МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

РАБОТА ПРОВЕРЕНА

Рецензент
Ведущий программист
ООО «ВОРТЕКСКОД»

_____ П.А. Михайлов

“___”_____ 2017 г.

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой,
д.ф.-м.н., профессор

_____ Л.Б. Соколинский

“___”_____ 2017 г.

РАЗРАБОТКА ПРОГРАММЫ ДЛЯ ЗАЩИТЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ С ПОМОЩЬЮ ТЕХНОЛОГИИ ЦИФРОВЫХ ВОДЯНЫХ ЗНАКОВ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.03.02.2017.13-024-1382.ВКР

Научный руководитель
старший преподаватель кафедры СП

_____К.Ю. Никольская

Автор работы,
студент группы КЭ-401

_____А.С. Зарыпов

Ученый секретарь
(нормоконтролер)

_____ О.Н. Иванова

“___”_____ 2017 г.

Челябинск-2017

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Глоссарий

Защита программного обеспечения (ПО) – комплекс мер, направленных на защиту ПО от несанкционированного приобретения, использования, распространения, изучения, воссоздания аналогов.

Стеганография – способ передачи или хранения информации с учетом сохранения в тайне самого факта такого хранения или передачи.

Цифровая стеганография – направление стеганографии, предполагающее сокрытие или вндерение информации в цифровых объектах, например, аудиофайлах или изображениях.

Цифровой водяной знак (ЦВЗ) – специальная метка, незаметно внедряемая в изображение или другой сигнал с целью тем или иным образом контролировать его использование.

Актуальность

Задача защиты информации от несанкционированного доступа решалась во все времена на протяжении истории человечества. Два основных направления, существующих с древних времен: криптография и стеганография. Криптографические методы позволяют скрывать информацию с помощью шифрования, а стеганографические методы скрывают сам факт наличия скрытой информации.

Буквальное значение слова стеганография – тайнопись. Исторически это направление появилось первым, сам термин введен в 1499 году в трактате «Стеганография», создатель – Иоганн Тритемий. Трактат был замаскирован под магическую книгу. Тайнопись осуществляется самыми различными способами. Основным отличием стеганографии от криптографии является сокрытие факта передачи или хранения информации. Стеганографические методы обычно используют вместе с криптографическими, дополняя их.

Существует большое количество различных методов, позволяющих скрыть факт наличия зашифрованной информации. Классическими приме-

рами являются крапленые колоды, в которой карты должны быть расположены в определенном порядке для восстановления информации. Также были распространены шифры, использующие форму информационного носителя. Информация восстанавливалась с использованием трафаретов или геометрических форм. С развитием фотографии стала применяться технология микроточек – маленьких снимков, вставляемых в текст писем. Эта технология успешно применялась во время Второй мировой войны.

Во время Второй мировой войны правительством США придавалось большое значение борьбе против тайных методов передачи информации. Были введены определенные ограничения на почтовые отправления. Так, не принимались письма и телеграммы, содержащие кроссворды, ходы шахматных партий, поручения о вручении цветов с указанием времени и их вида; у пересылаемых часов переводились стрелки. Был привлечен многочисленный отряд цензоров, которые занимались даже перефразированием телеграмм без изменения их смысла.

Методы современной стеганографии можно разделить по наличию обработки цифровых сигналов. При отсутствии такой обработки возникает вопрос надежности сокрытия данных в заголовках файлов или при передаче пакетов, ведь такие данные сравнительно легко обнаружить или уничтожить. Соответственно, на применимость данных методов накладываются ограничения. С другой стороны, происходит все больше исследований в области применимости стеганографии при обработке цифровых сигналов, что позволяет говорить о цифровой стеганографии. Например, при передаче голосового сообщения с использованием IP-телефонии в некоторые пакеты могут могут встраиваться секретные данные. Стандартный приемник распознает такой пакет как поврежденный и проигнорирует его, но если принимающий знает о наличии существования таких пакетов, то он может настроить приемник так, чтобы эти пакеты не удалялись, а сохранялись, и производить расшифровку данных. Другим более распространенным примером являются цифровые данные, как правило, имеющие аналоговую

природу, например, аудиозаписи, изображения, видео. Известны также предложения по встраиванию информации в текстовые файлы и в исполняемые файлы программ.

Разработка эффективного метода защиты ПО является одной из ключевой задач при разработке коммерческого программного обеспечения. При возникновении одного или нескольких видов несанкционированного использования ПО его разработчик может понести серьезные убытки. Методы защиты ПО позволяют разработчику свести к минимуму возможность нелегального использования своей продукции. Также актуальной проблемой является ограничение на использование шифрования в некоторых странах и отсутствие единых стандартов в области защиты цифровых данных. Поэтому исследования в этой области в настоящее время являются актуальными. Одним из методов является использование цифровых водяных знаков.

Структура и объем работы

Работа состоит из введения, 4 глав, заключения и библиографии. Объем работы составляет 34 страницы, объем библиографии – 15 источников.

Содержание работы

В первой главе приводится анализ предметной области, рассматриваются методы встраивания информации в цифровые носители.

Во второй главе приводится описание и анализ требований к разрабатываемой системе.

В третьей главе приводится проектирование разрабатываемой системы.

В четвертой главе приводится реализация разрабатываемой системы.

В заключении приводятся основные результаты работы и рассматриваются дальнейшие пути развития разрабатываемой системы.

Цель и задачи работы

В данной работе исследуются основные методы сокрытия информации в исполняемых файлах, их надежность и эффективность применения для защиты программного обеспечения. Целью работы является разработка программы для защиты программного обеспечения с использованием технологии цифровых водяных знаков.

Для разработки необходимо решить следующие задачи:

1) провести анализ предметной области;
2) исследовать методы встраивания скрытой информации в цифровые контейнеры;
3) определить требования к системе встраивания цифровых водяных знаков
4) спроектировать систему;
5) реализовать систему.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. PE-файлы и их структура

PE-файл (Portable Executable) – это формат исполняемых файлов, библиотек и объектного кода, используемый в операционной системе Windows. Структура PE-файла представлена на рис. 1.

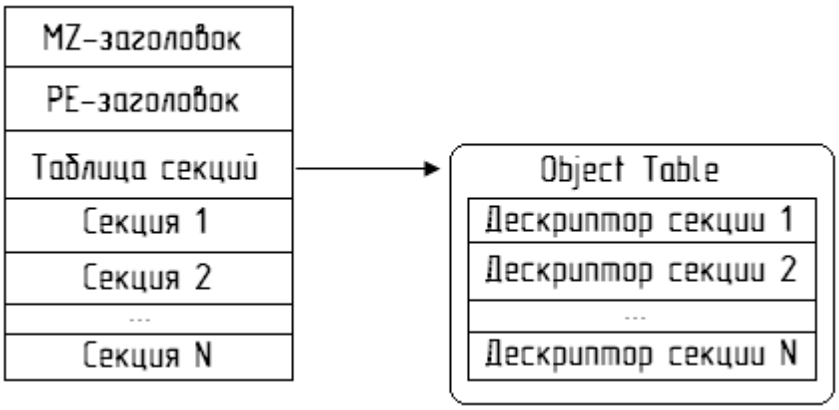

Рис. 1. Структура PE-файла

В MZ-заголовке находится специальная программа, называемая stub, она необходима для совместимости с ранними версиями ОС, и в случае запуска под ними выведет сообщение о несовместимости программы и операционной системы. PE-заголовок определяет структуру исполняемого файла, а также общие и специфические параметры файла. Он состоит из сигнатуры, которая равна 0x00004550, что при переводе в ASCII дает значение PE00, по которому файлы и получили свое название. За сигнатурой следуют основной и опциональный заголовки файла. В основном заголовке, который называется файловым, указаны наиболее общие свойства данного файла, такие как тип процессора, под который скомпилирован файл, количество заголовков секций, размер опционального заголовка и некоторую другую информацию. В опциональном заголовке хранится специфическая информация о данном файле, например, адреса точки входа в программу (относительный виртуальный адрес) и точки отображения образа файла в память. Еще в опциональном заголовке содержится важная информация о границах выравнивания секций. Выравнивание по сути своей

является дополнением до определенного числа байт и предназначено для оптимизации работы компьютера. В опциональном заголовке хранится информация о границах выравнивания двух типов: выравнивание секций в памяти и в файле на диске. Информация данного типа разделена из-за специфики работы компьютера с памятью. Чтение информации с диска происходит секторами, а из памяти страницами, и размер секторов и страниц может различаться в зависимости от операционной и файловых систем. При считывании файла размером 1 байт с диска будет считано количество байт, которое занимает 1 сектор, то есть 512 байт, где 1 байт будет значащим, а остальные – нулевыми.

PE-файл состоит из секций, каждая из которых содержит определенную информацию: код, данные, таблицу импорта. Однако возникает сложность в несоответствии секций на диске и в памяти из-за упомянутого выравнивания секций. До самих секций располагается блок, описывающий параметры каждой секции файла, называемый таблицей секций. В нем находятся не сами секции, а набор заголовков секций, или набор дескрипторов. Каждому дескриптору соответствует одна секция файла, и он описывает набор ее параметров, таких как размер, адрес на диске и в памяти. Дескрипторы имеют статическую длину в 40 байт и не подвергаются выравниванию.

После дескрипторов следуют тела секций. Согласно спецификации Microsoft определенные секции имеют фиксированные названия для того, чтобы определить назначение конкретной секции по умолчанию. Их имена начинаются с символа точка и несут смысловую нагрузку о содержимом секции, например:

1) в «.text» размещается код программы;
2) в «.bss» размещаются неинициализированные данные;
3) в «.data» размещаются инициализированные данные;
4) в «.edata» размещаются функции, экспортируемые файлом;
5) в «.idata» размещаются таблицы импортируемых функций;

6) в «.rsrc» размещаются ресурсы.

Такое именование не является единственным, различные компиляторы могут присваивать секциям различные имена. Назначение секций также не является строгим, в той же секции «.text» могут храниться ресурсы, экспортируемые функции. Более того, существует возможность программно создавать дополнительные секции со своими названиями и назначениями. Секции расположены подряд, между ними нет свободных мест, однако это не значит, что свободных мест нет внутри секций.

В конце PE-файла может присутствовать отладочная информация и дополнительные данные другого типа. Наличие такой информации необязательно.

1.2. Механизмы внедрения информации и их классификация

Существует несколько способов внедрения дополнительной информации в файлы. Классификация механизмов внедрения может осуществляться по нескольким признакам: по месту (начало, конец, середина), по способу внедрения (затирание исходных данных, внедрение в свободное пространство, переселение исходных данных на новое место обитания), по надежности, по рентабельности (рентабельное или нерентабельное) и т.д.

1.2.1. Классификация по характеру воздействия на образ файла

Рассмотрим подробнее классификацию, которая основана на характере воздействия на физический и виртуальный образ программы. Эта классификация обозначает, как изменяется адресация и размер файлов при встраивании в них информации. Согласно ей, можно выделить 4 категории механизмов:

– не вызывающие изменения адресации ни физического, ни виртуального образов;

– вызывающие изменения адресации только физического образа;

– вызывающие изменения адресации как физического, так и виртуального образов;

– не затрагивающие файл-носитель.

Для изучения механизмов внедрения информации введем следующие обозначения, соответствующие структуре PE-файла (рис. 2).

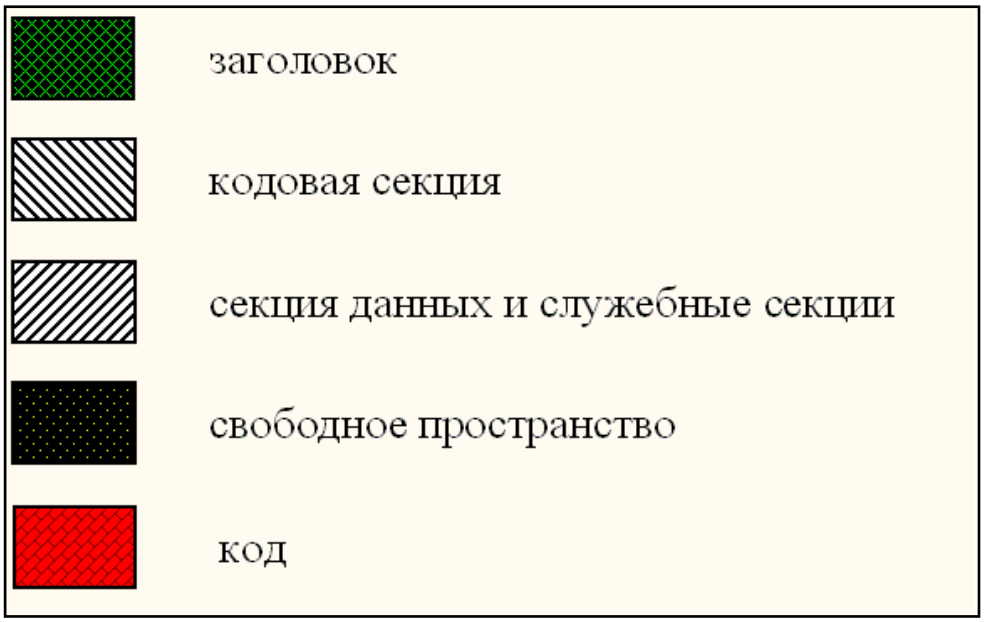

Рис. 2. Обозначения частей PE-файла

При использовании механизмов первой категории после внедрения в файл не изменяется ни его физический размер, ни количество выделенной памяти, ни адресация базовых структур. К этой категории относятся механизмы внедрения в пустое место файла (PE-заголовок, хвосты секций, регулярные последовательности), а также внедрение путем сжатия части секции.

При использовании механизмов второй категории после внедрения в файл увеличивается его физический размер, однако количество выделенной при загрузке памяти и адресация базовых структур не изменяется. В то же время изменяются их физические смещения, что с высокой степенью вероятности приведет к потере работоспособности файла-носителя. Использование таких механизмов предполагает полную или частичную перестройку структур, что затрудняет проектирование и реализацию данных методов. К этой категории относятся механизмы раздвижки заголовка, сброса части оригинального файла в оверлей и создание своего собственного оверлея.

При использовании механизмов третьей категории длина файла и выделяемая при загрузке память увеличиваются. Базовые структуры могут либо оставаться на своих местах (т.е. изменяются лишь смещения, отсчитываемые от конца образа/файла), либо перемещаться по страничному имиджу произвольным образом, требуя обязательной коррекции. К этой категории относятся механизмы расширения последней секции файла, создания своей собственной секции и расширения серединных секций.

При использовании механизмов четвертой категории файл-носитель остается неизменным, а внедрение в его адресное пространство производится косвенным путем. На основе данных механизмов работают сетевые черви и программы-шпионы.

1.2.2. Внедрение в пустое место файла

Проще всего внедриться в пустое место файла. На сегодня таких мест известно три: а) PE-заголовок; б) хвостовые части секций; в) регулярные последовательности. Рассмотрим их подробнее.

Внедрение в PE-заголовок

Типичный PE-заголовок вместе с MS-DOS заголовком занимает порядка 768 байт, а минимальная кратность выравнивания секций составляет 512 байт. Таким образом, между концом заголовка и началом первой секции практически всегда имеется 256 байт, которые можно использовать для внедрения скрытого сообщения, размещая здесь либо всю внедряемую информацию целиком, либо ей часть, если размер не позволяет большего (рис. 3).

Рис. 3. Внедрение информации в свободное пространство хвоста PE-заголовка

Для внедрения оставшейся части информации можно использовать комбинацию из нескольких методов внедрения информации, однако это усложняет процессы интеграции и восстановления цифрового водяного знака.

Перед внедрением в заголовок необходимо убедиться, что хвостовая часть заголовка действительно свободна. Сканирование таких заголовков обычно выявляет длинную цепочку нулей, расположенных в его хвосте и, очевидно, никак и никем не используемых. Туда-то и можно записать информацию, но только с предосторожностями. Необходимо отсчитать, по меньшей мере, 16 байт от последнего ненулевого символа, оставляя этот участок нетронутым (в конце некоторых структур присутствует до 16 нулей, искажение которых ни к чему хорошему не приведет). Внедрение в PE-заголовок в большинстве случаев можно распознать визуально.

Иногда за таблицей секций присутствует таблица импорта с перечнем имен загружаемых динамических библиотек. Дальше, вплоть до начала первой секции, не должно быть ничего, кроме нулей, использующихся для выравнивания. Если же это не так, то исследуемый файл содержит внедренный код.

Внедрение в хвост секции

Операционная система Windows требует, чтобы физические адреса секций были выровнены по меньшей мере на 512 байт, поэтому между секциями практически всегда есть некоторое количество свободного пространства, в которое легко можно внедрить информацию.

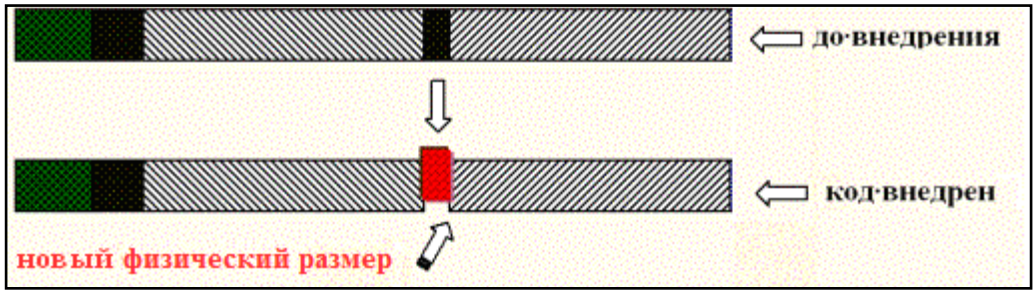

Рис. 4. Внедрение информации в хвост секции, оставшийся после выравнивания

Перед внедрением необходимо найти секцию с подходящими атрибутами и достаточным свободным пространством в конце или рассредоточить внедряемый код в нескольких секциях. Для этого необходимо просканировать хвостовую часть секции на предмет наличия непрерывной цепочки нулей – если таковая там действительно присутствует, ее можно безбоязненно использовать для внедрения.

Распознать внедрение этого типа достаточно трудно, особенно если код полностью помещается в первой кодовой секции файла, которой, как правило, является секция .text. Внедрение в секцию данных разоблачает себя наличием характерного кода в ее хвосте.

Внедрение в регулярную последовательность байт

Цепочки нулей необязательно искать в хвостах секций, также как необязательно искать именно нули, для внедрения подходит любая регулярная последовательность (например, цепочка FF FF FF... или даже FF 00 FF 00...). Если внедряемых цепочек больше одной, код придется разбить по всему телу файла. Соответственно, стартовые адреса и длины этих цепочек придется где-то хранить, для их последующего восстановления.

Регулярные последовательности чаще всего обнаруживаются в ресурсах, а точнее – в файлах .bmp и иконках. Технически внедриться сюда ничего не стоит, но пользователь тут же заметит искажение иконки, чего допускать ни в коем случае нельзя (даже если это и не главная иконка приложения, так как проводник показывает остальные по нажатию кнопки «сменить значок» в меню свойств ярлыка). Внедрение кода в регулярные цепочки продемонстрировано на рис. 5.

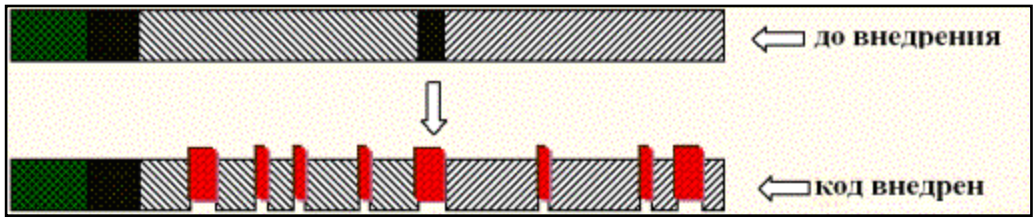

Рис. 5. Внедрение информации в регулярные цепочки

Алгоритм внедрения выглядит так: сканируем файл на предмет поиска регулярных последовательностей и отбираем среди них цепочки наибольшей длины, причем сумма их длин должна несколько превышать размеры кода, т.к. на каждую цепочку в среднем приходится 11 байт служебных данных: четыре байта на стартовую позицию, один байт – на длину, один – на оригинальное содержимое и еще пять байт на машинную команду перехода к другой цепочке.

Разбиваем код на части, добавляя в конец каждой из них команду перехода на начало следующей. Запоминаем начальные адреса, длины и исходное содержимое всех цепочек в импровизированном хранилище, сооруженном либо внутри PE-заголовка, либо внутри одной из цепочек. Если этого не сделать, потом будет невозможно извлечь код из файла. После чего записываем код в выбранные цепочки.

1.2.3. Внедрение путем сжатия части файла

Внедрение в регулярные последовательности фактически является разновидностью более общей техники внедрения в файл путем сжатия его части, в данном случае осуществляемое по алгоритму RLE. Если же использовать более совершенные алгоритмы (например, Хаффмана или LZW), то стратегия выбора подходящих частей значительно упрощается. Мы можем сжать кодовую секцию, а на освободившееся место записать наш код. Для компрессии можно использовать функционал, реализованный в самой ОС (аудио/видео-кодеки, экспортеры графических форматов, сетевые функции сжатия и т.д.).

Естественно, упаковка оригинального содержимого секции (или ее части) не обходится без проблем. Во-первых, следует убедиться, что секция вообще поддается сжатию. Во-вторых, предотвратить сжатие ресурсов, таблиц экспорта/импорта и другой служебной информации, которая может присутствовать в любой подходящей секции файла и кодовой секции, в том числе. В-третьих, перестроить таблицу перемещаемых элементов (если, конечно, она вообще есть), исключая из нее элементы, принадлежащие

сжимаемой секции и поручая настройку перемещаемых адресов непосредственно самому внедряемому коду (рис. 6).

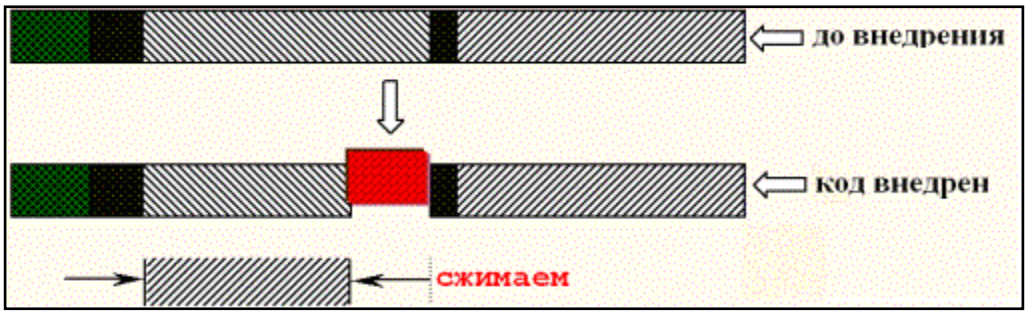

Рис. 6. Внедрение информации путем сжатия секции

Распознать факт внедрения в файл путем сжатия части секции трудно, но все-таки возможно. Дизассемблирование сжатой секции обнаруживает некоторое количество бессмысленного мусора, настораживающего опытного исследователя, но зачастую ускользающего от новичка. Также, стоит обратить внимание на размеры секций. Если виртуальные размеры большинства секций много больше физических, то файл, по всей видимости, сжат каким-либо упаковщиком.

1.2.4. Раздвижка заголовка

Когда пространства, имеющегося в PE-заголовке (или какой либо другой части файла) оказывается недостаточно для размещения всего кода целиком, мы можем попробовать растянуть заголовок на величину, выбранную по своему усмотрению. До тех пор, пока размер заголовка (SizeOfHeaders) не превышает физического смещения первой секции, такая операция осуществляется элементарно, но вот дальше начинаются проблемы, для решения которых приходится кардинально перестраивать структуру исполняемого файла. Как минимум, необходимо увеличить физические адреса начала всех секций на величину, кратную принятой степени выравнивания, прописанной в поле «Физическое выравнивание секций» и физически переместить хвост файла, записав код на освободившееся место.

Максимальный размер заголовка равен виртуальному адресу первой секции, т.к. заголовок не может перекрываться с содержимым страничного

имиджа. Учитывая, что минимальный виртуальный адрес составляет 4096 байт, а типичный размер заголовка – 768 байт, мы получаем в свое распоряжение порядка 3 Кбайт свободного пространства, достаточного для размещения практически любого кода (рис. 7).

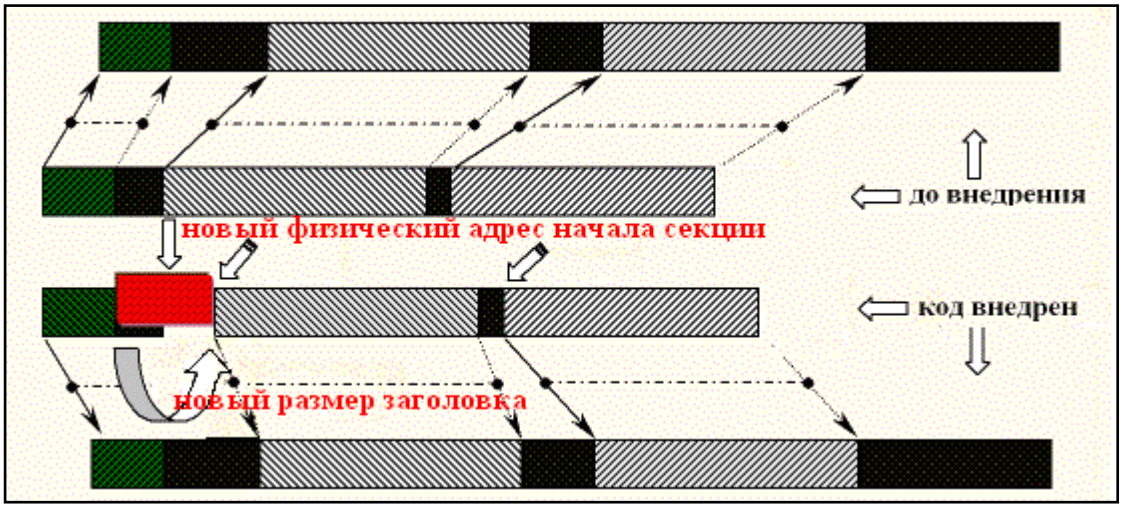

Рис. 7. Исполняемый файл и его проекция в память до и после внедрения информации путем раздвижки заголовка

Данный метод внедрения распознается аналогично обычному методу внедрения в PE-заголовок.

1.3. Обзор аналогов

Hydan – программа для создания цифровых водяных знаков на основе правил записи машинных команд. Эта программа для создания ЦВЗ использует такую особенность машинных команд, как возможность описания несколькими способами. Например, для того, чтобы увеличить значение регистра eax, можно воспользоваться разными командами, например, «add %eax, $5» или «sub %eax, $-5.» Некоторые машинные команды могут иметь две формы записи: «insn r/m, reg» и «insn reg, r/m». Если полагать, что указатель r/m является указателем только на регистр, то можно пользоваться и той, и другой формой записи, в одном из случаев поменяв порядок параметров в вызове машинной команды.

Некоторые машинные команды имеют до 7 альтернативных форм записи. Этого достаточно, чтобы закодировать 2 бита, но недостаточно для

3 бит. Для того, чтобы избежать игнорирования 3 возможных способов записей инструкций, предлагается сделать одну из них маркером замены. Она будет использоваться тогда, когда значение не может быть представлено в текущем наборе машинных команд напрямую. Вместо этого такие значения будут разбиваться на более мелкие части, которые и будут использоваться для создания цифрового водяного знака. Тогда при использовании набора из 7 инструкций можно будет вставить $\log_2(7-1)$ бит информации, или приблизительно 2,58 бита вместо 2 бит. Данный метод предполагает, что в кодируемом сообщении выполняется условие равномерного распределения команд.

При создании цифрового водяного знака программа использует равнозначную замену машинных команд. Одна инструкция заменяется ровно на одну инструкцию. Это сделано для избегания сдвига последовательности вызовов функций и ссылок на секции данных, т.е. алгоритм относится к категории не изменяющих изменения адресации физического и виртуального образов программы. Замена одной команды несколькими возможна, но она значительно усложняет процесс интеграции и восстановления цифрового водяного знака.

При интеграции цифрового водяного знака программа предварительно шифрует и интегрирует в контейнер длину цифрового водяного знака. Для шифрования используется алгоритм Blowfish в режиме CBC (Cipher Block Chaining) и требуется введение пользовательского ключа шифрования. Интеграция длины цифрового водяного знака необходима для его восстановления, а шифрование длины необходимо для сокрытия факта встраивания цифрового водяного знака и предотвращения применения криптоанализа последовательности данных. Для вставки используется метод случайного выбора машинных команд, который так же зависит от пользовательского ключа. Метод используется для повышения криптостойкости программы. Способ заключается в генерации случайного числа пропускаемых бит в промежутке, зависящего от длины оставшейся части,

в которую интегрируется цифровой водяной знак, а также от длины самого цифрового водяного знака. Промежуток, на котором генерируется число пропускаемых бит, обновляется каждый раз после того, как был встроен 1 байт информации.

1.4. Вывод

Проведенный обзор методов встраивания цифровых водяных знаков позволил выявить, что использование методов, не вызывающих изменения адресации физических и виртуальных образов, является наиболее подходящим. Эти методы обеспечивают достаточную скрытность встраиваемой информации при относительной легкости реализации системы.

Проведенный обзор существующих аналогов показал, что разработка программы встраивания ЦВЗ для защиты программных продуктов является актуальной задачей.

2. ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

Система встраивания ЦВЗ в программный продукт предназначена для предотвращения несанкционированного использования программного продукта.

2.1. Функциональные требования

Разрабатываемая система встраивания ЦВЗ в программный продукт должна удовлетворять следующим функциональным требованиям:

1) система встраивания ЦВЗ в программный продукт должна предоставлять возможность создания цифрового водяного знака на основе некоторого файла;

2) система встраивания ЦВЗ в программный продукт должна предоставлять возможность интегрирования цифрового водяного знака в некоторый файл-контейнер;

3) система встраивания ЦВЗ в программный продукт должна предоставлять возможность детекции наличия цифрового водяного знака в некотором файле;

4) система встраивания ЦВЗ в программный продукт должна предоставлять возможность восстановления цифрового водяного знака, извлекаемого из некоторого файла.

2.2. Нефункциональные требования

Согласно принципу Керкгоффса, при разработке криптографических систем в засекреченном виде должен храниться только определенный набор параметров алгоритма шифрования, называемый ключом. Однако, если использовать стеганографические методы, не используя этот принцип, скрываемая информация может быть легко обнаружена. Использование стеганографических методов должно гарантировать защиту информации даже в случае, когда злоумышленник знает о факте сокрытия информации и, возможно, об используемых методах.

Разрабатываемая система встраивания ЦВЗ в программный продукт должна удовлетворять следующим нефункциональным требованиям:

1) разрабатываемая система должна соответствовать принципу Керкгоффса;

2) для обеспечения соответствия используется модифицированный алгоритм шифрования AES.

2.3. Варианты использования системы

В ходе выполнения выпускной квалификационной работы была разработана диаграмма вариантов использования системы встраивания ЦВЗ в программный продукт, она представлена на рис. 8. Диаграмма включает в себя одного основного актера, взаимодействующего с системой.

Менеджер – это актер, который использует систему защиты ПО. Он может являться как человеком, так и другой программой, которая использует возможности разрабатываемой системы.

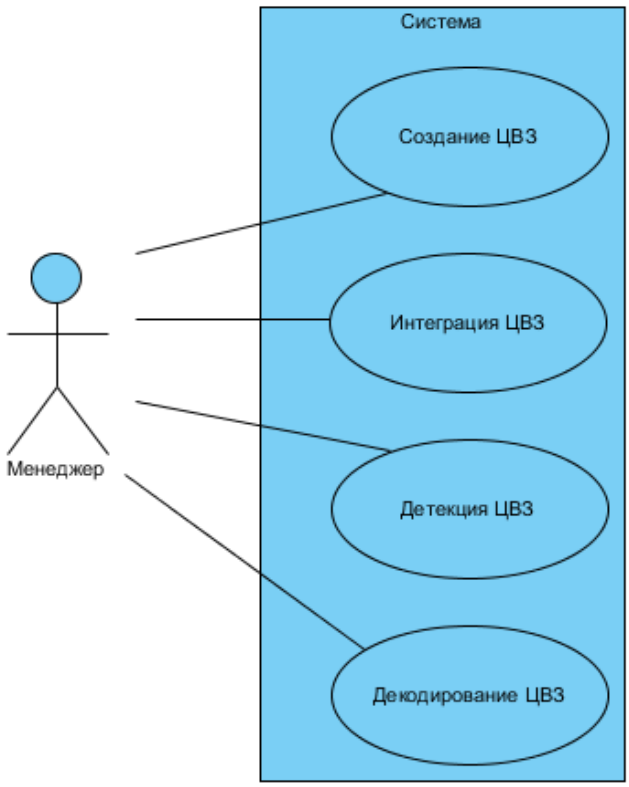

Рис. 8. Диаграмма вариантов использования

Можно определить следующие основные варианты использования разрабатываемой системы.

Вариант использования «Создание ЦВЗ». Менеджер может создать цифровой водяной знак на основе некоторого файла.

Вариант использования «Интеграция ЦВЗ». Менеджер может интегрировать цифровой водяной знак в некоторый файл-контейнер.

Вариант использования «Детекция ЦВЗ». Менеджер может узнать, содержится ли в некотором файле цифровой водяной знак.

Вариант использования «Декодирование ЦВЗ». Менеджер может получить исходный цифровой водяной знак из файла-контейнера.

2.4. Вывод

В процессе анализа требований были сформулированы основные функциональные и нефункциональные требования, предъявляемые к разрабатываемой системе, определены основные актеры и варианты использования системы.

3. ПРОЕКТИРОВАНИЕ

3.1. Компоненты системы

На рис. 9 представлена диаграмма компонентов системы встраивания ЦВЗ в программный продукт.

Диаграмма показывает разбиение системы на структурные компоненты и зависимости между ними. Система включает 5 основных компонентов:

1) Обработчик запросов – компонент, отвечающий за связь между менеджером и другими компонентами системы. Компонент обрабатывает запросы и передает управление соответствующему компоненту системы.

2) Стегокодер – компонент, отвечающий за создание цифрового водяного знака на основе некоторого файла.

3) Стегодетектор – компонент, отвечающий за детекцию цифрового водяного знака в некотором файле.

4) Обработчик файлов – компонент, отвечающий за обработку файлов.

5) AES – компонент, реализующий модифицированный алгоритм шифрования AES-CTR.

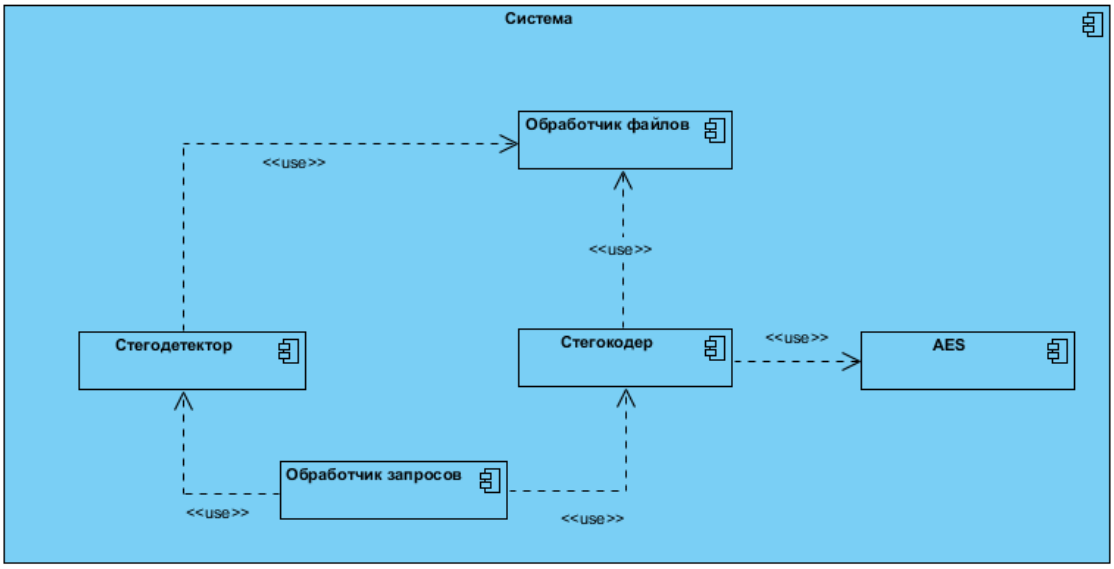

Рис. 9. Диаграмма компонентов системы

3.2. Диаграммы последовательности

На основе диаграммы компонентов была разработана диаграмма последовательности создания цифрового водяного знака (рис. 10). На диаграмме в общем виде представлена последовательность действий, необходимых для создания цифрового водяного знака.

При создании цифрового водяного знака не производится шифрование, т.к. полученный цифровой знак может быть использован и в других целях. Шифрование производится при интеграции цифрового водяного знака.

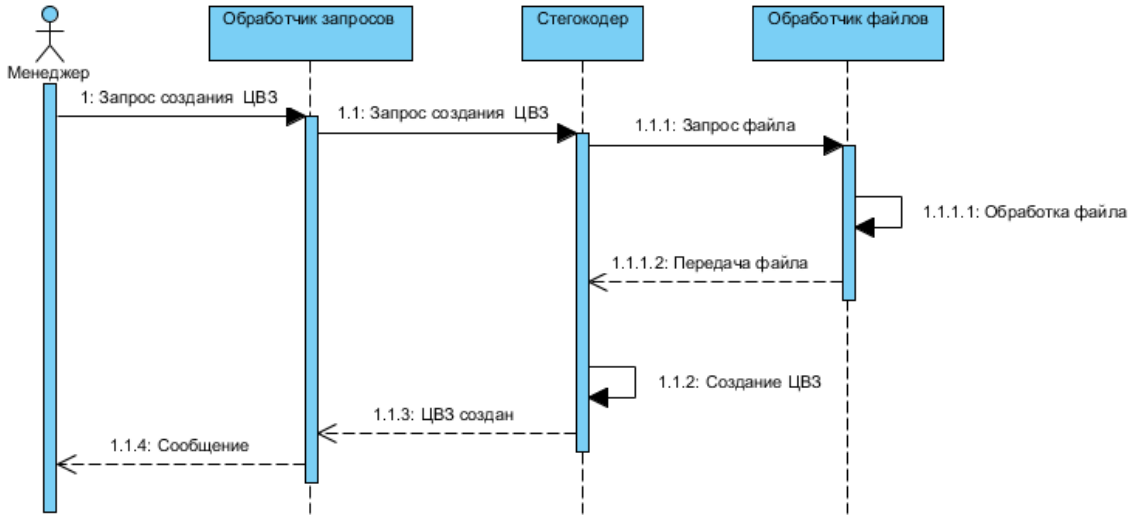

Рис. 10. Диаграмма последовательности создания ЦВЗ

На основе диаграммы компонентов была разработана диаграмма последовательности интеграции цифрового водяного знака (рис. 11). На диаграмме в общем виде представлена последовательность действий, необходимых для создания цифрового водяного знака.

При интеграции цифрового водяного знака обработчик запросов посылает запрос стегокодеру на интеграцию. Стегокодер запрашивает у обработчика файлов информацию о файле, в который будет производиться интеграция. После получения информации, если файл является валидным для интеграции, производится шифрование цифрового водяного знака. Затем стегокодер запрашивает у обработчика файлов интеграцию зашифро-

ванного цифрового водяного знака. В случае, когда файл не является валидным для встраивания цифрового водяного знака, процесс интеграции будет отменен, а менеджеру вернется соответствующее сообщение.

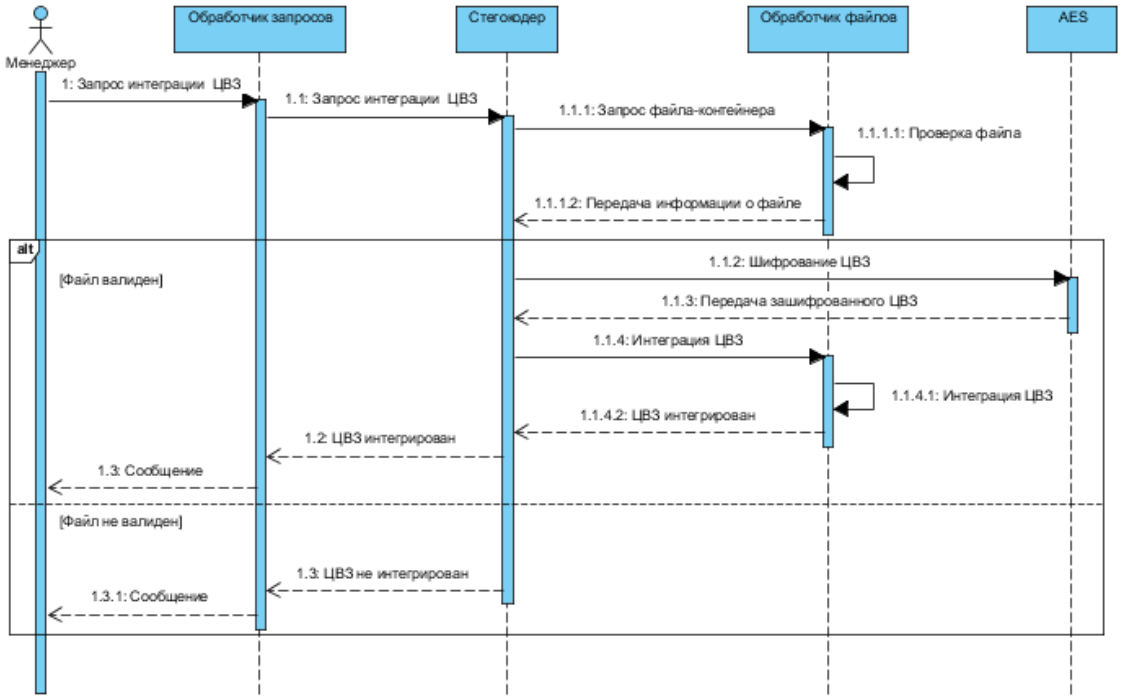

Рис. 11. Диаграмма последовательности интеграции ЦВЗ

На основе диаграммы компонентов была разработана диаграмма последовательности детекции цифрового водяного знака (рис. 12). На диаграмме в общем виде представлена последовательность действий, необходимых для детекции цифрового водяного знака.

При детекции цифрового водяного знака обработчик запросов посылает запрос стегодетектору на поиск цифрового водяного знака. Стегодетектор запрашивает у обработчика файлов данные о файле, в котором будет производиться поиск. Обработчик файлов обрабатывает файл и передает его стегодетектору для поиска. В независимости от результата поиска стегодетектор передает результат поиска обработчику запросов. При обнаружении цифрового водяного знака будет передано соответствующее сообщение об успешном обнаружении, в случае отсутствия цифрового водяного знака – сообщение об его отсутствии.

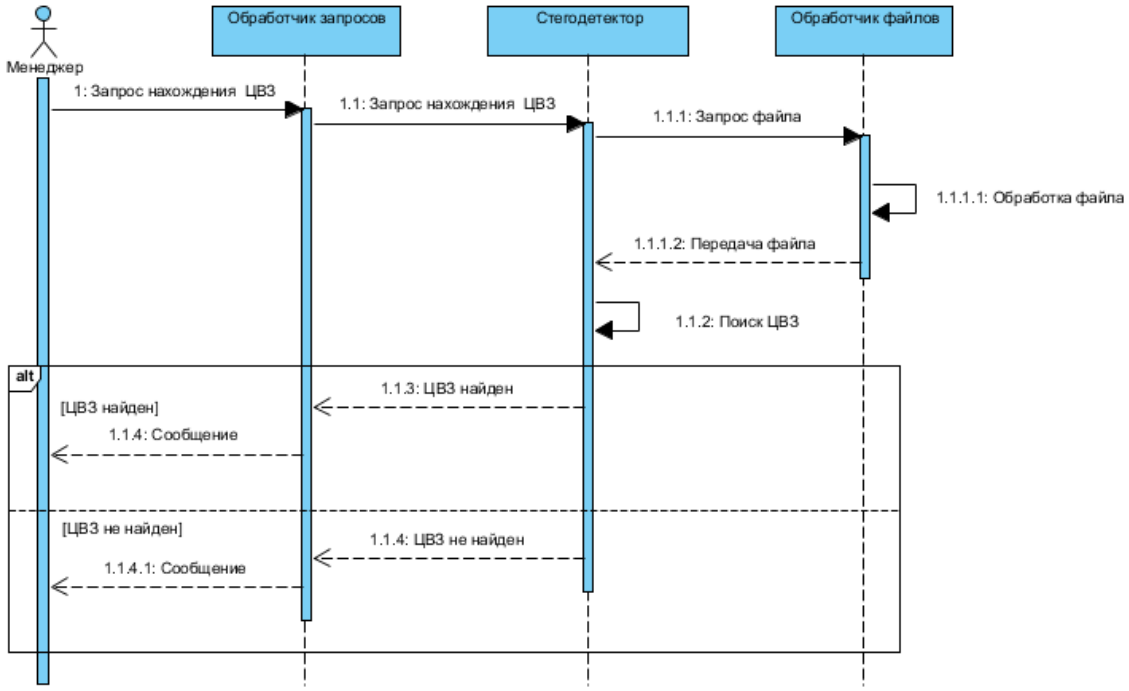

Рис. 12. Диаграмма последовательности детекции ЦВЗ

На основе диаграммы компонентов была разработана диаграмма последовательности декодирования цифрового водяного знака (рис. 13). На диаграмме в общем виде представлена последовательность действий, необходимых для декодирования цифрового водяного знака.

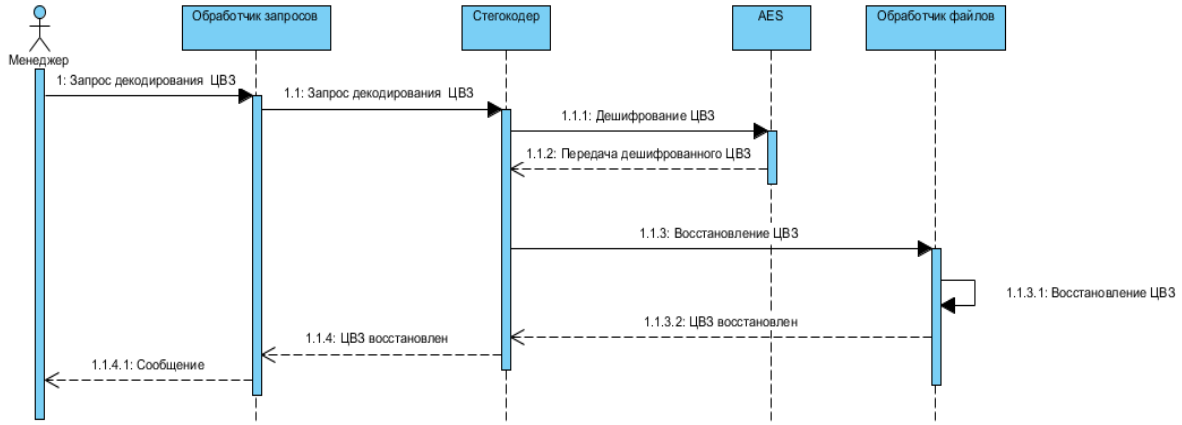

Рис. 13. Диаграмма последовательности декодирования ЦВЗ

При декодировании цифрового водяного знака обработчик запросов посылает запрос стегокодеру на декодирование цифрового водяного знака.

Стегокодер обращается к модулю AES для дешифрования скрытого блока встраиваемой информации. После этого управление передается обработчику файлов, который восстанавливает исходный цифровой водяной знак.

3.3. Вывод

В соответствии с требованиями была спроектирована архитектура системы встраивания ЦВЗ в программный продукт. На основе диаграммы компонентов и функциональных требований была подробно рассмотрена реализация отдельных вариантов использования.

4. РЕАЛИЗАЦИЯ

4.1. Средства разработки

Для реализации системы встраивания ЦВЗ в программный продукт был выбран язык программирования C++, среда разработки - Visual Studio 2013. Выбор средств программной разработки был произведен по следующим критериям. Модуль алгоритма шифрования AES уже был разработан и протестирован. Среда разработки поддерживает функционал, который обеспечивает обратную совместимость с разработками, проведенными ранее.

4.2. Разработка программы

Интерфейс программы представлен на рис. 14.

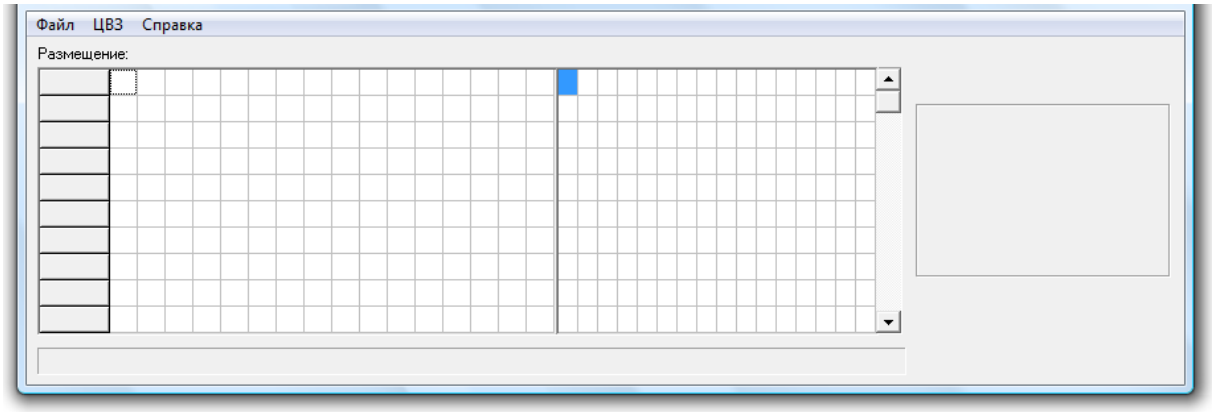

Рис. 14. Система встраивания ЦВЗ

Основные компоненты интерфейса программы:

1) таблица просмотра данных о файле-контейнере;

2) область предпросмотра файла, из которого будет получен цифровой водяной знак.

При загрузке файла-контейнера в таблице просмотра данных появится шестнадцатиричное и ASCII-представление файла-контейнера. При выборе файла, на основе которого будет строиться цифровой водяной знак, в окне предпросмотра появится его изображение. Элементы файла-контейнера, куда будет производиться встраивание, будут выделены.

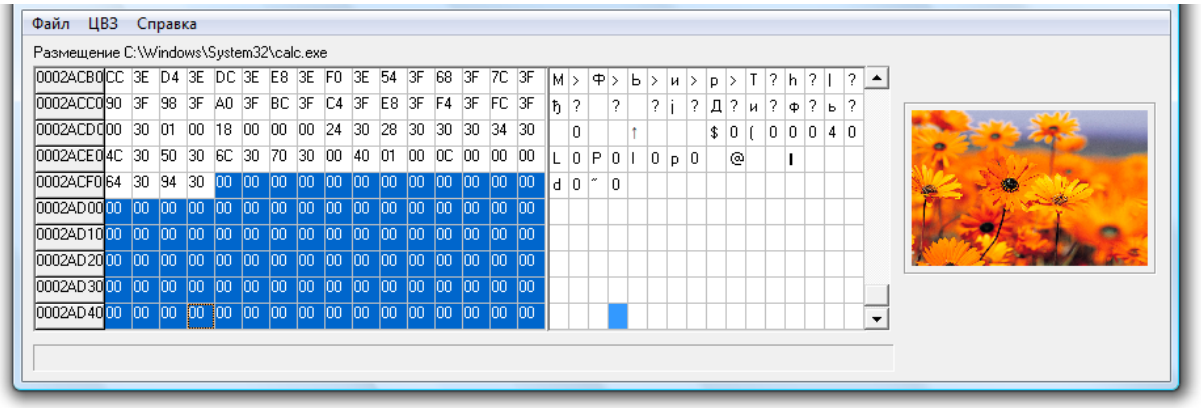

Рис. 15. Состояние после выбора файла для создания ЦВЗ

При загрузке файла-контейнера в таблице просмотра данных появится шестнадцатиричное и ASCII-представление файла-контейнера. При выборе файла, на основе которого будет строиться цифровой водяной знак, в окне предпросмотра появится его изображение. Элементы файла-контейнера, куда будет производиться встраивание, будут выделены.

При создании ЦВЗ из файла, на основе которого создается цифровой водяной знак, последовательно считываются байты до тех пор, пока это возможно. Производится преобразование типа и добавление элемента файла в цифровой водяной знак. Листинг процесса создания представлен на рис. 16.

```
while (!input_file.eof())
{
    j++;
    for (int i = 1; i <= ColCount && !input_file.eof(); i++)
    {
        letter2 = letter = input_file.get();
        if (letter != -1)
        {
            AnsiString hex = IntToHex(letter, 2);
            Cells[i][j] = hex;
            Cells[i - 1][j] = letter2;
        }
        else
        {
            Cells[i][j] = IntToHex('\0', 2);
            Cells[i - 1][j] = '\0';
        }
    }
}
input_file.close();
```

Рис. 16. Создание ЦВЗ на основе файла-контейнера

Для интеграции ЦВЗ необходимо будет ввести пароль – ключ, необходимый для шифрования цифрового водяного знака с помощью модифицированного алгоритма шифрования AES. Если цифровой водяной знак будет пригоден к встраиванию в файл-контейнер, то появится сообщение об успехе интеграции, иначе – о провале интеграции.

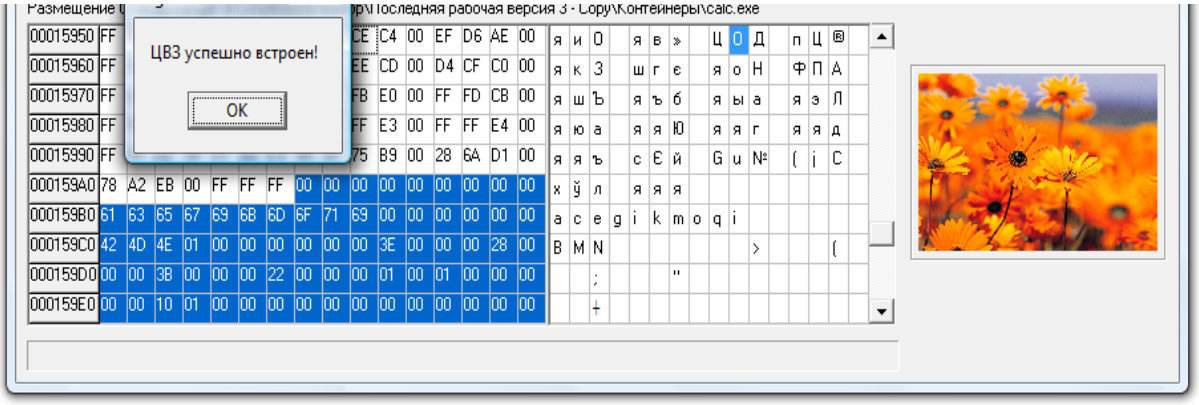

Рис. 17. Состояние после встраивания ЦВЗ

При извлечении ЦВЗ из файла происходит дешифрование последовательности байт, полученной при детекции ЦВЗ в файле. Затем полученная последовательность байт восстанавливается в виде файла, который показывается в области предпросмотра. Листинг процесса извлечения представлен на рис. 18.

```
if (decrypted)
{
    ofstream output_file(out_file, ios::binary);
    for (int j = 0; j < SRowCount; j++)
    {
        for (int i = 0; i < ColCount; i++)
        {
            char *b = Cells[i][j].c_str();
            output_file.put(*b);
        }
    }
    output_file.close();
    DeleteFile(out_file);
    message = "ЦВЗ успешно извлечён!"
}
else
    message = "ЦВЗ не извлечен!"
return message;
```

Рис. 18. Восстановление ЦВЗ

4.3. Вывод

На основе разработанной архитектуры была реализована система встраивания ЦВЗ в программные продукты. Разработанная система соответствует предложенным требованиям, а также основным параметрам проектирования стегосистем.

ЗАКЛЮЧЕНИЕ

В данной работе была разработана система встраивания ЦВЗ в программный продукт.

На основе требований были выделены основные компоненты системы, установлены связи между ними. На основе диаграмм вариантов использования и последовательности были рассмотрены методы реализации отдельных компонентов системы.

Основные результаты

1. Проведен анализ предметной области.
2. Изучены методы встраивания информации в исполняемые файлы.
3. Выявлены основные требования к разрабатываемой системе встраивания ЦВЗ в программные продукты.
4. Проведено проектирование системы.
5. Выполнена реализация системы.

Направления дальнейших исследований

Дальнейшие исследования и практические разработки могут быть направлены на расширение функционала системы. Перспективными разработками являются добавление большего числа методов встраивания ЦВЗ в контейнер и большего числа методов шифрования цифровых водяных знаков.

ЛИТЕРАТУРА

1. AES-Based Authenticated Encryption Modes in Parallel High-Perfomance Software, 2014. [Электронный ресурс] URL: http://eprint.iacr.org/2014/186.pdf (дата обращения: 10.03.2017).

2. Daemen J., Rijmen V. AES Proposal: Rijndael, 1999. [Электронный ресурс] URL: http://www.eng.tau.ac.il/~yash/crypto-netsec/Rijndael.pdf (дата обращения: 15.04.2017).

3. Lu, C.-S. Multimedia security: Steganography and digital watermarking techniques for protection of intellectual property – Hershey: Idea Group Publishing, 2005. – 255 p.

4. Microsoft Portable Executable and Common Object File Format Specification, 2017. [Электронный ресурс] URL: http://www.microsoft.com/whdc/system/platform/firmware/PECOFF.mspx (дата обращения: 21.03.2017).

5. National Policy on the Use of the Advanced Encryption Standard (AES). [Электронный ресурс] URL: http://cryptome.org/aes-natsec.htm (дата обращения: 19.02.2017).

6. Parallel AES Encryption Engines for Many-Core Processors Arrays, 2014. [Электронный ресурс] URL: http://www.rroij.com/open-access/parallel-aes-encryption-engines-for-manycoreprocessor-arrays.pdf (дата обращения: 25.02.2017).

7. Pietrek M. An In-Depth Look into the Win32 Portable Executable File Format, 2002. [Электронный ресурс] URL: https://msdn.microsoft.com/ru-ru/magazine/bb985992(en-us).aspx (дата обращения: 28.03.2017).

8. Pietrek M. An In-Depth Look into the Win32 Portable Executable File Format, Part 2, 2002. [Электронный ресурс] URL: https://msdn.microsoft.com/ru-ru/magazine/bb985994(en-us).aspx (дата обращения: 28.03.2017).

9. Recommendation for Block Chiper Mods of Operation, 2001. [Электронный ресурс] URL: http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf (дата обращения: 12.03.2017).

10. Shin D., Kim Y., Byun K., Lee S. Data Hiding in Windows Executable Files, 2008. [Электронный ресурс] URL: http://ro.ecu.edu.au/cgi/viewcontent.cgi?article=1050&context=adf (дата обращения: 24.04.2017).

11. Specification for the Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197, 2001. [Электронный ресурс] URL: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf (дата обращения: 18.02.2017).

12. Аграновский А.В., Балакин А.В., Грибунин В.Г., Сапожников С.А. Стеганография, цифровые водяные знаки и стеганоанализ. – М.: Вузовская книга, 2009. – 220 с.

13. Зарыпов А. Распараллеливание алгоритма шифрования AES с использованием функциональной парадигмы программирования. – Челябинск: ЮУрГУ, 2016. – 25 с.

14. Конахович Г.Ф., Пузыренко А.Ю. Компьютерная стеганография. Теория и практика – М.: МК-Пресс, 2006. – 288 с.

15. Pietrek M. Peering Inside the PE: A Tour of the Win32 Portable Executable File Format. 1994. [Электронный ресурс] URL: https://msdn.microsoft.com/en-us/library/ms809762.aspx (дата обращения: 28.03.2017).