Михаил В. Малахов[1], Павел В. Слипенчук[2]
[1,2]Московский государственный технический университет им. Н.Э. Баумана,
2-я Бауманская ул., 5, стр. 1, Москва, 105005, Россия
[1]e-mail: misha.malaxow@yandex.ru, https://orcid.org/0000-0001-5460-7332
[2]e-mail: pavelslipenchuk@stego.su, https://orcid.org/0000-0003-0308-250X

СПОСОБ СТЕГАНОГРАФИЧЕСКОГО ХРАНЕНИЯ ИНФОРМАЦИИ В ФАЙЛОВЫХ СИСТЕМАХ ИЗВЛЕКАЕМЫХ ЭЛЕКТРОННЫХ НОСИТЕЛЕЙ

DOI: http://dx.doi.org/10.26583/bit.2020.1.08

Аннотация. Задачи скрытого хранения и передачи информации успешно решаются посредством использования стеганографии. Наиболее распространенными стеганографическими маскираторами для передачи и хранения информации в настоящий момент времени являются медиафайлы, такие как фотографии, видеозаписи и аудиофайлы. Использование таких маскираторов накладывает ряд ограничений, одним из которых является ограничение на размер вкрапляемой информации. Так, увеличение объёма скрываемых данных меняет энтропийные свойства маскиратора и позволяет обнаружить факт стеганографии. Одним из альтернативных способов обеспечения возможности стеганографического сокрытия информации больших объёмов является использование принципиально иного метода сокрытия. В связи с этим становится актуальной разработка способа стеганографического хранения информации в файловой системе, как маскиратора, в котором можно скрытно хранить достаточно большие объёмы данных. В настоящей работе представлен подход к стеганографическому хранению информации в файловой системе извлекаемого носителя информации, в качестве которого обычно выступает электронный носитель информации с файловой системой FAT32. При проведении анализа уже существующих инструментов стеганографии, предназначенных для сокрытия информации в файловых системах, были выявлены их слабые и сильные стороны, на основе которых были выдвинуты требования к разрабатываемому инструменту. В статье представлено детальное описание алгоритмов, использующихся в новом подходе, приведены его основные достоинства и описаны ситуации, при которых его применение обосновано, а также доказана его совершенность по Кашену.
Ключевые слова: защита информации, безопасность данных, стеганография, криптография, файловые системы, совершенность стеганографических систем.
Для цитирования: МАЛАХОВ, Михаил В.; СЛИПЕНЧУК, Павел В. СПОСОБ СТЕГАНОГРАФИЧЕСКОГО ХРАНЕНИЯ ИНФОРМАЦИИ В ФАЙЛОВЫХ СИСТЕМАХ ИЗВЛЕКАЕМЫХ ЭЛЕКТРОННЫХ НОСИТЕЛЕЙ. Безопасность информационных технологий, [S.l.], v. 27, n. 1, p.98-110, 2020. ISSN 2074-7136. Доступно на: <https://bit.mephi.ru/index.php/bit/article/view/1255>. Дата доступа: 10 feb. 2020. DOI: http://dx.doi.org/10.26583/bit.2020.1.08.

Mihail V. Malahov[1], Pavel V. Slipenchuk[2]
[1,2]Bauman Moscow State Technical University,
ul. Baumanskaya 2-ya, 5/1, Moscow, 105005, Russia
[1]e-mail: misha.malaxow@yandex.ru, https://orcid.org/0000-0001-5460-7332
[2]e-mail: pavelslipenchuk@stego.su, https://orcid.org/0000-0003-0308-250X

Approach to steganography storage in removable media file system

DOI: http://dx.doi.org/10.26583/bit.2020.1.08

Abstract. Problems of hidden storage and transmission of information are successfully solved by using steganography. The most common steganographic maskers for transmitting and storing information at a given time are media files, such as photos, videos, and audio files. The use of such maskers imposes a number of restrictions, one of which is a limit on the size of the embedded information. Thus, increasing the amount of hidden data changes the entropy properties of the masker and allows you to detect the fact of steganography. One of the alternative ways to ensure the possibility of steganographic concealment of large amounts of information is to use a fundamentally different method of concealment. In this regard, it

becomes relevant to develop a method for steganographic storage of information in the file system, as a masker, in which you can secretly store quite large amounts of data. This paper presents an approach to steganographic storage of information in the file system of the extracted data carrier, which is usually an electronic data carrier with the FAT32 file system. During the analysis of existing steganography tools designed to hide information in file systems, their weaknesses and strengths were identified, on the basis of which the requirements for the developed tool were put forward. The paper presents a detailed description of the algorithms used in the new approach. Its main advantages and the situations in which its application is justified are described, as well as its perfection by the Cachin method is proved.
Keywords: information security, data security, steganography, cryptography, file systems, perfectly secure steganography systems.
For citation: MALAHOV, Mihail V.; SLIPENCHUK, Pavel V. Approach to steganography storage in removable media file system. IT Security (Russia), [S.l.], v. 27, n. 1, p.98-110, 2020. ISSN 2074-7136. Available at: <https://bit.mephi.ru/index.php/bit/article/view/1255>. Date accessed: 10 feb. 2020. DOI: http://dx.doi.org/10.26583/bit.2020.1.08.

Введение

Современные криптографические алгоритмы являются достаточно стойкими, чтобы можно было говорить о невозможности их взлома за разумное время [1–4]. Недостатком криптографии, как метода безопасного хранения и передачи информации, является то, что противнику известен сам факт попытки сокрытия хранимой или передаваемой информации. Этот недостаток является существенным в случае, если противоборствующая сторона готова применить меры противоправного воздействия к субъекту, осуществляющего передачу информации.

Также в некоторых странах существует законодательная база, позволяющая правоохранительным органам изымать средства хранения и передачи информации без постановления суда, таким образом безосновательно вторгаясь в частную жизнь человека. Например, в США при пересечении границы смартфон и ноутбук любого человека могут забрать на экспертизу и запросить пароль для расшифровки данных [5], а в Великобритании при отказе лица предоставить пароль к зашифрованным данным, указанному лицу грозит лишение свободы сроком до пяти лет [6].

Однако при применении стеганографии осуществление подобного вторжения в частную жизнь человека затруднено, так как стеганографические методы позволяют скрывать несколько сообщений в одном маскираторе. Данный факт позволяет скрыть конфиденциальную или личную информацию, показав не представляющие ценность данные.

Перед сокрытием информации, её целесообразно сжать; закодировать кодами, исправляющими ошибки и зашифровать потоковым криптографическим шифром (рис. 1).

Рис. 1. Последовательность действий при сокрытии информации
(Fig. 1. Sequence of actions when hiding information)

В этом случае, если шифр потоковый (т.е. шифртекст получается наложением определённой гаммы), то возникающие ошибки могут быть исправлены корректирующим кодом. Таким образом, посредством приведенной последовательности действий

удовлетворяются требования по стеганографической стойкости и скрытности [7], формальное доказательство которых будет приведено в следующих разделах.

Процесс извлечения данных является обратным к процессу их сокрытия, он представлен на рис. 2. Сначала данные извлекаются с помощью стеганографического алгоритма, потом расшифровываются, затем подаются на декодер корректирующего кода, а в конце происходит декомпрессия информации.

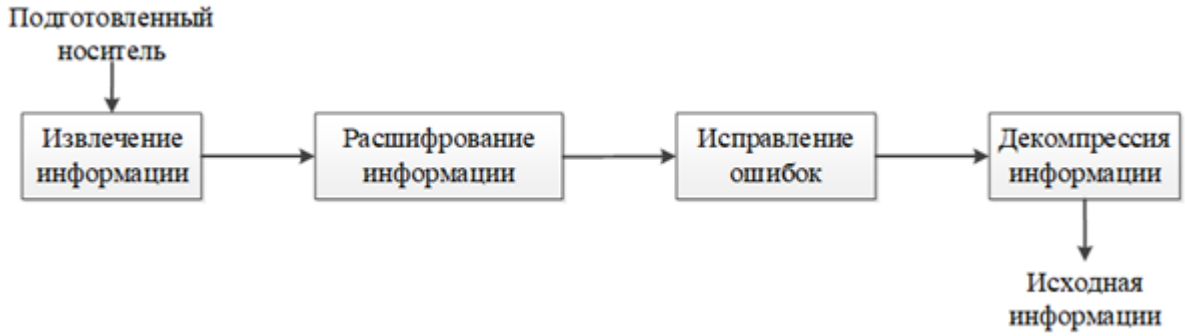

Рис. 2. Последовательность действий при извлечении данных
(Fig. 2. Sequence of actions when extracting data)

В дальнейшем будет дано описание операций встраивания и извлечения данных, а также будет охарактеризован ряд определенных действий, связанных с процессами исправления ошибок и шифрования, обусловленных особенностями работы с файловой системой извлекаемого носителя информации.

В настоящей работе будет рассматриваться сокрытие информации в файловой системе (ФС) FAT32, так как она чаще всего применяется на извлекаемых электронных носителях информации, но следует отметить, что предложенные алгоритмы можно адаптировать и для работы с другими блочными файловыми системами.

В следующем разделе будут рассмотрены существующие решения по сокрытию информации в файловых системах.

1. Существующие решения в области стеганографии с использованием маскиратора в виде файловой системы

Самая ранняя работа, посвященная данному вопросу, по всей видимости, опубликована в [8]. В [8] предложены два различных подхода.

В первом подходе для сокрытия информации используются файлы, уже хранящиеся в ФС. Запись информации осуществляется выбором n файлов из всех файлов, представленных в ФС и применения к ним операции «ИСКЛЮЧАЮЩЕЕ ИЛИ», ключом являются идентификаторы выбранных файлов. Достоинством данного подхода является возможность реализации многоуровневой системы доступа к сокрытым файлам. К его недостаткам относятся уязвимость перед атаками по открытому тексту и вычислительная сложность подбора необходимых файлов для сокрытия информации.

Второй подход заключается в заполнении ФС случайными битами, с последующей записью частей файла в случайные места файловой системы.

Этот подход получил развитие и реализацию в файловой системе StegoFS [9]. Она была разработана на базе файловой системы ext2. Реализация стеганографических функций осуществлялась за счет специального драйвера. К недостаткам данной разработки можно отнести тот факт, что работа осуществляется только на машинах с операционной системой Linux с заранее установленными

специальными драйверами. Также недостатком является изменение базовой ФС, что привлечет дополнительное нежелательное внимание к электронному носителю информации с такой файловой системой.

Попытки сокрытия данных в ФС FAT32 предпринимались в основном с помощью изменений таблицы FAT [10–12]. В [10] сообщение представляется в виде последовательности натуральных чисел n_i. Для сокрытия сообщения выбирается файл, занимающий m кластеров, номер первого кластера такого файла выбирается произвольно, все последующие номера кластеров получаются путем добавления к номеру текущего кластера, соответствующего n_i.

В [11] сообщение скрывалось посредством расположения кластеров нескольких заранее определенных файлах через определенные промежутки, зависящие от содержания скрываемого сообщения.

В [12] в основу был положен тот же подход, но была увеличена максимальная емкость маскиратора за счет сокрытия части информации при помощи несортированного порядка следования кластеров заранее определенных файлов.

Недостатками данных методов являются сильная дефрагментация диска при их применении и существенные ограничения на размер скрываемой информации.

Способ скрытого хранения информации, предлагаемый в настоящей работе, рассчитан на работу с современной файловой системой, для его применения не нужно модифицировать принцип работы файловой системы, а также с его помощью можно скрытно хранить большие объёмы информации, что выгодно отличает предложенный маскиратор в виде ФС от широко известных на сегодняшний день маскираторов в виде медиафайлов. Все вышеперечисленные факторы являются существенными достоинствами предложенного подхода по сравнению с аналогами и основными требованиями к нему. В дальнейших разделах приводится описание разработанного решения.

2. Основные сведения о файловой системе FAT32, используемые для создания маскиратора на ее основе

Документом, наиболее полно описывающим структуру и функционирование ФС FAT32, является официальное руководство от компании Microsoft [13].

Файловая система FAT состоит из четырех основных областей, приведенных в табл. 1.

Таблица 1. Области файловой системы FAT

Наименование области	Назначение области
Зарезервированная область (Reserved Region)	Содержит служебные структуры, предназначенные для драйверов операционных систем. Эти структуры в основном используются при инициализации тома.
FAT область (File Allocation Table Region)	FAT область или область таблицы FAT содержит массив указателей, указывающих на кластеры ФС и содержащих информацию о состоянии этих кластеров. Обычно на диске хранится несколько копий таблиц FAT.
Область корневой директории (Root Directory Region)	В данной области находится корневая директория, которая является первой директорией диска и которая содержит в себе все директории и файлы текущей файловой системы.
Область файлов и директорий (File and Directory Data Region)	Является самой большой областью файловой системы, в ней хранятся пользовательские данные.

Первая важная структура на FAT диске называется BPB (BIOS Parameter Block), которая расположена в первом секторе диска в зарезервированной области. Это первый блок, который читается при обработке устройства. Наиболее важные для данной работы поля BPB приведены в табл. 2.

Таблица 2. Важные поля BPB

Наименование	Смещение	Размер	Описание
$BPB_{BytesPerSec}$	0x0B	2	Количество байт в секторе.
$BPB_{SecPerClus}$	0x0D	1	Количество секторов в кластере.
$BPB_{RsvdSecCnt}$	0x0E	2	Количество секторов в зарезервированной области.
$BPB_{NumFATs}$	0x10	1	Количество таблиц FAT на диске.
$BPB_{TotSec32}$	0x20	4	Общее количество секторов на диске.
BPB_{FATSz}	0x24	4	Количество секторов одной FAT.
$BPB_{RootClus}$	0x2C	4	Номер первого кластера корневой директории.

По данным, приведенным в табл. 1 можно рассчитать смещение для начала таблицы FAT по формуле:

$$FAT_{Begin} = BPB_{RsvdSecCnt} \times BPB_{BytesPerSec}. \quad (1)$$

Структура FAT содержит однонаправленные списки блоков (кластеров) файлов. Директория в FAT представляет собой обычный файл со специальным атрибутом. FAT в ходе своей работы распределяет данные файлов по номерам кластеров. Первой и обязательной директорией является корневая директория, ее смещение можно найти по формуле:

$$Root_{dir} = FAT_{Begin} + BPB_{NumFATs} \times BPB_{FATSz} \times BPB_{BytesPerSec.} \quad (2)$$

Данные файла ассоциируются с файлом следующим образом. В директории содержится номер первого кластера, в котором располагаются данные файла. Данные первого кластера ассоциированы с номером первого кластера, и расположение данных вычисляется из номера первого кластера. Смещение до первого сектора кластера с номером N можно рассчитать по формуле:

$$First_{DataofCluster}(N) = (N - BPB_{RootClus}) \times BPB_{SecPerClus} \times BPB_{BytesPerSec} + Root_{dir.}$$

Файлы нулевой длины имеют в директории номер первого кластера 0. Кластер, расположенный в FAT может содержать значение EOC (End Of Clusterchain) или номер следующего кластера файла. Значение EOC для FAT32 равно 0x0FFFFFFF. Список свободных кластеров нигде отдельно не хранится; он должен вычисляться каждый раз при подключении диска к системе, путем сканирования таблицы FAT.

В таблице FAT зарезервированы значения для первых двух кластеров. Первый зарезервированный кластер, FAT{0}, содержит значение 0x0FFFFFF8. Во второй зарезервированный кластер, FAT{1}, устанавливается значение EOC.

Предлагаемая в данной работе стеганографическая система основана на сокрытии информации в не занятых на данный момент времени кластерах ФС FAT32 в области файлов и директорий. Скрываемая информация первоначально шифруется.

Обеспечение сокрытия факта встраивания дополнительной информации в ФС происходит за счет особенностей механизма удаления файлов. В FAT32 удаление файлов происходит следующим образом:

1. В директории, где хранился файл, первый байт из имени файла заменяется на специальный символ (0xE5). Это означает, что данная ячейка в таблице освободилась.
2. Очищается цепочка кластеров в таблице FAT.

Тело файла при удалении не стирается, оно удаляется только когда происходит запись нового файла в те же кластеры ФС, где был расположен старый файл. Процесс удаления файла и данных о нем приведен на рис. 3.

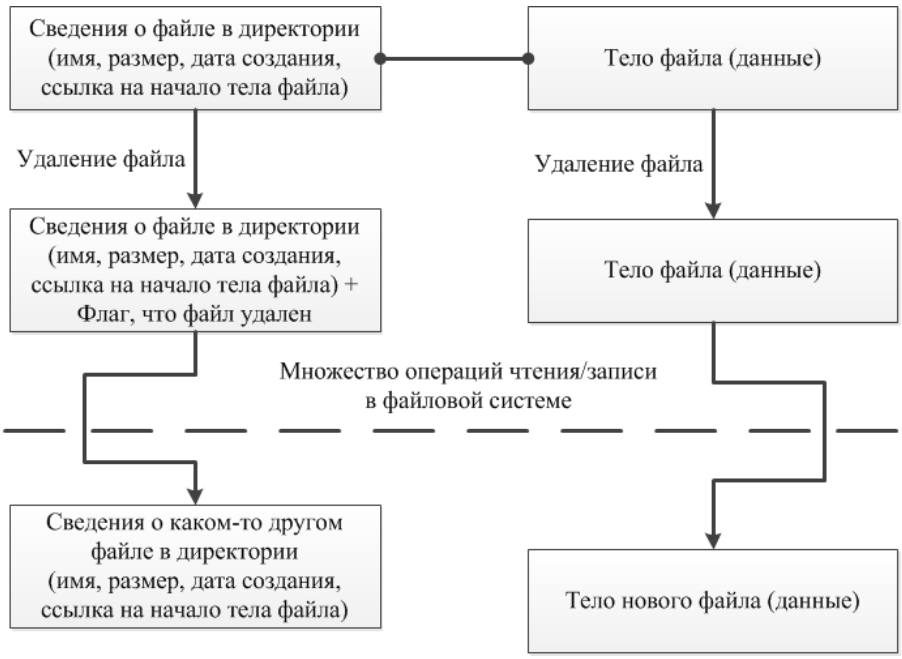

Рис. 3. Процесс удаления файла и данных о нем
(Fig. 3. The process of deleting a file and its data)

Таким образом для скрытного хранения информации могут быть использованы пустые кластеры ФС FAT32, при условии того, что ранее в этих кластерах хранилась какая-то информация.

3. Особенности применения кодов, исправляющих ошибки, в разработанной стегосистеме

Необходимо отметить важную особенность предложенного подхода к сокрытию информации в пустых кластерах ФС – при таком способе скрытого хранения файлов запись новой информации на диск может повредить сокрытый файл, в случае если записываемая информация попадет в один из кластеров, используемых для скрытого хранения информации. Эта особенность является ограничением использования предложенного способа, но и его достоинством, позволяющим в случае необходимости обычными манипуляциями с ФС уничтожить следы присутствия скрываемой информации.

Также данная особенность оказывает влияние на процедуру применения кодов, исправляющих ошибки. Обычные коды, исправляющие ошибки, предназначены для исправления одиночных ошибок или не очень больших пакетов ошибок [14–15]. Информация на извлекаемом носителе защищена от одиночных ошибок с помощью специальных алгоритмов, содержащихся во встроенном программном обеспечении [16].

В связи с вышеизложенным возникает необходимость защищаться только от ошибок, связанных с возможной перезаписью кластера ФС новыми данными. Обычно размер кластера в ФС равен нескольким килобайтам. Стандартные коды исправления ошибок могут справиться с таким пакетом ошибок только с применением техники перемежения [17], но небольшие файлы, размером в несколько десятков килобайт, нельзя будет восстановить в случае перезаписи одного кластера даже с применением этой техники. Поэтому в описываемых в следующих разделах алгоритмах записи и чтения маскируемого файла вместо кодов исправления ошибок применяется техника множественной записи информации, то есть ее дублирование.

4. Описание основных алгоритмов предложенного подхода

4.1. Алгоритм записи маскируемого файла в ФС

Входными данными для алгоритма записи маскируемого файла являются образ файловой системы, куда будет встраиваться информация; скрываемый файл, общий ключ шифрования и число копий файла, которые необходимо записать для исправления ошибок. Алгоритм записи маскируемого файла представлен на рис. 4.

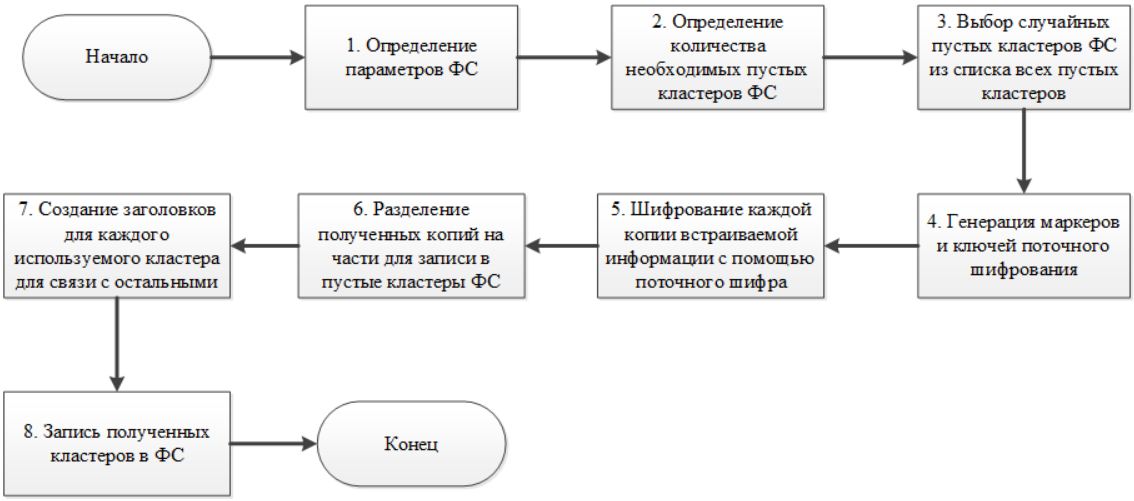

Рис. 4. Алгоритм записи маскируемого файла
(Fig. 4. Algorithm for writing a hidden file)

Первым шагом алгоритма записи является определение параметров ФС, приведенных в табл. 2. По полученным параметрам и с помощью формул (1) и (2) определяются адреса таблицы FAT и корневой директории. Также в таблице FAT проводится поиск всех пустых кластеров.

На втором шаге алгоритма определяется количество необходимых для записи информации кластеров ФС. Эта операция осуществляется на основе сведений о размере файла, размере кластера ФС и размере заголовков.

Разработанный алгоритм использует два вида заголовков: заголовок для первого кластера записываемого файла, и заголовок для всех последующих кластеров. Структура первого кластера записываемого файла представлена в табл. 3.

Таблица 3. Структура первого кластера записываемого файла

Поле	Маркер	Длина файла	Номер копии	Адрес следующих кластеров с данными файла во всех копиях	Данные файла
Длина	16 байт	6 байт	1 байт	4 байта × число копий	Длина кластера минус длина заголовка

Поле «маркер» необходимо для поиска первого кластера в алгоритме чтения. Оно формируется путем шифрования общего ключа шифрования, сцеплённого с номером текущей копии, на этом же ключе шифрования. От полученного значения в маркер сохраняются первые 16 байт. Следует отметить, что использование другого принципа формирования маркера не окажет существенного влияния на свойства предлагаемого способа.

Поле «длина файла» используется для хранения длины файла, оно необходимо, чтобы при считывании файла не была прочитана лишняя информация. Для кодирования этого и всех последующих значений используется обратный порядок байт.

Поле «номер копии» используется для определения какой копии файла принадлежит информация в поле данных файла, это поле необходимо при исправлении ошибок и расшифровке информации.

Поле «Адрес следующих кластеров с данными файла во всех копиях» указывает на следующий кластер ФС с данными файла для каждой копии файла. Это обеспечивает достаточную защиту данных файла, так как для корректного восстановления файла необходимо и достаточно, чтобы на носителе информации нашлась хотя бы одна копия каждой части файла, на которые он был разбит перед записью в кластеры ФС. Связь между различными частями различных копий одной и той же информации представлена на рис. 5. В случае если текущий кластер оказывается последним кластером, используемым для записи файла, т.е. все данные файла обработаны, данное поле указывает на нулевой кластер.

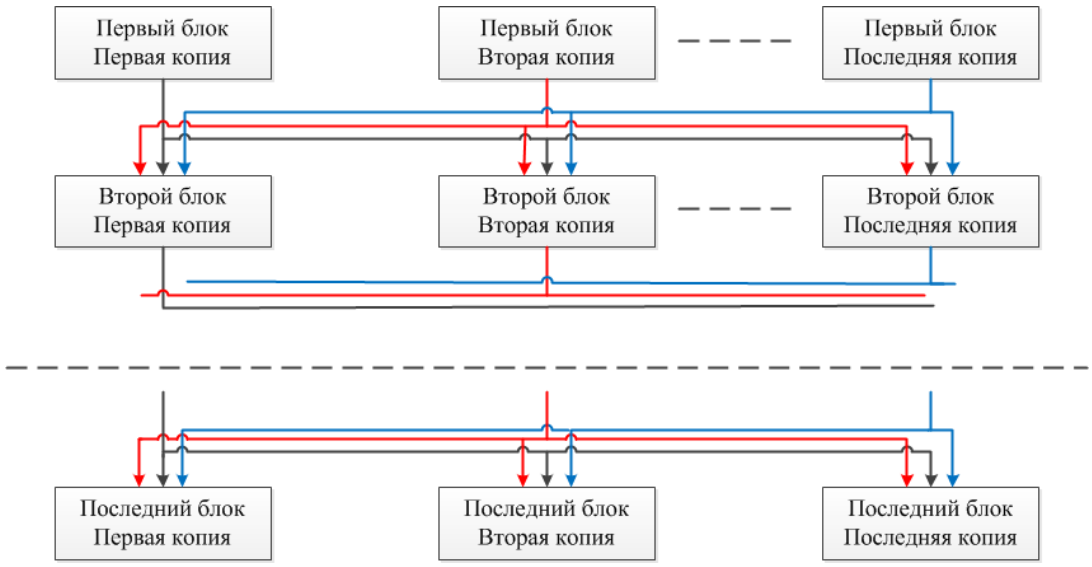

Рис. 5. Связь между различными частями копий вкрапляемой информации
(Fig. 5. Relationship between different parts of copies of embedded information)

В поле «данные файла» хранятся непосредственно данные файла. В случае если текущий кластер оказывается последним кластером, используемым для записи файла, т.е. все данные файла обработаны, данные файла дополняются нулевыми байтами, которые при чтении будут удалены.

Структура остальных кластеров, записываемого файла, представлена в табл. 4.

Таблица 4. Структура остальных кластеров, записываемого файла

Поле	Номер копии	Адрес следующих кластеров с данными файла во всех копиях	Данные файла
Длина	1 байт	4 байта × число копий	Длина кластера минус длина заголовка

На третьем шаге алгоритма производится выбор случайных кластеров из списка пустых кластеров ФС, в которые будет записан файл.

На четвертом шаге алгоритма вырабатываются маркеры для каждой копии скрываемого файла, а также различные ключи поточного шифрования для каждой копии файла. Один из способов выработки различных поточных ключей для каждой копии файла может быть описан формулой:

$$key[i] = Encrytion(marker[i] + key, key),$$

где *Encryption* – алгоритм блочного шифрования, *key* – общий ключ шифрования.

На пятом шаге алгоритма происходит шифрование каждой копии встраиваемого файла поточным шифром с использованием ключа, выработанного на предыдущем шаге.

На шестом шаге полученные зашифрованные копии разделяются на части для последующей записи в отдельные кластеры ФС.

На седьмом шаге алгоритма с использованием ранее полученных сведений формируются необходимые заголовки. Все данные заголовков шифруются с помощью блочного шифра за исключением поля «маркер». Затем эти данные добавляются к полученным на предыдущем этапе частям файла.

На последнем шаге алгоритма осуществляется запись полученных данных в соответствующие кластеры ФС.

4.2. Алгоритм чтения маскируемого файла из ФС

Входными параметрами для алгоритма чтения маскируемого файла являются образ файловой системы, где содержится скрытая информация; общий ключ шифрования и число копий файла, которые были записаны для исправления ошибок. Алгоритм чтения маскируемого файла представлен на рис. 6.

Первым шагом алгоритма чтения, как и алгоритма записи, является определение параметров ФС.

Вторым шагом алгоритма является генерация маркеров и поточных ключей шифрования по тому же алгоритму и с использованием того же общего ключа шифрования, что и при записи.

На третьем шаге работы алгоритма осуществляется перебор кластеров, начиная с корневого. В этих кластерах осуществляется поиск маркера. В случае успешного нахождения маркера осуществляется переход к четвертому шагу. Если все кластеры были обработаны, а кластер с маркером не был найден, делается вывод об отсутствии скрываемой информации.

Рис. 6. Алгоритм чтения маскируемого файла
(Fig. 6. Algorithm for reading a hidden file)

На четвертом шаге работы алгоритма производится расшифровка заголовка. Если заголовок был расшифрован без ошибок, то происходит переход на шаг 6.

Пятый шаг алгоритма предназначен для поиска неповрежденного кластера с данными в случае, если какая-то из копий была повреждена. В таком случае ищется кластер, принадлежащий следующей копии файла, адреса всех таких кластеров в соответствии с рис. 5 и табл. 3 находится в расшифрованном заголовке.

На шестом шаге сохраняются зашифрованные данные файла с указанием копии файла, из которой они были получены.

На седьмом шаге определяется адрес следующего кластера с данными файла, по соответствующему полю заголовка. Если полученный адрес равен нулю, то происходит переход на шаг 9.

На восьмом шаге считывается новый кластер по полученному на предыдущем шаге адресу и осуществляется переход на шаг 4.

На девятом шаге происходит расшифрование данных файла в соответствии с номером копии изначального файла, которой они принадлежали, и объединение полученных данных в один выходной файл.

На десятом шаге работы алгоритма осуществляется коррекция длины полученного выходного файла, представляющая собой удаление лишних нулевых байт, с учетом размера файла, полученного на третьем шаге работы алгоритма.

5. Характеристики предложенного способа сокрытия информации

5.1. Оценка пригодности предложенного способа сокрытия информации

Пригодность предложенного способа сокрытия информации может быть оценена, как совокупность следующих параметров:

- сохранность сокрытой информации;
- время работы алгоритма;
- процент от общего объёма памяти носителя информации, в котором может быть помещена скрываемая информация.

Особенностью разработанного способа является то, что сокрытая информация может быть повреждена при последующей записи данных в ФС. Данный недостаток устраняется путём дублирования скрываемой информации, хотя необходимо отметить, что в случае активного использования ФС информация все равно будет уничтожена.

Таким образом система в основном предназначена для скрытой передачи информации с помощью извлекаемого носителя информации или для длительного хранения данных без активного использования ФС носителя.

Время работы алгоритма записи в текущей реализации зависит в основном от размера записываемого файла и почти не зависит от размера ФС. Время работы алгоритма чтения зависит от размера сокрытого файла, но также на этот параметр влияет номер кластера, в котором впервые встретятся данные файла, и соответственно косвенным образом размер ФС.

Предложенный способ позволяет использовать для сокрытия информации все свободное место ФС в области файлов и директорий. При удалении всех файлов из ФС процент от общего объема памяти носителя информации, в котором может быть размещена скрываемая информация, равняется практически 100%.

Также необходимо отметить такое достоинство предложенного способа к сокрытию информации в файловой системе, как универсальность. Данный способ применим не только при работе с файловой системой FAT32, которая рассматривалась в работе из-за ее частого применения на внешних носителях информации, но и при использовании других блочных ФС, например, файловых систем из группы *ext,* встречающихся в операционных системах семейства *Linux.*

5.2. Доказательство совершенности стегосистемы

Оценить разработанную стеганографическую систему с точки зрения безопасности можно посредством доказательства её совершенности по Кашену [18]. Для этого необходимо рассчитать расстояние Кульбака-Лейблера [19] по формуле:

$$D(P_C \vee P_S) = \sum_{i=0}^{n-1} p_i log_2 \frac{p_i}{q_i},$$

где P_c – вероятностное распределение пустых контейнеров; а P_s – вероятностное распределение стеганографических контейнеров; p_i – вероятность пустого контейнера i; q_i – вероятность стеганографического контейнера i. Величина i – это число в двоичной форме, представляющее собой последовательность из нулей и единиц, определяющее контейнер. Максимально возможное $i_{max} = n\text{-}1 = 11...1$, где количество единиц – длина носителя.

С учетом того, что в данной работе используется качественный алгоритм шифрования данных, становится верной формула:

$$p_i \equiv p_j \equiv \frac{1}{2^n}.$$

Аналогично, из-за того, что стеганографическая информация вкрапляется после работы поточного шифра, становится верна формула:

$$q_i \equiv q_j \equiv \frac{1}{2^n}.$$

Следовательно, расстояние Кульбака-Лейблера равно строгому нулю. Таким образом разработанная стегосистема является совершенной по Кашену.

Заключение

В настоящей работе представлен способ стеганографического хранения информации в пустых кластерах файловой системы FAT32. Данный способ может быть адаптирован и для других блочных файловых систем.

Ограничение подхода к стеганографическому хранению информации, связанное с возможной перезаписью данных при работе с ФС, может быть решено с помощью кодов, исправляющих ошибки, и техники перемежения либо же посредством дублирования информации. К достоинствам предложенного решения следует отнести высокую степень безопасности сокрытых данных, отсутствие необходимости внесения изменений в принципы работы файловой системы, а также возможность скрытного хранения больших объемов данных.

СПИСОК ЛИТЕРАТУРЫ:

1. Авдошин С.М. Криптоанализ и криптография: история противостояния / С.М. Авдошин, А.А. Савельева // Бизнес-информатика, 2009, № 2(8). С. 3–11.
2. Жуков К.Д. Обзор атак на AES-128: к пятнадцатилетию стандарта AES / К.Д. Жуков // Прикладная дискретная математика, 2017, № 35. С. 48–62.
3. Задорожный Д.И. Сравнение криптостандартов ГОСТ Р 34.12-2015 и ГОСТ Р 34.13-2015 с ГОСТ 28147-89 / Д.И. Задорожный, А.М. Коренева // Защита информации. INSIDE, 2019, № 2. С. 80–88.
4. Бабенко Л.К. Особенности применения методов линейного и дифференциального криптоанализа к симметричным блочным шифрам / Л.К. Бабенко, Е.А. Ещукова // Вопросы кибербезопасности, 2015, №1 (9). С. 11–19.
5. U.S. Customs and border protection, CBP directive No 3340-049A, 04.01.2018 // URL: https://www.cbp.gov/sites/default/files/assets/documents/2018-Jan/CBP-Directive-3340-049A-Border-Search-of-Electronic-MediaCompliant.pdf (дата обращения: 26.07.2019).
6. Regulation of Ivestigatory Powers Act 2000, Part III // URL: http://www.legislation.gov.uk/ukpga/2000/23/part/III (дата обращения: 26.07.2019).
7. Макаренко С.И. Эталонная модель взаимодействия стеганографических систем и обоснование на ее основе новых направлений развития теории стеганографии / С.И. Макаренко // Вопросы кибербезопасности, № 2(3), 2014. С. 24–32.
8. Anderson R. The Steganographic File System / R. Anderson, R. Needham, A. Shamir // IH’98, Springler-Verlag, Berlin, LNCS 1525, 1999. P. 73–82.
9. McDonald A.D. StegFS: A Steganographic File System for Linux / A.D. McDonald, M.G. Kuhn // IH’99, Springler-Verlag, Berlin, LNCS 1768, 2000. P. 463–477.
10. Khan H. Evading Disk Investigation and Forensics using a Cluster-Based Covert Channel / H. Khan, M. Javed, F. Mirza, S.A. Khayam // NUST Technical Report, Islamabad, 2009. URL: http://web.lums.edu.pk/~mobin/publications/2009/CCS09.pdf (дата обращения: 26.07.2019).
11. Morkevichius N. Covert Channel for Cluster-based File Systems Using Multiple Cover Files / N. Morkevichius, G. Petraitis, A. Vechkauskas, J. Cheponis // Information technology and control, Vol. 42, № 3, Kaunas, Lithuania, 2013. P. 260–267.
12. Kuznetsov A. Hiding Data in the Structure of the FAT Family File System / A. Kuznetsov, K. Shekhanin, A. Kolhatin and others // The night IEEE International Conference on Dependable Systems, Services and Technologies, DESERT’2018, Kyiv, 2018. URL: https://ieeexplore.ieee.org/document/8409155 (дата обращения: 26.07.2019).
13. Microsoft Extensible Firmware Initiative FAT32 File System Specification / Microsoft Corporation, 2000. URL: download.microsoft.com/download/0/8/4/084c452b-b772-4fe5-89bba0cbf082286a/fatgen103.doc (дата обращения: 26.07.2019).
14. Морелос-Сарагоса Р. Искусство помехоустойчивого кодирования. Методы, алгоритмы, применение. – М.: Техносфера, 2006. – 320 с.
15. Блейхут Р. Теория и практика кодов, контролирующих ошибки. – М.: Мир, 1986. – 572 с.
16. Open NAND Flash Interface Specification / Intel Corporetion, Micron Technology, Phison Electronics Corporation, and others // Revision 3.2, 2013. – 304 p.
17. Новиков Р.С. Анализ эффективности методов перемежения данных для помехоустойчивых кодов в каналах связи с помехами / Р.С. Новиков. Вестник науки и образования Северо-Запада России, 2015, том 1, № 2. С. 1–7.
18. Cachin C. An Information-Theoretic Model for Steganography // 2002 Information Hiding, Lecture Notes in Computer Science, vol. 1525, Springer.
19. Kullback S., Leibler R.A. On information and sufficiency // The Annals of Mathematical Statistics. 1951. Vol. 22. № 1. P. 79–86.

REFERENCES:

[1] Avdoshin S.M. Kriptoanaliz i kriptografiya: istoriya protivostoyaniya S.M. Avdoshin, A.A. Savel'yeva. Biznes-informatika, 2009, № 2(8). S. 3–11 (in Russian).
[2] Zhukov K.D. Obzor atak na AES-128: k pyatnadtsatiletiyu standarta AES K.D. Zhukov. Prikladnaya diskretnaya matematika, 2017, № 35. S. 48–62 (in Russian).
[3] Zadorozhnyy D.I. Sravneniye kriptostandartov GOST R 34.12-2015 i GOST R 34.13-2015 s GOST 28147-89 D.I. Zadorozhnyy, A.M. Koreneva. Zashchita informatsii. INSIDE, 2019, № 2. S. 80–88 (in Russian).
[4] Babenko L.K. Osobennosti primeneniya metodov lineynogo i differentsial'nogo kriptoanaliza k simmetrichnym blochnym shifram L.K. Babenko, Ye.A. Yeshchukova. Voprosy kiberbezopasnosti, 2015, № 1(9). S. 11–19 (in Russian).
[5] U.S. Customs and border protection, CBP directive No 3340-049A, 04.01.2018. URL: https://www.cbp.gov/sites/default/files/assets/documents/2018-Jan/CBP-Directive-3340-049A-Border-Search-of-Electronic-MediaCompliant.pdf (accessed: 26.07.2019).
[6] Regulation of Ivestigatory Powers Act 2000, Part III URL: http://www.legislation.gov.uk/ukpga/2000/23/part/III (accessed: 26.07.2019).
[7] Makarenko S.I. Etalonnaya model' vzaimodeystviya steganograficheskikh sistem i obosnovaniye na yeye osnove novykh napravleniy razvitiya teorii steganografii S.I. Makarenko. Voprosy kiberbezopasnosti, № 2(3), 2014. S. 24–32 (in Russian).
[8] Anderson R. The Steganographic File System R. Anderson, R. Needham, A. Shamir. IH'98, Springler-Verlag, Berlin, LNCS 1525, 1999. P. 73–82.
[9] McDonald A.D. StegFS: A Steganographic File System for Linux A.D. McDonald, M.G. Kuhn. IH'99, Springler-Verlag, Berlin, LNCS 1768, 2000. P. 463–477.
[10] Khan H. Evading Disk Investigation and Forensics using a Cluster-Based Covert Channel H. Khan, M. Javed, F. Mirza, S.A. Khayam. NUST Technical Report, Islamabad, 2009. URL: http://web.lums.edu.pk/~mobin/publications/2009/CCS09.pdf (accessed: 26.07.2019).
[11] Morkevichius N. Covert Channel for Cluster-based File Systems Using Multiple Cover Files N. Morkevichius, G. Petraitis, A. Vechkauskas, J. Cheponis. Information technology and control, Vol. 42, № 3, Kaunas, Lithuania, 2013. P. 260–267.
[12] Kuznetsov A. Hiding Data in the Structure of the FAT Family File System A. Kuznetsov, K. Shekhanin, A. Kolhatin and others. The night IEEE International Conference on Dependable Systems, Services and Technologies, DESERT'2018, Kyiv, 2018. URL: https://ieeexplore.ieee.org/document/8409155 (accessed: 26.07.2019).
[13] Microsoft Extensible Firmware Initiative FAT32 File System Specification / Microsoft Corporation, 2000. URL: download.microsoft.com/download/0/8/4/084c452b-b772-4fe5-89bba0cbf082286a/fatgen103.doc (accessed: 26.07.2019).
[14] Morelos-Saragosa R. Iskusstvo pomekhoustoychivogo kodirovaniya. Metody, algoritmy, primeneniye. – Moskva: Tekhnosfera, 2006. – 320 s. (in Russian).
[15] Bleykhut R. Teoriya i praktika kodov, kontroliruyushchikh oshibki. – Moskva: Mir, 1986. – 572 s. (in Russian).
[16] Open NAND Flash Interface Specification Intel Corporetion, Micron Technology, Phison Electronics Corporation, and others. Revision 3.2, 2013. – 304 p.
[17] Novikov R.S. Analiz effektivnosti metodov peremezheniya dannykh dlya pomekhoustoychivykh kodov v kanalakh svyazi s pomekhami R.S. Novikov. Vestnik nauki i obrazovaniya Severo-Zapada Rossii, 2015, tom 1, № 2. S. 1–7 (in Russian).
[18] Cachin C. An Information-Theoretic Model for Steganography. 2002 Information Hiding, Lecture Notes in Computer Science, vol. 1525, Springer.
[19] Kullback S., Leibler R.A. On information and sufficiency. The Annals of Mathematical Statistics. 1951. Vol. 22. № 1. P. 79–86.

Поступила в редакцию – 22 января 2020 г. Окончательный вариант – 09 февраля 2020 г.
Received – January 22, 2020. The final version – February 09, 2020.