государственный технологический университет,

Минск, Республика Беларусь,

e-mail: asuschenya@gmail.com

В докладах [1, 2] представлен алгоритм, а также реализующее его программное средство стеганографического преобразования файла формата *EPUB*. В результате выполнения процедуры осаждения информации в оригинальную электронную книгу встраивается сообщение, его копия, а также хэш осаждаемого сообщения. Наличие осаждаемых частей призвано увеличить стеганографическую стойкость. Однако наряду с рядом преимуществ, ввиду особенностей алгоритма, контейнер претерпевает изменения, существенно влияющие на его размер. Это по большей части обусловлено тем, что происходит работа с изображениями, которые вследствие дублирования существенно увеличивают размер контейнера. В статье предлагается поиск мест в алгоритме за счет которых происходит чрезмерное увеличение размера финального контейнера, а также оптимизация этих мест. В результате чего ожидается уменьшение размера итогового стеганографического контейнера.

Рассмотрим изменения, влияющие на размер контейнера на конкретном примере. Возьмем электронную книгу формата *EPUB* размером 1051 *KB*. Используя программное средство, реализующее стеганографический алгоритм, описанное в [1], осадим сообщение «*Secret Message*» в выбранную электронную книгу. После успешного завершения операции внедрения сообщения размер исходного контейнера увеличился с 1051 *KB* до 2094 *KB*.

Внедрение информации, согласно алгоритму [1], производится в три дополнительных контейнера, каждый из которых представляет собой определенный формат данных: главы книги, обложка книги, а также ее стилистическое оформление. За каждый отдельный тип данных отвечает соответствующий ему участок программного кода. Все участки участвующие в процедуре осаждения сведены в отдельный класс приложения *EmbedController* (листинг 1).

```
31  Embedder chapterEmbedder = new ChapterEmbedder(book, messgeDTO.MessageHash);
32  Embedder cssEmbedder = new CSSEmbedder(book, messgeDTO.Message);
33  Embedder lsbEmbedder = new LSBEmbedder(book, messgeDTO.Message);
34
35  chapterEmbedder.Next = cssEmbedder;
36  cssEmbedder.Next = lsbEmbedder;
37
38  chapterEmbedder.EmbedMessage();
```

Листинг 1 — Класс, конструирующий цепочку экземпляров для запуска процедуры стеганографического осаждения информации

С 31 по 33 строки создаются экземпляры соответствующих классов, которые будут конструировать будущий контейнер. Для того, чтобы определить какой из используемых методов больше всего увеличивает размер финального контейнера S_{Full} поочередно отключим добавление каждого последующего элемента. Параллельно необходимо заново проделывать процедуру осаждения идентичного сообщения, наблюдая за тем, как уменьшается размер итогового стеганографического контейнера. Затем, для вычисления размера, добавляемого каждым методом, необходимо вычесть из размера стеганографического контейнера, размер без учета каждого метода. Тем самым, мы получим дельту, на которую увеличивается размер контейнера с использованием каждого отдельного метода.

Приведем пример вычисления размера S_{XHTML}, на который увеличивается первоначальный контейнер $S_{Original}$ после применения метода замены кавычек в файлах разметки [3-7]. Размер итогового стеганографического контейнера вычисляется по формуле 1.

$$S_{Full} = S_{Original} + S_{XHTML} + S_{LSB} + S_{CSS}, \quad (1)$$

где S_{LSB} размер в *KB* на который увеличивается $S_{Original}$ после применения стеганографического метода *LSB*, а S_{CSS} размер, на который увеличивается $S_{Original}$ после применения метода закодированного изображения в каскадных таблицах стилей [8]. В соответствии с формулой 1, S_{XHTML} можно определить следующим образом (формула 2).

$$S_{XHTML} = S_{Full} - (S_{Original} + S_{LSB} + S_{CSS}). \quad (2)$$

На основании полученных результатов построим график размеров, которые занимают отдельные методы, увеличивающие место в S_{Full} (рисунок 1).

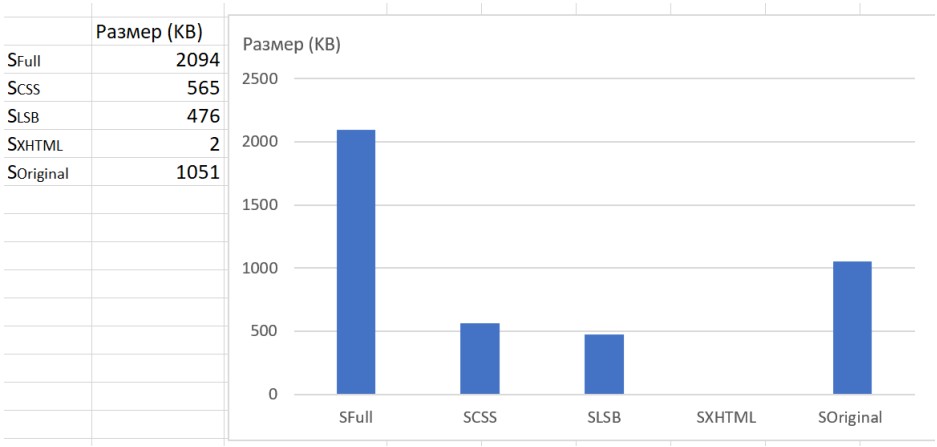

Рисунок 1 — График размеров контейнеров,
занимаемых отдельными стеганографическими методами

На основании рисунка 1 можно сделать следующие выводы: методами, наиболее сильно влияющими на S_{Full} — являются метод *LSB*, а также метод закодированного изображения в каскадных таблицах стилей, где S_{CSS} = 565 *KB*, а S_{LSB} = 476 *KB*. Следовательно, для достижения цели уменьшения S_{Full} после всех преобразований либо же перед ними необходимо, на сколько это возможно, уменьшить размеры S_{CSS} и S_{LSB}.

Для уменьшения размера итогового контейнера предлагается выполнять операцию сжатия изображения до внедрения в него сообщения. Согласно стеганографической модели, представленной в источнике [1], новая функция встраивания сообщения **F** будет выглядеть следующим образом:

$$\mathbf{F} : \mathbf{M}\ \{M_O, M_H\} \times \mathbf{C}\ \{\mathrm{Compr}(C_{JPG}), \mathrm{Compr}(C_{CSS}), C_{XHTML}\} \times \mathbf{K}\ \{K_{LSB}, K_{CSS}, K_Q\} \longrightarrow \mathbf{S},$$

где *Compr* — функция сжатия контейнера, содержащего изображение перед произведением процедуры стеганографического осаждения информации.

За счет того, что сжатие контейнера происходит перед основной процедурой осаждения, функция извлечения $\mathbf{F}^{-1}$, а также алгоритм извлечения остаются неизменными. В алгоритм же внедрения добавляется два дополнительных элемента, отвечающих за сжатие двух типов контейнеров (рисунок 2).

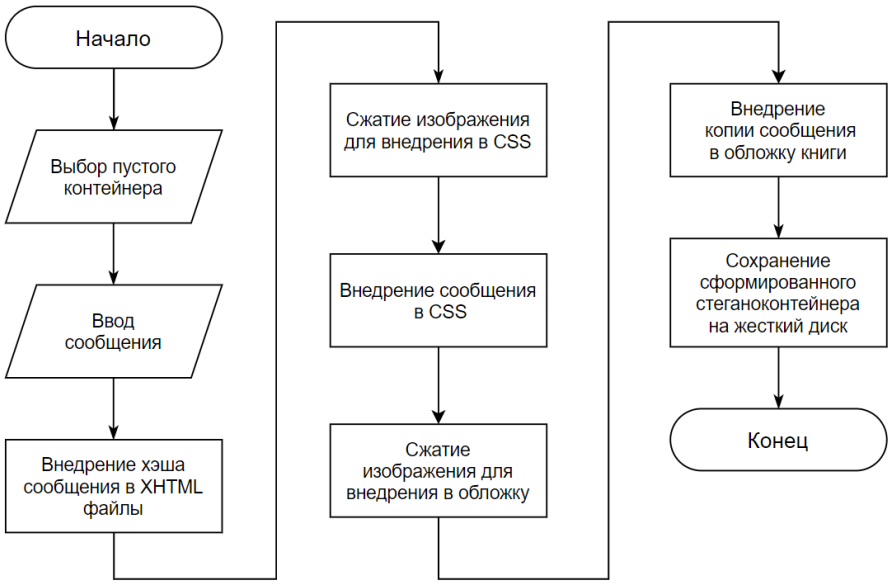

Рисунок 2 — Блок-схема модифицированного алгоритма осаждения сообщения в книгу формата EPUB

Несмотря на то, что изображение, которое внедряется в обложку и таблицы стилей является одинаковым для обоих контейнеров процедуры сжатия для него будет различаться. В случае с таблицами стилей сжатие изображения можно произвести с упором на минимальный размер финального изображения. Так как в итоге сообщение будет сконвертировано в строку формата *Base64* и не будет доступно для просмотра пользователем, качеством самого изображения можно пренебречь для достижения минимально возможного размера. В случае же с обложкой степень сжатия необходимо будет регулировать в зависимости от параметров картинки для того, чтобы качество полученного изображения никак не повлияло на его семантический смысл и четкость.

Начнем с оптимизации алгоритма осаждения в таблицы стилей. Для сжатия изображения используем функционал класса *Image* входящего в *.NET Framework*. Класс *Image* – абстрактный базовый класс предоставляет функциональные возможности для производных классов *Bitmap* и *Metafile*. *Bitmap* используется для работы с пиксельными изображениями. *Metafile* определяет графический метафайл, содержащий записи, описывающие последовательность графических операций, которые могут быть записаны (созданы) и воспроизведены (отображается). С помощью класса *Image* можно создавать изображения на основе данных из таких графических файлов, как *BMP*, *JPG* и *PNG*. Обложки именно этих типов файлов чаще всего используются в книгах формата *EPUB*.

Создадим отдельный класс *CompressionHelper* в котором определим метод *CompressImageWithNewQuality*. Возвращаемым параметром метода будет являться объект класса *Image*. Именно уменьшенная версия этого объекта в дальнейшем будет сконвертирована в строку формата *Base64*. На вход метода подается четыре параметра. Первый *image* – объект класса *Image* который нужно изменить. Далее следует три целочисленных аргумента *newWidth* – ширина сжатого изображения, *newHeight* – высота сжатого изображения и *newQuality* – задающий качество для нового изображения. Параметр *newQuality* определяет уровень сжатия изображения. При использовании конструктора класса *EncoderParameter* диапазон значений для качества изображения составляет от 0 до 100. Чем меньше указанное число, тем выше степень сжатия и, следовательно, тем ниже качество изображения. Ноль дает нам изображение самого низкого качества что в конкретном случае позволит изображение нужного размера.

Для произведения процедуры сжатия изображения необходимо на объекте класса *Image* вызвать метод *Save*. Параметрами для данного метода буду являться: созданный при помощи конструктора по умолчанию объект класса *MemoryStream*, параметры о типе изображения в виде объекта *ImageCodecInfo*, а также заполненный объект *EncoderParameters* который хранит информацию о качестве. После вызова *Save*, изображение с заданными

параметрами будет сохранено в *MemoryStream*. Извлекаем сохраненное изображение из потока вызовом *Image.FromStream* возвратив полученный результат из метода. Вызов созданного метода *CompressImageWithNewQuality* с передачей необходимых для проведения процедуры сжатия изображения представлен в листинге 2.

```
var imageInByteArrayReadyToLSB = StegoHelpers.CompressionHelper
                    .CompressImageWithNewQuality(imageContent, 10, 10, 50);
```

Листинг 2 — Вызов метода *CompressImageWithNewQuality* с передачей параметров для осаждения в *CSS* контейнер

В листинге 2 метод вызывается со следующими параметрами: высота и ширина нового изображения будут по 10 пикселей, в то время как качество изображения будет ухудшено на 50 процентов. Экспериментальным путем установлено, что значение 50 в данном случае является оптимальным при уменьшении размера картинки.

Проверим внесенные изменения на практике. Возьмем книгу, которую мы использовали в самом начале размером 1051 *KB*. Произведем процедуру стеганографического осаждения информации с внесенными в программный код изменениями. Результатом внесенных правок в алгоритм размер полученного контейнера $\mathbf{S_{Full}}$ составил 1537 *KB*. Таким образом размер контейнера $\mathbf{S_{CSS}}$ уменьшился на 557 *KB*. За вычетом размера, на который уменьшился итоговый контейнер, новый размер $\mathbf{S_{CSS}}$ составляет 8 *KB*. На основании полученных результатов можно сказать, что алгоритм внедрения сообщения в *CSS* контейнер был успешно оптимизирован.

Далее необходимо оптимизировать алгоритм внедрения сообщения в саму обложку книги. Используя уже созданный метод *CompressImageWithNewQuality* необходимо подобрать такие параметры нового изображения, при которых конечному пользователю не будет заметен результат компрессии.

Основным отличием от осаждения в *CSS* в дополнение к качеству изображения добавляется его ширина и высота. Так как сжимаемое изображение будет видно конечному пользователю необходимо сохранить пропорции, а также оригинальное разрешение изображения. В листинге 3 представлен вызов метода *CompressImageWithNewQuality* для сжатия обложки.

```
Image extractedImage;
using (var ms = new MemoryStream(imageContent))
{
    extractedImage = Image.FromStream(ms);
}
var imageInByteArrayReadyToLSB = StegoHelpers.CompressionHelper
    .CompressImageWithNewQuality(imageContent, extractedImage.Width, extractedImage.Height, 35);
```

Листинг 3 — Вызов метода *CompressImageWithNewQuality* с передачей параметров для осаждения в обложку книги

В списке аргументов, идущих после переданного изображения, следуют высота и ширина изменяемой картинки, что позволит уменьшить искажение при сжатии. Заключительным аргументом является число 35 представляющее собой качество изображения, являющееся оптимальным минимумом. Данное число было экспериментально проверено в результате проведения операций сжатия с другими числами. На рисунке 3 можно увидеть результаты сравнения качества сжатых изображений в зависимости от переданного параметра качества.

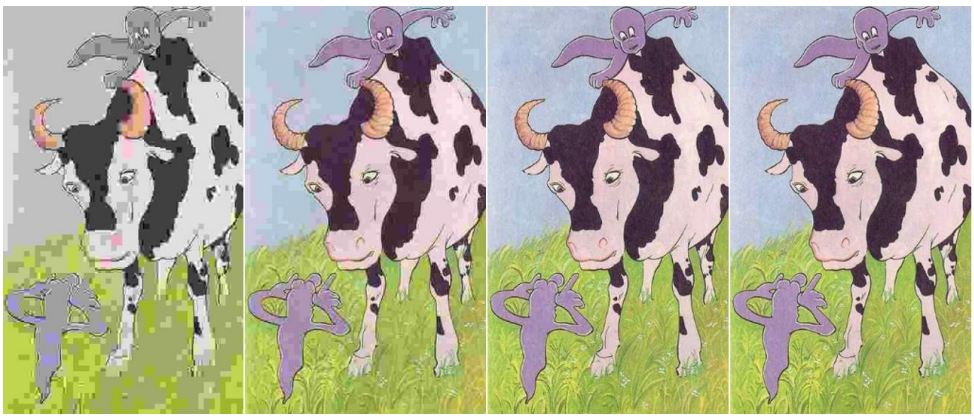

Рисунок 3 — Полученные изображения в результате изменения параметра качества при следующих значениях (слева на право): 1, 10, 35, 50.

Как видно на рисунке 3 разница в качестве изображений между значениями 35 и 50 не заметна для пользователя, следовательно 35 является приемлемым для использования аргументом.

После внесения описанных изменений в программный код проследим за изменением размера контейнеров $\mathbf{S_{Full}}$ и $\mathbf{S_{LSB}}$. При успешном встраивании сообщения «Secret Message» во взятую для примера книгу значение $\mathbf{S_{Full}}$ после применении всех изменений по оптимизации становится равным 1452 *KB*. За вычетом полученного значения размер $\boldsymbol{S_{LSB}}$ после оптимизации становится равным 391 *KB*. В результате оптимизации размер финального контейнера сократился на 642 *KB*.

В процессе исследования процедуры стеганографического преобразования книги формата *EPUB* были выявлены особенности алгоритма осаждения оптимизация которых уменьшила итоговый размер стеганоконтейнера. Для оптимизации были применены процедуры сжатия изображения перед стенографическим преобразованием файлов *CSS*, а также файлов обложки книги. Внесены изменения в реализующее алгоритм программное средство с учетом добавления нового функционала по сжатию изображений. Проведено исследование измененного программного средства, подтверждающее успешность оптимизации алгоритма.

Список литературы:

1. Сущеня, А. А. Математическая модель стеганографической системы с использованием стеганографического контейнера в виде электронной книги формата EPUB / А. А. Сущеня, Е. А. Блинова // Труды БГТУ. Сер. 3, Физико-математические науки и информатика. – Минск: БГТУ, 2020. – № 1 (230). – С. 57-62
2. Сущеня, А. А. Методы и программное средство стеганографического преобразования текстов-контейнеров на основе языков разметки / А. А. Сущеня // Сборник научных работ студентов Республики Беларусь «НИРС 2018» / редкол. : И. А. Старовойтова (пред.) [и др.]. — Минск : Изд. центр БГУ, 2019. – С. 219-222.
3. Сущеня, А. А. Стеганографическое преобразование текстов-контейнеров на основе языков разметки / А. А. Сущеня // 68-я научно-техническая конференция учащихся, студентов и магистрантов, 17-22 апреля, Минск : сборник научных работ : в 4 ч. Ч. 4 / Белорусский государственный технологический университет. - Минск : БГТУ, 2017. - С. 145-149.
4. Урбанович П. П. Защита информации методами криптографии, стеганографии и обфускации. Минск: БГТУ, 2016. 220 с.
5. Конахович Г. Ф., Пузыренко А. Ю. Компьютерная стеганография. Теория и практика. Киев: МК-Пресс, 2006. 288 с.
6. Сущеня, А. А. Программное средство стеганографического преобразования текстов контейнеров на основе языка разметки XML/ А. А. Сущеня // 69-я научно-техническая конференция учащихся, студентов и магистрантов, 2-13 апреля, 2018.

7. Сущеня, А. А. Применение форматов электронных книг при передаче конфеденциальной информации методами компьютерной стеганографии / А. А. Сущеня // 83-я научно-техническая конференция профессорско-преподавательского состава, научных сотрудников и аспирантов (с международным участием), 4-15 февраля, Минск, БГТУ 2019 - С. 39-40.

8. Сущеня, А. А. Использование закодированного изображения в каскадных таблицах стилей для увеличения объема стеганографических контейнеров формата EPUB/ А. А. Сущеня // Тезисы докладов X Международной научно-технической конференции «Информационные технологии в промышленности, логистике и социальной сфере», 23-24 мая, Минск / 2019. - С. 200-203.