Оглавление

[Введение 3](#_Toc122688093)

[1 Анализ существующих методов и алгоритмов скрытой передачи данных между USB-носителями информации и ПК 5](#_Toc122688094)

[1.1 Расширенная постановка задачи 5](#_Toc122688095)

[1.2 Компьютерная стеганография 8](#_Toc122688096)

[Стеганография в файловой системе USB-носителя с использованием избыточности (cтеганография мимо файлов) 10](#_Toc122688097)

[Выводы по первому разделу. 13](#_Toc122688098)

[2 Разработка алгоритмов и методов скрытой передачи данных на USB-носителях 14](#_Toc122688099)

[2.1 Разработка структуры программного обеспечения скрытой передачи данных 14](#_Toc122688100)

[2.1.1 Серверное приложение 14](#_Toc122688101)

[2.1.2 Клиентское приложение 14](#_Toc122688102)

[2.2 Разработка протокола взаимодействия 15](#_Toc122688103)

[2.3 Разработка алгоритма работы макета программного средства 17](#_Toc122688104)

[2.3.1 Алгоритма работы макета программного средства 17](#_Toc122688105)

[2.3.2 Серверное приложение 18](#_Toc122688106)

[2.3.3 Клиентское приложение 22](#_Toc122688107)

[Выводы по второму разделу 26](#_Toc122688108)

[Заключение 27](#_Toc122688109)

[Список использованных источников 28](#_Toc122688110)

# Введение

Операционные технологии, или ОТ, являются критически важным сегментом сети, используемым предприятиями, которые производят товары или занимаются физическими процессами. Такие отрасли, как производство, химическая, нефтегазовая и горнодобывающая промышленности, транспортная и логистическая сферы, используют специализированные технологии для управления объектами: сборочными и производственными площадками, энергосистемами. Контроль, мониторинг и управление в соответствующей области были постепенно автоматизированы за последние несколько десятилетий, и специализированные комплексы, которые решают эти задачи, представлены промышленными системами управления (АСУ ТП), диспетчерского контроля и сбора данных (SCADA) или просто OT. Сети, в которых они работают, традиционно были отделены от корпоративной среды информационных технологий (IT) и от интернета, часто отграничены «воздушной прослойкой». Они обычно управляются оперативным персоналом, а не IT-специалистами — и не зря. Производственные мощности могут приносить компании миллионы долларов в час, а для обычных граждан инфраструктура, обеспечивающая чистую воду и энергию, попросту жизненно важна. Когда эти системы выходят из строя даже на несколько минут, это может принести сотни тысяч долларов убытка и даже подвергать риску работников и живущих поблизости людей. Проще говоря, IT — это управление данными, а ОТ — создание вещей. И поскольку системы OT были полностью изолированными, мир операционных технологий чувствовал себя невосприимчивым к взлому, который стал повседневным фактом для IT-сред.

Но недавние атаки на ОТ изменили привычный порядок вещей. Количество кибератак в этих системах и в инфраструктурах ОТ в целом растёт, и они наносят реальный ущерб. Вероятно, первая такая атака произошла десять лет назад, когда червь Stuxnet поразил объекты иранской ядерной программы. Это была система с «воздушным зазором», то есть она не имела связи с внешними сетями, но, тем не менее, была взломана. В 2017 году активность вымогателя NotPetya прервала производство и закрыла офисы. В том же году вредоносная программа Trisis/Triton нанесла ущерб средствам безопасности в оборудовании для добычи нефти и газа. А в 2020 году появился шифровальщик Ekans (он же Snake), который специально предназначен для АСУ ТП.

Изоляция SCADA, АСУ ТП или ОТ-инфраструктуры от корпоративной среды и интернета — традиционная мера защиты объектов КИИ от киберугроз. Однако «воздушная прослойка» никогда не обеспечивала полной безопасности, хотя и усложняла взлом критически важных систем. Связь с изолированной сетью обычно осуществляется с помощью физического устройства — USB-флешки или внешнего жесткого диска.

Целью настоящей работы является создание макета программного средства скрытной коммуникации между компьютерными сетями с воздушным зазором. Актуальные задачи, решаемые в рамках настоящей работы, включают в себя:

1. Анализ существующих методов и алгоритмов скрытой передачи данных между USB-носителями информации и ПК.

2. Разработка структуры программного обеспечения скрытой передачи данных.

3. Разработка протокола взаимодействия «клиентов» и «серверов» для передачи данных между ними.

4. Разработка алгоритма работы макета программного средства скрытой передачи данных.

В настоящей работе реализовано макет программного средства для решения задач получения данных из изолированной компьютерной сети. В первой части работы проанализированы существующие методы и алгоритмы скрытой передачи данных между USB-носителями информации и ПК. Во второй части разработаны структура ПО скрытой передачи данных, протокол взаимодействия клиентской и серверной стороны программного средства для передачи данных между ними, а также алгоритм работы макета программного средства.

# Анализ существующих методов и алгоритмов скрытой передачи данных между USB-носителями информации и ПК

## Расширенная постановка задачи

Рассмотрим задачу организации передачи данных между сетями с воздушным зазором.

Во-первых, «воздушный зазор» никогда не обеспечивал полной безопасности, хотя изоляция действительно усложняла взлом системы ОТ. Получение физического доступа всегда было возможным с помощью инструментов социальной инженерии, таких как оставление заражённой USB-флешки на парковке или уверенный проход на территорию организации под видом сотрудника.

Во-вторых, если вы думаете, что ваша OT-среда отделена «воздушной прослойкой», то вы, вероятно, ошибаетесь. Доступ для технического обслуживания к промышленным машинам, удалённое обновление инструментов АСУ ТП или встроенного ПО — всё это оставляет потенциальные уязвимости в среде OT, о которых вы, возможно, даже не подозреваете.

Основным отличительным признаком инструментов атаки, является способ заражения USB-накопителя, поэтому условно разделим фреймворки на две группы — работающие в системе, обеспечивающей связь с изолированной сетью, и работающие оффлайн. Первые после установки развертывают вредоносный компонент, который отслеживает подключение флешки и автоматом внедряет в нее код для взлома системы, отделенной «воздушным зазором» (Рис. 1.1).

В этом сценарии атака первоначально нацелена на подключенную к Интернету систему, которая взаимодействует со средой с «воздушным зазором». Однажды скомпрометированная, эта система используется для оснащения USB-накопителей вредоносной полезной нагрузкой и некоторым механизмом для компрометации следующей цели: системы с воздушным зазором. Обычно это становится возможным благодаря тому, что USB-накопитель используется для передачи информации между обеими сторонами, в большинстве случаев, ничего не подозревающей жертвой.

Полезная нагрузка, работающая в системе с воздушным зазором, обычно внедряется с целью разведки и кражи данных, которые сохраняются на USB-накопитель. Когда USB-накопитель попадает в первоначально скомпрометированную систему, его содержимое извлекается, а данные передаются злоумышленнику через Интернет.

Некоторые фреймворки идут еще дальше и поддерживают протокол двусторонней связи: через первоначально скомпрометированную систему на подключенной к Интернету стороне злоумышленник отправляет команды вредоносному ПО.

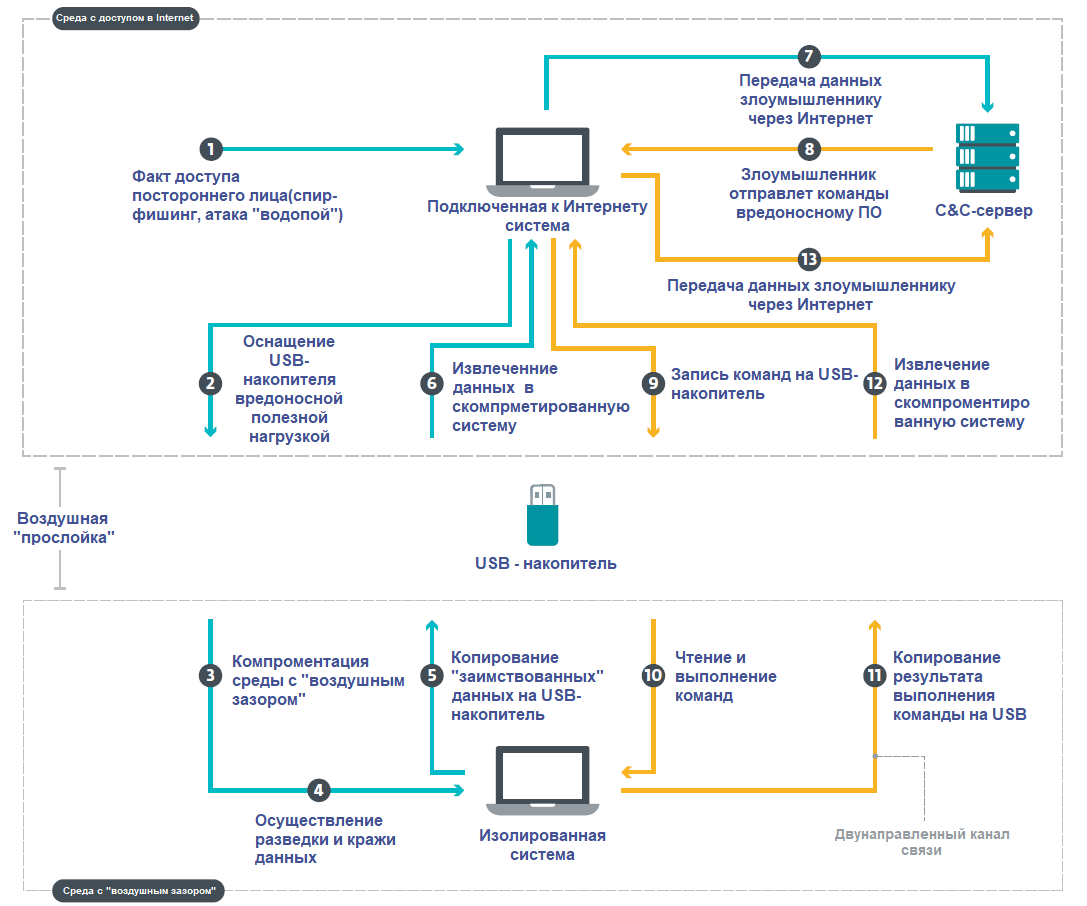


Рис. 1.1. Связный фреймворк.

В других, более редких случаях, сценарий атаки вообще не подразумевает никаких подключенных к Интернету систем. Их называют «автономными фреймворками» или «оффлайн-фреймворками» (Рис. 1.2). В этих случаях злоумышленник выполняет роль координатора процесса скрытого хищения данных с помощью заранее подготовленного вредоносного USB-носителя.

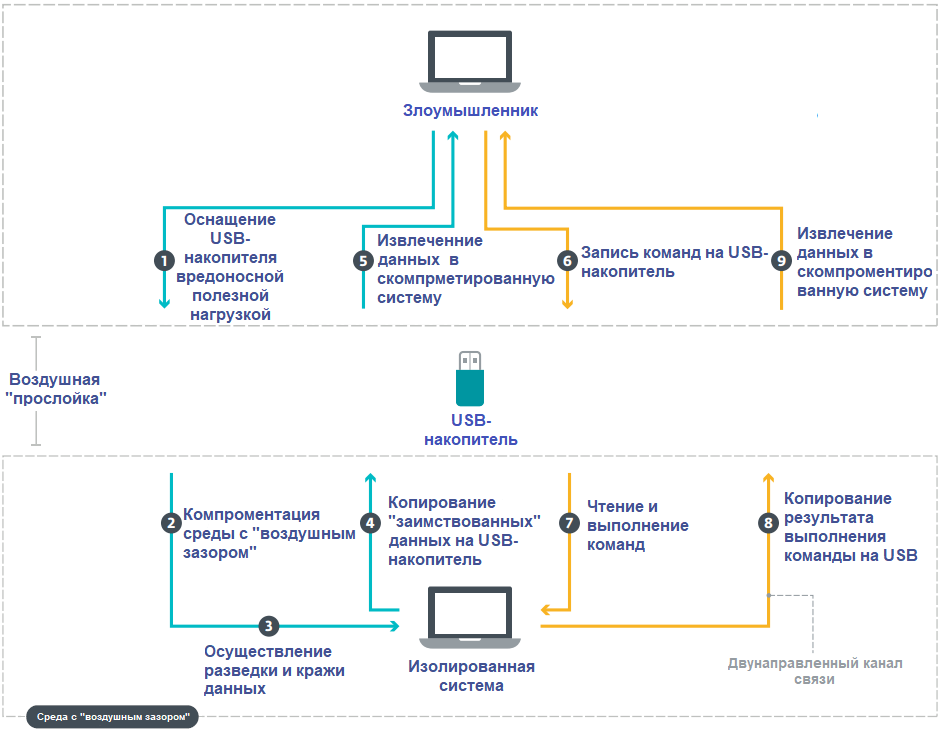


Рис. 1.2. Автономный фреймворк.

В рамках данной работы подразумевалось, что физически изолированная среда, задачу взаимодействия с которой необходимо решить, осуществляет внутреннюю коммуникацию с помощью USB-носителя. Ни о какой физически изолированной локальной сети речи не идет: существует лишь некоторое количество электронных вычислительных машин (компьютеров), которые осуществляют обмен данными с помощью флэшки. Помимо всего прочего, подразумевается, что одна из вычислительных машин (условный сервер) имеет доступ к глобальной сети Internet, что обеспечило бэкдор для C&C-сервера, осуществляющего управление внедренным шпионским ПО.

Макет программного средства, реализованного в настоящей работе, выступает в роли приложения, осуществляющего клиент-серверное взаимодействие, используя USB-накопитель как среду передачи данных от источника к приемнику по разработанному протоколу передачи данных, скрытно от пользователя (и программиста, вооруженного HEX-редактором).

## Компьютерная стеганография

Всем известно, что предназначение криптографии – сделать невозможным чтение секретной информации. Разумеется, криптография имеет свои области применения, но есть и другой подход к защите данных. Можно не шифровать информацию, а сделать вид, что у нас её нет. Именно для этого и придумана стеганография.

Стеганография (от греч. στεγανοσ — скрытый и греч. γραφω) — это наука о скрытой передаче информации путём сохранения в тайне самого факта передачи. В отличие от криптографии, которая скрывает содержимое секретного сообщения, стеганография скрывает само его существование. Стеганографию обычно используют совместно с методами криптографии, таким образом, дополняя её.

Компьютерная стеганография — направление классической стеганографии, основанное на особенностях компьютерной платформы. Например, стеганографическая файловая система StegFS для Linux, скрытие данных в неиспользуемых областях форматов файлов, подмена символов в названиях файлов, текстовая стеганография и т. д. Примеры:

• Использование зарезервированных полей компьютерных форматов файлов — суть метода состоит в том, что часть поля расширений, не заполненная информацией о расширении, по умолчанию заполняется нулями. Соответственно мы можем использовать эту «нулевую» часть для записи своих данных.

Недостатком этого метода является низкая степень скрытности и малый объем передаваемой информации.

• Метод скрытия информации в неиспользуемых местах гибких дисков — при использовании этого метода информация записывается в неиспользуемые части диска, к примеру, на нулевую дорожку.

Недостатки: маленькая производительность, передача небольших по объему сообщений.

• Метод использования особых свойств полей форматов, которые не отображаются на экране — этот метод основан на специальных «невидимых» полях для получения сносок, указателей. К примеру, написание черным шрифтом на черном фоне.

Недостатки: маленькая производительность, небольшой объём передаваемой информации.

• Использование особенностей файловых систем — при хранении на жестком диске файл всегда (не считая некоторых ФС, например, ReiserFS) занимает целое число кластеров (минимальных адресуемых объемов информации). К примеру, в ранее широко используемой файловой системе FAT32 (использовалась в Windows98/Me/2000) стандартный размер кластера — 4 Кб. Соответственно для хранения 1 Кб информации на диске выделяется 4 Кб информации, из которых 1Кб нужен для хранения сохраняемого файла, а остальные 3 ни на что не используются — соответственно их можно использовать для хранения информации.

Недостаток данного метода: лёгкость обнаружения.

### Стеганография в файловой системе USB-носителя с использованием избыточности (cтеганография мимо файлов)

Основная идея данного метода состоит в том, что на носителе информации имеются области, в которые операционная система никогда не пишет (или пишет в редких случаях). Чтобы не нужно было искать эти области хитрыми алгоритмами, воспользуемся избыточностью, то есть много-много раз продублируем скрытую информацию по всем секторам носителя. После чего провести форматирование данного носителя для дальнейшего использования. Так как форматирование не затрагивает некоторые сектора носителя, часть скрытых данных сохранится и ее можно будет извлечь, а многократное дублирование поможет составить из частей исходное целое.

При разделении носителя информации на разделы можно обратить внимание, что не всегда получается использовать все свободное пространство. Первый раздел всегда начинается с некоторым отступом (обычно это 1 мегабайт, или 2048 секторов). За последним разделом тоже часто остается небольшой «хвост» из неиспользуемых секторов. Между разделами также, но реже, остаётся неиспользованное пространство. Другими словами, на носителе есть сектора, к которым нет доступа при обычной работе, но данные в эти сектора записать можно.

При создании файловой системы на пустом разделе практически всегда остаются нетронутые сектора. Так как при быстром форматировании задача сводится к перезаписи только заголовка.

Также можно заметить, что файловая система не всегда может занять все отведенное ей пространство до последнего сектора. Связанно это с тем, что файловая система может полностью занять только пространство кратное размеру её кластера. Таким образом в конце такого раздела будет оставаться «хвост» размером в несколько секторов, недоступный для хранения пользовательских данных.

Итак, чтобы максимизировать место, доступное под стеганограмму, нужно использовать файловую систему с размером кластера как можно больше. Создавать пустые разделы или оставлять нераспределенные области не нужно – это привлечет лишнее внимание.

Понятно, что если просто размазать секретные данные по всему диску, то скрыты они будут только от невооруженного взгляда. При подробном рассмотрении диска при помощи HEX-редактора, данные все равно будет видно. Поэтому данные необходимо зашифровать. В качестве метода шифрования был выбран алгоритм aes256-cbc.

Следующий вопрос состоит в том, как отличить «правильные» данные от испорченных. Для решения этой проблемы предлагается использовать контрольную сумму SHA1. После добавления контрольной суммы в каждый сохраненный фрагмент информации будет понятно, удалась ли расшифровка: если после расшифровки она совпала – значит, расшифровка удалась.

Также необходим номер фрагмента и общая длина данных. Номер фрагмента – чтобы отслеживать, какие части информации уже расшифрованы, а какие – нет. Общая длина нам понадобится при обработке последнего фрагмента, чтобы не писать лишние данные. Также для удобства было добавлено имя файла.

Тестирование данного метода происходила поэтапно, проверяя после каждого этапа, сохранилась ли скрытая информация:

* Быстрое форматирование в формате FAT16 с размером кластера 16 кб ;
* Заполнение носителя случайными данными на 50%;
* Заполнение носителя случайными данными на 100%;
* «Долгое» форматирование в формате FAT16 (с перезаписью всего).

Первые два испытания вполне ожидаемо закончились успешно: утилита смогла успешно извлечь 10 мегабайт скрытых данных. Но после третьего этапа произошел сбой:

Total clusters read: 250752, decrypted: 158

ERROR: cannot write incomplete secretFile

Как видим, удалось успешно расшифровать только 158 кластеров (632 килобайта необработанных данных, что дает 636424 байта полезной нагрузки). Очевидно, что это меньше 10 мегабайт, а среди этих кластеров явно есть дубликаты. Даже 1 мегабайт таким образом уже не восстановишь. Но можно гарантировать, что 3 килобайта скрытых данных восстановить даже после того, как ее отформатируют и запишут под завязку.

Последнее испытание, к сожалению, показало, что носитель перезаписался полностью и ни одного кластера не сохранилось:

Device size: 250752 clusters

250700 99%

Total clusters read: 250752, decrypted: 0

ERROR: cannot write incomplete secretFile

Таблица 1.1 – Сводная информация о тестировании

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Размер тестируемого файла | 3 КБ | 30 КБ | 120 КБ | 1 МБ | 10 МБ |
| Быстрое форматирование | + | + | + | + | + |
| Заполнение на 50% | + | + | + | + | + |
| Заполнение на 100% | + | + | + | – | – |
| Полное форматирование | – | – | – | – | – |

Достоинства метода:

* Не зависит от формата файлов;
* Не зависит от используемой файловой системы.

Недостатки метода:

* Скрываемые данные можно будет изменить только полной перезаписью всего диска, с последующим воссозданием видимого пользователю содержимого, что очень быстро может привести носитель в нерабочее состояние;
* Чем больше объем скрываемых данных, тем больше вероятность потери части информации;
* Извлечение данных с носителя может занять много времени: от нескольких минут до нескольких дней.

## Выводы по первому разделу.

Рассмотренный метод не удовлетворяет поставленным задачам, так как для записи скрытой информации требуется полная перезапись всего носителя с последующим воссозданием видимого пользователю содержимого, что очень быстро может привести носитель в неисправное состояние.

Из чего сделан вывод о необходимости создания своего метода, удовлетворяющего задачам, поставленным в рамках данной работы.

# Разработка алгоритмов и методов скрытой передачи данных на USB-носителях

## Разработка структуры программного обеспечения скрытой передачи данных

### Серверное приложение

Макет серверного приложения состоит из трех модулей:

* Основной модуль;
* Подпрограмма записи запроса на диск;
* Подпрограмма чтения ответа с диска.

В первом модуле выполняется основная работа программного средства: работа с удаленным управляющим узлом посредством сетевого взаимодействия, вызов подпрограмм.

Второй модуль выполняет функцию записи полученного от управляющего узла запроса на USB-носитель.

Последний модуль манипулирует с данными, имеющимися на USB-накопителе: если на флешке имеется ответ на отправленный ранее запрос, то он считывает данный ответ и передает его основному модулю.

### Клиентское приложение

Макет клиентского приложения состоит из трех модулей:

* Основной модуль;
* Подпрограмма чтения запроса с USB-носителя;
* Подпрограмма записи ответа на USB-носитель.

В первом модуле происходит основная работа программного средства: обработка полученных запросов, формирование пакетов ответов, вызов подпрограмм.

Второй модуль выполняет чтение запросов с флэшки и передачу их основному модулю.

Третий модуль производит запись сформированных пакетов ответов на полученные запросы на диск.

## Разработка протокола взаимодействия

Для возможности взаимодействия серверного и клиентских ПК требуется создать протокол передачи данных. Для начала стоит определить, что будет находиться в заголовке пакета:

* Идентификатор отправителя пакета;
* Идентификатор получателя пакета;
* Тип пакета;
* Код пакета;
* Номер запроса;
* Длина сообщения;
* Контрольная сумма;
* Данные.

Идентификаторы отправителя и получателя состоят из 16 байт и нужны для однозначного определения кому и от кого назначается пакет. Идентификатор серверного ПК равен 0xFFFFFFFFFFFFFFFF. Идентификатор клиентского ПК рассчитывается по алгоритму MD5 из ключа реестра HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Cryptography\MachineGuid, который генерируется уникальным образом во время установки Windows и не изменится независимо от замены оборудования (кроме замены загрузочного жесткого диска, на котором установлена ​​ОС).

Поле «Тип пакета» состоит из 1 байта и предназначено для идентификации запросов и ответов на запросы. Равен 0x00 для запроса и 0xFF для ответа.

Поле «Код пакета» – числовой идентификатор, состоящий из 1 байта, более точно определяющий тип ошибки:

* 0x00 – ошибок нет;
* 0x01 – файл не существует;
* 0xFF – другие ошибки.

Поле «Номер запроса» состоит из 4 байт и содержит в себе порядковый номер запроса в пакете с типом «Запрос». В пакете с типом «Ответ» содержит номер запроса, на который отправляется ответ.

Поле «Длина сообщения» состоит из 6 байт и содержит размер всего пакета в байтах.

Контрольная сумма вычисляется для всего сообщения аналогично подсчету контрольной суммы в протоколе TCP.

Поле данных зависит от типа пакета: если это запрос, то в этом поле будет записан абсолютный путь до запрашиваемого файла, если ответ – то содержание запрашиваемого файла. Поле данных шифруется по алгоритму AES.

## Разработка алгоритма работы макета программного средства

### Алгоритма работы макета программного средства

Данный метод основывается на том, что в файловой системе носителя информации имеются служебные области, предназначенные для функционирования самой файловой системы. В структуре семейства файловых системы Ext есть так называемы Superblock, Block Bitmap и Inode Table, в которых дана основная информация о системе. Отсюда вытекает идея хранения скрытой информации в пустых блоках ФС.

Если хранить скрываемую информацию без шифрования, то, даже несмотря на ее размытость по ФС, она будет все равно слишком бросаться в глаза, особенно если знать, что следует искать. Поэтому было принято решение шифровать все блоки исходного файла. В качестве алгоритма шифрования был выбран AES.

Для отделения нужных блоков от всех остальных при чтении в каждый блок было решено добавить по специальному маркеру в начало блока. Этот маркер шифровался в зависимости от номера блока в исходном файле. Такая уловка сразу позволила не только находить нужные блоки, но и узнавать их правильный порядок.

Алгоритм записи:

1. Сначала записать в исходную файловую систему какую-либо информацию;
2. Удалить эту информацию;
3. Файл для сокрытия разбить на блоки одинаковой длины, добавив маркер;
4. Зашифровать эти блоки;
5. Поместить зашифрованные блоки в пустые блоки ФС.

Соответственно после того, как файл записан, его требуется считать:

1. Со знанием ключа и способа построения маркеров составить первые N маркеров, с гарантией что N, умноженное на длину блока файловой системы больше длины застеганографированного файла;
2. Произвести поиск блоков в ФС, начинающихся с маркеров;
3. Расшифровать полученные блоки и отделить маркеры;
4. Собрать полученные блоки в правильном порядке и получить исходный файл.

### Серверное приложение

Работа серверного приложения подразумевает под собой управление через пользовательский интерфейс: все запросы приходят по сети с удаленного узла. Также результат работы данного ПС должен отсылаться на управляющий узел посредством интерфейса сетевого взаимодействия.

Все запросы, полученные от управляющего узла, записываются в очередь и выполняются согласно порядку поступления. Так, если в момент получения запроса в компьютер не вставлен USB-носитель, то данный запрос будет записан в очередь, и не утерян.

Алгоритм работы приведен в виде блок-схем.

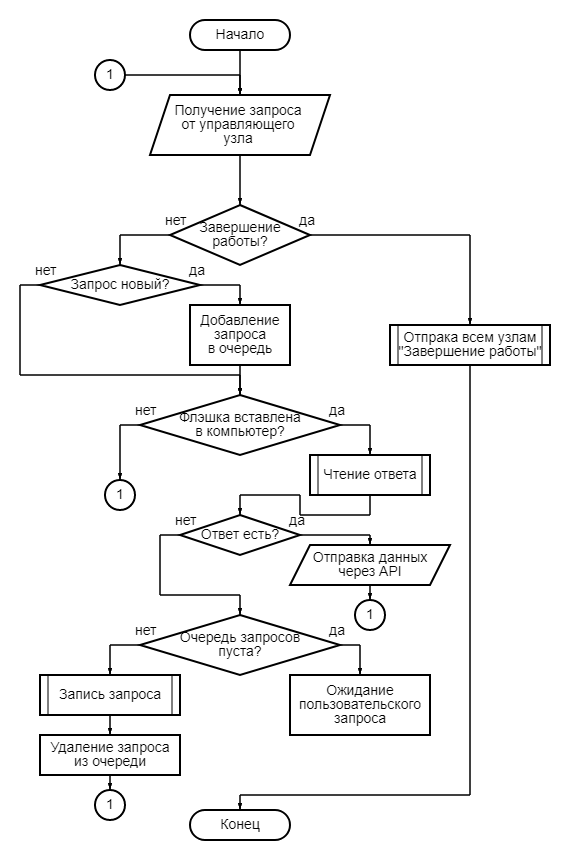


Рис. 2.3.1 – Основной алгоритм работы

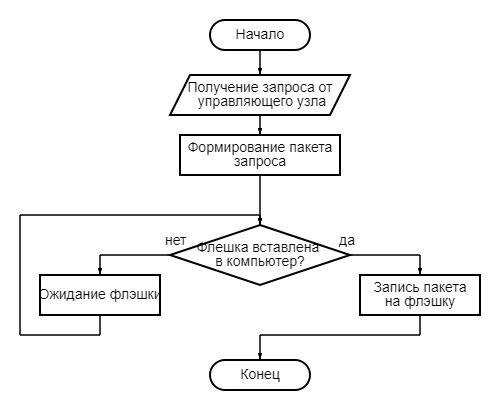


Рис. 2.3.2 – Подпрограмма «Запись запроса»

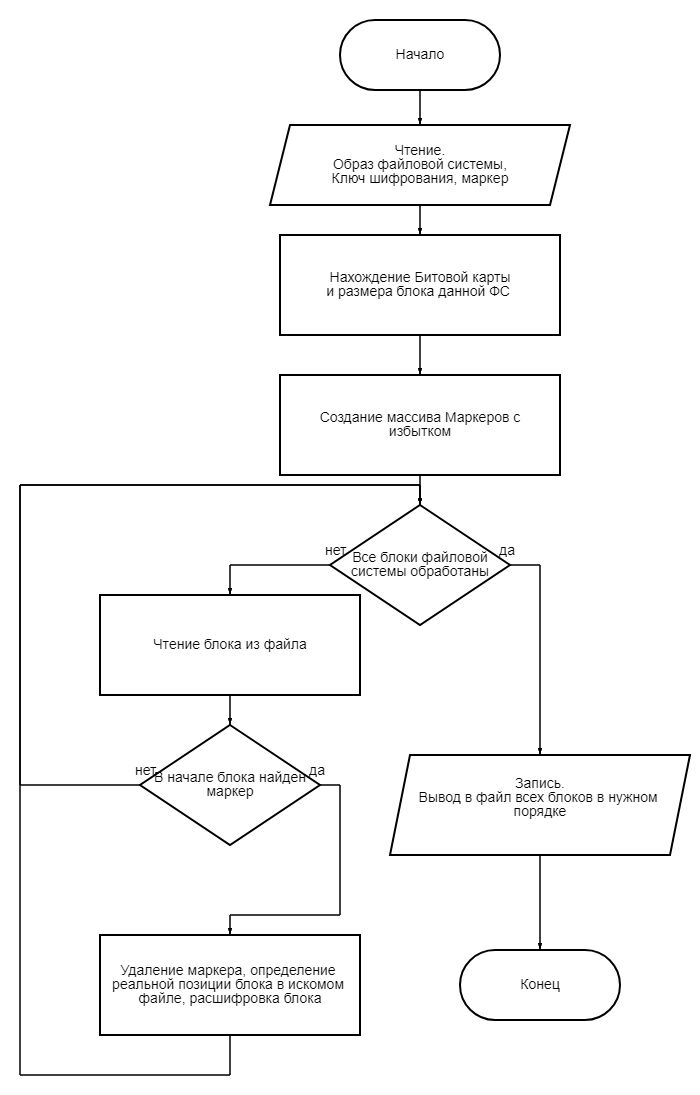


Рис. 2.3.3 – Подпрограмма «Чтение ответа»

### Клиентское приложение

Работа клиентского программного средства подразумевает под собой работу с запросами, полученными с флэшки.

При получении запроса, происходит его обработка, формирование пакета ответа и запись его на USB-накопитель. Если запросов на носителе информации не имеется и очередь запросов пуста, то программное средство ожидает получения запроса от следующей флэшки.

Алгоритм работы приведен в виде блок-схем.

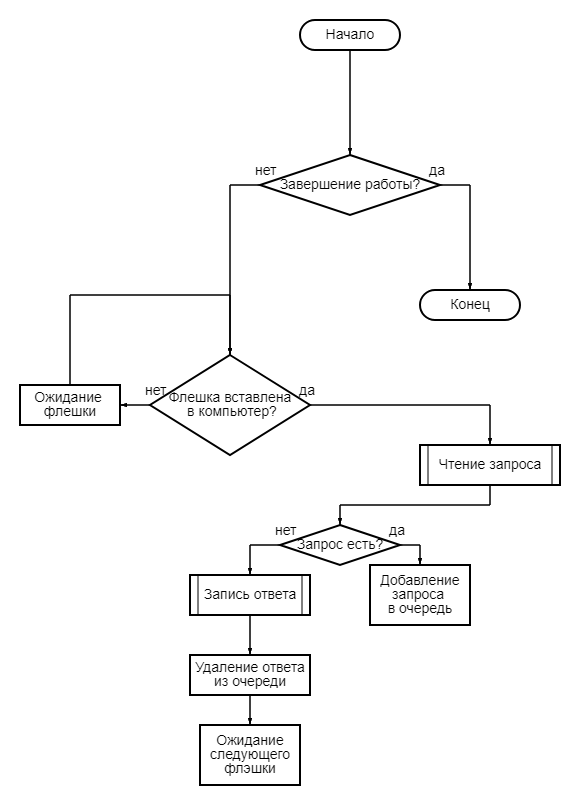


Рис 2.3.4 – Основной алгоритм работы

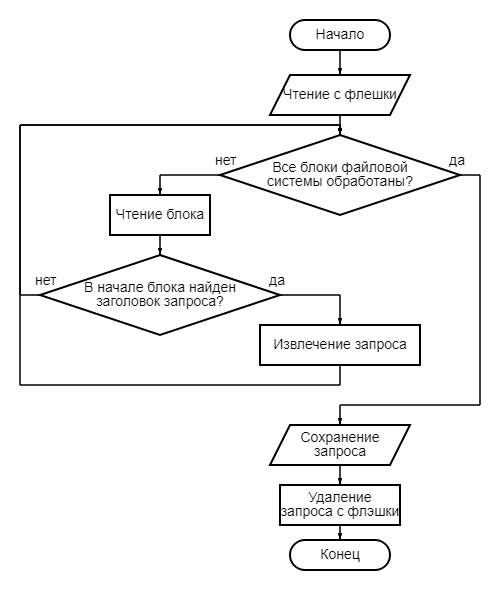


Рис 2.3.5 – Подпрограмма «Чтение запроса»

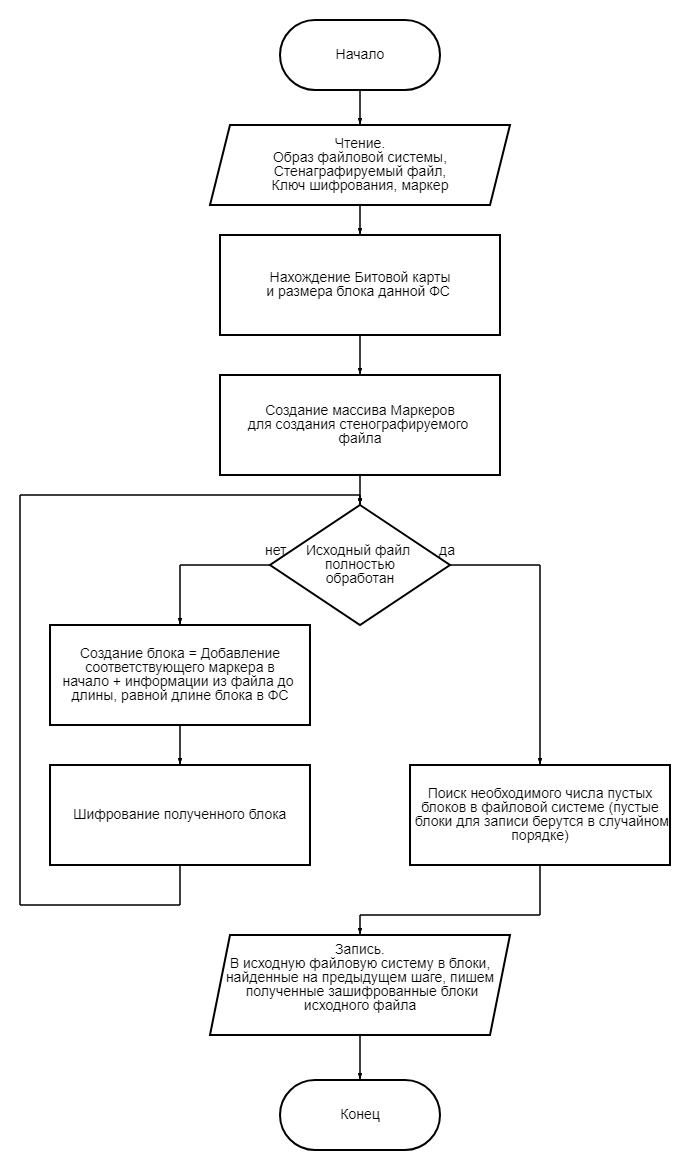


Рис 2.3.6 – Подпрограмма «Запись ответа»

## Выводы по второму разделу

Разработана структура клиент-серверного программного средства обеспечения скрытой передачи данных. Была выбрана модульная структура, состоящая из независимых компонентов, которые взаимодействуют между собой посредством описанных правил. Подробно описан алгоритм работы каждого компонента.

Разработаны алгоритмы работы программного средства скрытой передачи данных между ПК без сетевого взаимодействия с использованием USB-носителей информации в качестве среды передачи данных с использованием разработанного протокола взаимодействия клиентских и серверных программных средств для передачи данных между ними.

Описан заголовок пакета, посредством которого происходит обмен данными между клиентскими и серверными программными средствами.

Данные методы и алгоритмы позволяют в дальнейшем реализовать макет программного средства для его проверки и тестирования на практике.

С помощью подобных алгоритмов можно безбоязненно скрывать информацию, и если при знании ключа такую систему взломать возможно не полным перебором (но весьма долгим алгоритмом), то без знания ключа данная система представляется абсолютно стойкой.

# Заключение

В ходе выполнения работы была разработана структура макета программного средства скрытой передачи данных между ПК с различными ОС (Windows, Linux) без сетевого взаимодействия. Для решения проблемы воздушного зазора в качестве среды передачи данных были использованы USB-носители информации.

Были проанализированы существующие методы и алгоритмы скрытой передачи данных между носителями информации и ПК.

Разработана структура клиент-серверного программного средства обеспечения скрытой передачи данных. Была выбрана модульная структура, состоящая из независимых компонентов, которые взаимодействуют между собой посредством описанных правил.

Помимо самого программного средства разработан протокол взаимодействия клиентских и серверных программных средств для передачи данных между ними. Описан заголовок пакета, посредством которого происходит однозначная идентификация запросов и ответов.

# Список использованных источников

1. Криминалистический анализ файловых систем для профессионалов / Кэрриэ Б.: Пер. с англ. СПб.: Питер, 2017. 470 с.
2. AntiMlware [электронный ресурс] URL:  https://www.anti-malware.ru/news/2021-12-03-114534/37653Web
3. Стеганография с использованием избыточности [электронный ресурс] URL: https://habr.com/ru/post/447028/ (дата обращения 21.10.2022)
4. Стеганография с использованием служебных таблиц файловой системы [электронный ресурс] URL: https://habr.com/ru/post/347604/ (дата обращения 23.10.2022)
5. Windows для профессионалов. Создание эффективных Win32-приложений с учетом специфики 64-разрядной версии Windows / Д. Рихтер.: Пер. с англ. Издательство «Русская Редакция»; 2008. 720 с.
6. Внутреннее устройство Microsoft Windows. М. Русинович, Д. Соломон, А Ионеску, П. Йосифович.: СПб.: Питер, 2018. 944 с.
7. Linux глазами хакера / М. Е. Фленов.: БХВ-Петербург, 2005. 544 с.