ТЕМА 7. Методы и средства защиты программного обеспечения

7.1. Классификация угроз ПО.

Необходимость использования систем защиты (СЗ) ПО обусловлена рядом проблем, среди которых следует выделить:

незаконное использование алгоритмов, являющихся интеллектуальной собственностью автора, при написании аналогов продукта (промышленный шпионаж);

несанкционированное использование ПО (кража и копирование);

несанкционированная модификация ПО с целью внедрения программных злоупотреблений;

незаконное распространение и сбыт ПО (пиратство).

7.2. Классификация систем защиты ПО

Таблица 7.1 Классификация систем защиты ПО

№п/п	По методу установки	По используемым механизмам защиты
1	Системы, устанавливаемые на скомпилированные модули ПО	Системы, использующие сложные логические механизмы
2	Системы, встраиваемые в исходный код ПО до компиляции	Системы, использующие шифрование защищаемого ПО;
3	Комбинированные системы	Комбинированные системы

Системы первого типа используют различные методы и приёмы, ориентированные на затруднение дизассемблирования, отладки и анализа алгоритма СЗ и защищаемого ПО.

Этот тип СЗ наименее стоек к атакам, так как для преодоления защиты достаточно проанализировать логику процедур проверки и должным образом их модифицировать.

Более стойкими являются системы второго типа.

Для дезактивации таких защит необходимо определение ключа дешифрации ПО.

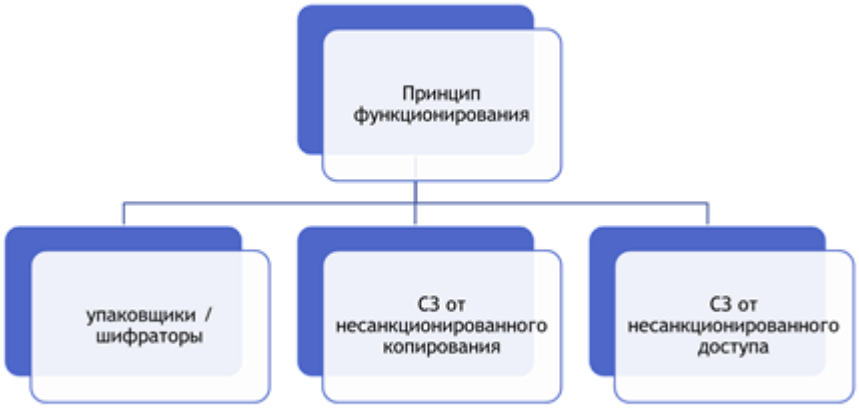

Рисунок 7.1. Классификация систем защиты по принципу функционирования

Упаковщики/шифраторы

Первоначально, использовались для уменьшения объема исполняемого модуля на диске, но позднее на первый план вышла цель защиты ПО от анализа его алгоритмов и несанкционированной модификации.

Для этих целей используются алгоритмы компрессии данных; приёмы, связанные с использованием недокументированных особенностей операционных систем (ОС), криптографические методы, алгоритмы мутации, запутывание логики программы.

Отрицательные стороны:

Замедляют выполнение кода ПО.

Шифрование/упаковка кода ПО вызывает затруднения при обновлении (update) и исправлении ошибок (bugfix, servicepack).

Данный класс систем уязвим, так как программный код, в конечном итоге, распаковывается или расшифровывается для выполнения.

Упаковка и шифрование исполняемого кода вступает в конфликт с запрещением самомодифицирующегося кода в современных ОС.

СЗ от несанкционированного копирования

СЗ от несанкционированного копирования осуществляют "привязку" ПО к дистрибутивному носителю (гибкий диск, CD ...).

При этом на физическом уровне создаётся дистрибутивный носитель, обладающий неповторимыми свойствами (нестандартной разметкой носителя, записи на него дополнительной информации - паролей или меток), а на программном уровне создаётся модуль, настроенный на аутентификацию носителя по его уникальным свойствам.

При этом возможно применение приёмов, используемых упаковщиками/шифраторами.

Положительные факторы:

Затруднение нелегального копирования и распространения ПО;

Защита прав пользователя на приобретённое ПО.

Отрицательные факторы:

Большая трудоёмкость реализации системы защиты;

Затруднение продаж из-за необходимости физической передачи дистрибутивного носителя;

Снижение отказоустойчивости ПО;

На время работы ПО занимается накопитель;

СЗ от НСД

СЗ от НСД осуществляют предварительную или периодическую аутентификацию пользователя ПО или его компьютерной системы путём запроса дополнительной информации.

К этому типу СЗ можно отнести:

системы парольной защиты ПО,

системы «привязки» ПО к компьютеру пользователя,

системы с «ключевыми дисками»

аппаратно-программные системы с электронными ключами.

Парольные защиты.

Парольные защиты осуществляют аутентификацию пользователя ПО путём запроса дополнительных данных, имя, пароль, серийный номер.

Эта информация запрашивается в различных ситуациях, например, при старте программы, по истечении срока бесплатного использования ПО, при вызове процедуры регистрации либо в процессе установки на ПК пользователя.

Положительные стороны:

очень просты в реализации

большинство парольных СЗПО использует логические механизмы, сводящиеся к проверке правильности пароля/кода и запуске или не запуске ПО, в зависимости от результатов проверки.

Отрицательные стороны:

если СЗПО не использует шифрования, достаточно принудительно изменить логику проверки для получения беспрепятственного доступа к ПО.

Системы "привязки" ПО

Системы этого типа при установке ПО на ПК пользователя осуществляют поиск уникальных признаков компьютерной системы либо сами устанавливают такие признаки.

(оценки скоростных и иных показателей процессора, параметров материнской платы, версий BIOS, ОС, запись скрытых файлов, реестр Windows. (скрытые места :ms, Filemon, Regmon))

Отрицательные факторы:

Ложные срабатывания СЗПО при любых изменениях в параметрах ПК.

Низкая стойкость при доступе злоумышленника к ПК пользователя.

Возможность конфликтов с системным ПО.

Программно-аппаратные средства защиты ПО с электронными ключами

Программно-аппаратные средства защиты, основанные на использовании так называемых «аппаратных (электронных) ключей», являются одними из самых стойких систем защиты ПО от НСД.

Электронный ключ - это аппаратная часть системы защиты, представляющая собой плату с микросхемами памяти либо с микропроцессором, помещенную в корпус и предназначенную для установки в один из стандартных портов ПК (COM, LPT, PCMCIA, USB ...) или слот расширения материнской платы.

Так же в качестве такого устройства могут использоваться SMART-карты.

Электронные ключи по архитектуре можно подразделить на:

ключи с памятью (без микропроцессора)

ключи с микропроцессором и памятью.

Ключи с памятью

Ключи с памятью являются менее стойкими, в таких системах критическая информация (ключ дешифрации, таблица переходов) хранится в памяти электронного ключа.

Для дезактивации таких защит в большинстве случаев необходимо наличие у злоумышленника аппаратной части системы защиты (перехват диалога между программной и аппаратной частями для доступа к критической информации), либо снятие логической защиты.

Системы с микропроцессором.

Самыми стойкими являются системы с микропроцессором.

Такие комплексы содержат в аппаратной части:

ключ дешифрации,

блоки шифрации/дешифрации данных.

При работе защиты в электронный ключ передаются блоки зашифрованной информации, и принимаются оттуда расшифрованные данные.

Так как все процедуры выполняются аппаратной частью, достаточно сложно перехватить ключ дешифрации.

Программно-аппаратные средства защиты ПО с электронными ключами.

Положительные факторы:

Значительное затруднение нелегального распространения и использования ПО;

Избавление производителя ПО от разработки собственной системы защиты;

Высокая автоматизация процесса защиты ПО;

Отрицательные факторы:

Дополнительные затраты на приобретение системы защиты и обучение персонала;

Замедление продаж из-за необходимости физической передачи аппаратной части;

Повышение системных требований из-за защиты (совместимость, драйверы);

Несовместимость защиты и аппаратуры пользователя;

Затруднения использования защищенного ПО в мобильных ПК.

7.3. Средства защиты от несанкционированного использования.

При защите программ от несанкционированного копирования применяются методы, которые позволяют привносить в защищаемую программу функции привязки процесса выполнения кода программы только на тех ЭВМ, на которые они были инсталлированы.

Инсталлированная программа для защиты от копирования при каждом запуске должна выполнять следующие действия:

анализ программно-аппаратной среды компьютера;

формирование на основе этого анализа текущих характеристик своей среды выполнения;

проверка подлинности среды выполнения путем сравнения ее текущих характеристик с эталонными;

блокирование дальнейшей работы программы при несовпадении текущих характеристик с эталонными.

Снятие защиты от копирования

Для снятия защиты от копирования применяют два основных метода: статический и динамический.

Статические методы предусматривают анализ текстов защищенных программ в естественном или преобразованном виде.

Динамические методы предусматривают слежение за выполнением программы с помощью специальных средств снятия защиты от копирования.

Основные методы защиты от копирования

Криптографические методы:

с использованием односторонней функции;

с использованием шифрования.

Функции инсталлятора при криптографическом методе защиты:

анализ программно-аппаратной среды компьютера;

формирование на основе анализа эталонных характеристик среды выполнения программы;

запись криптографически преобразованных эталонных характеристик программно-аппаратной среды компьютера на винчестер.

Эталонные характеристики среды заносятся в следующие области

жесткого диска:

в любые места области данных (в созданный для этого отдельный файл, в отдельные кластеры, которые должны помечаться затем в FAT как зарезервированные под операционную систему или дефектные);

в зарезервированные сектора системной области винчестера;

непосредственно в файлы размещения защищаемой программной системы, например, в файл настройки ее параметров функционирования.

Метод привязки к идентификатору формируемому инсталлятором используется если характеристики программно-аппаратной среды отсутствуют в явном виде или их определение значительно замедляет запуск программ или снижает удобство их использования.

На винчестере при инсталляции защищаемой от копирования программы формируется уникальный идентификатор, наличие которого затем проверяется инсталлированной программой при каждом ее запуске.

При отсутствии или несовпадении этого идентификатора программа блокирует свое дальнейшее выполнение.

Требования к уникальному идентификатору:

Идентификатор не должен копироваться стандартным способом;

Идентификатор целесообразно записывать в следующие области жесткого диска:

в отдельные кластеры области данных, которые должны помечаться затем в FAT как зарезервированные под операционную систему или как дефектные;

в зарезервированные сектора системной области винчестера.

Ключевой носитель – некопируемый стандартным образом идентификатор может помещаться на носитель, к которой должна будет обращаться при каждом своем запуске программа.

Такой носитель называют ключевым.

Защищаемая от копирования программа может быть привязана к уникальным характеристикам ключевого носителя.

Следует учитывать, что при использовании ключевого носителя значительно увеличивается неудобство пользователя, так как он всегда должен его вставлять перед запуском защищаемой от копирования программы.

Методы, основанные на работе с переходами и стеком

Данные методы основаны на включение в тело программы переходов по динамически изменяемым адресам и прерываниям, а также самогенерирующихся команд *(например, команд, полученных с помощью сложения и вычитания; использования вместо команды безусловного перехода (JMP) - возврата из подпрограммы (RET).)*

В стек записывается адрес перехода.

В процессе работы программы адрес модифицируется непосредственно в стеке.

При работе со стеком, стек определяется непосредственно в области исполняемых команд, что приводит к затиранию при работе со стеком.

Манипуляции с кодом программы.

При манипуляциях с кодом программы можно привести два следующих способа:

включение в тело программы "пустых" модулей;

изменение защищаемой программы.

Включение в тело программы "пустых" модулей

В тело программы включают модули, на которые имитируется передача управления, но реально никогда не осуществляется.

Эти модули содержат большое количество команд, не имеющих никакого отношения к логике работы программы.

«Ненужность» этих программ не должна быть очевидна потенциальному злоумышленнику.

Изменение защищаемой программы.

Изменяется начало защищаемой программы таким образом, чтобы стандартный дизассемблер не смог ее правильно дизассемблировать.

Например, такие программы, как Nota и Copylock, внедряя защитный механизм в защищаемый файл, полностью модифицируют исходный заголовок EXE-файла.

Все перечисленные методы были, в основном направлены на противодействия статическим способам снятия защиты от копирования.

Методы противодействия динамическим способам снятия защиты программ от копирования:

Периодический подсчет контрольной суммы;

Проверка количества свободной памяти;

Проверка незадействованных областей памяти;

Проверка содержимого векторов прерываний;

Переустановка векторов прерываний;

Чередование разрешения и запрещения прерывания;

Контроль времени выполнения частей программы.

Периодический подсчет контрольной суммы, занимаемой образом задачи области оперативной памяти, в процессе выполнения позволяет:

заметить изменения, внесенные в загрузочный модуль;

выявить контрольные точки, установленные отладчиком.

Проверка количества свободной и незадействованной памяти.

Проверка количества свободной памяти и сравнение и с тем объемом, к которому задача «привыкла» позволит застраховаться от грубой слежки с помощью резидентных модулей.

Проверка содержимого незадействованных для решения защищаемой программы областей памяти, которые не попадают под общее распределение

оперативной памяти, доступной для программиста, позволяет добиться "монопольного" режима работы программы.

Проверка содержимого векторов прерываний.

Проверка содержимого векторов прерываний (особенно 13h - *программное прерывание, программы BIOS управления дисками*; и 21h - *программное прерывание, вызов функций DOS*) на наличие тех значений, к которым задача "приучена".

Иногда бывает полезным сравнение первых команд операционной системы, отрабатывающих этим прерывания, с теми командами, которые там должны быть.

Вместе с предварительной очисткой оперативной памяти проверка векторов прерываний и их принудительное восстановление позволяет избавиться от большинства присутствующих в памяти резидентных программ.

Переустановка векторов прерываний.

Содержимое некоторых векторов прерываний (например, 13h и 21h) копируется в область свободных векторов.

Соответственно изменяются и обращения к прерываниям.

При этом слежение за известными векторами не даст желаемого результата.

Например, самыми первыми исполняемыми командами программы копируется содержимое вектора 21h (4 байта) в вектор 60h, а вместо команд int 21h в программе везде записывается команда int 60h.

В результате в явном виде в тексте программы нет ни одной команды работы с прерыванием 21h.

Чередование разрешения и запрещения прерывания.

Контроль времени выполнения частей программы.

Постоянное чередование команд разрешения и запрещения прерывания затрудняет установку отладчиком контрольных точек.

Контроль времени выполнения отдельных частей программы позволяет выявить "остановы" в теле исполняемого модуля.

7.4. Средства защиты ПО от обратного проектирования и модификации.

Под обратным проектированием (reverse engineering) понимают процесс исследования и анализа машинного кода, нацеленный на понимание общих механизмов функционирования программы, а также на его перевод на более высокий уровень абстракции (более высокий уровень языка программирования) вплоть до восстановления текста программы на исходном языке программирования.

Основными угрозами для программного продукта, защищённого от несанкционированного использования, являются угроза нарушения функциональности модуля защиты и угроза раскрытия эталонных

характеристик среды.

Реализация первой угрозы - обход либо полное отключение модуля защиты путём модификации кода программы.

Реализация второй – выяснение эталонных характеристик среды путём исследования программно-аппаратной среды функционирования защищённой программы.

Задачи, решаемые злоумышленником:

1. Задача локализации кода защитного механизма в коде программы.
2. Задача исследования модуля защиты и понимания принципов его действия.

Рисунок 7.2. Классификация средств обратного проектирования

Основные методы обратного проектирования.

Основными методами обратного проектирования являются отладка и дизассемблирование программ.

При этом используются следующие средства (инструменты):

Отладчики – программные средства, позволяющие выполнять программу в пошаговом режиме, контролировать ее выполнение, вносить изменения в ход выполнения. Позволяют проследить весь механизм работы программы и являются средствами динамического исследования работы программ.

Дизассемблеры – программные средства, позволяющие получить листинг программы на языке ассемблера, с целью его дальнейшего статического изучения. Дизассемблеры являются средствами статического исследования.

Мониторы событий – программные средства, позволяющие отслеживать определенные типы событий, происходящие в системе.

Редакторы кода. Как правило, включают функции дизассемблирования, но позволяют также вносить изменения в код программы. Предназначены, в основном, для исследования программ, занимающих небольшой объём.

Методы защиты ПО от обратного проектирования и модификации:

1) Алгоритмы запутывания;
2) Алгоритмы мутации;
3) Алгоритмы компрессии данных;
4) Алгоритмы шифрования данных;
5) Методы затруднения дизассемблирования;
6) Методы затруднения отладки;
7) Эмуляция процессоров и операционных систем;
8) Нестандартные методы работы с аппаратным обеспечением.

Алгоритмы запутывания - используются хаотические переходы в разные части кода, внедрение ложных процедур - "пустышек", холостые циклы, искажение количества реальных параметров процедур ПО, разброс участков кода по разным областям ОЗУ и т.п. (метод «спагетти»)

Запутывающие преобразования можно разделить на несколько групп в зависимости от того, на трансформацию какой из компонент программы они нацелены.

Преобразования форматирования, которые изменяют только внешний вид программы.

Преобразования структур данных, изменяющие структуры данных, с которыми работает программа.

Преобразования потока управления программы, которые изменяют структуру её графа потока управления.

Превентивные преобразования, использующие ошибки в определённых инструментальных средствах декомпиляции.

К преобразованиям форматирования относятся:

Удаление комментариев и переформатирование программы применимы, когда запутывание выполняется на уровне исходного кода программы. Эти преобразования не требуют только лексического анализа программы. При переформатировании программы исходное форматирование теряется безвозвратно, но программа всегда может быть переформатирована с использованием какого-либо инструмента для автоматического форматирования программ (например, indent для программ на Си).

Удаление отладочной информации приводит к тому, что имена локальных переменных становятся невосстановимы.

Изменение имён локальных переменных требует семантического анализа (привязки имён) в пределах одной функции.

Изменение имён всех переменных и функций программы помимо полной привязки имён в каждой единице компиляции требует анализа межмодульных связей. Имена, определённые в программе и не используемые во внешних библиотеках, могут быть изменены произвольным, но согласованным во всех единицах компиляции образом, в то время как имена библиотечных переменных и функций меняться не могут.

Преобразования структур данных, изменяющие структуры данных, с которыми работает программа.

К этой группе, например, относятся:

– преобразование, изменяющее иерархию наследования классов в программе,

– преобразование, объединяющее скалярные переменные одного типа в массив.

Преобразования потока управления программы изменяют структуру её графа потока управления. Они могут приводить к созданию в программе новых функций.

Открытая вставка функций:

Вынос группы операторов (function outlining);

Непрозрачные предикаты (opaque predicates);

Внесение недостижимого кода (adding unreachable code);

Внесение мёртвого кода (adding dead code);

Внесение избыточного кода (adding redundant code);

Преобразование сводимого графа потока управления к несводимому (transforming reducible to non-reducible flow graph);

Устранение библиотечных вызовов (eliminating library calls);

Переплетение функции (function interleaving);

Клонирование функций (function cloning);

Развёртка циклов (loop unrolling);

Разложение циклов (loop fission);

Реструктуризация графа потока управления;

Локализация переменных в базовом блоке;

Расширение области действия переменных.

Превентивные преобразования и Нестандартные методы работы с аппаратным обеспечением.

Превентивные преобразования, нацеленные против определённых методов декомпиляции программ или использующие ошибки в определённых инструментальных средствах декомпиляции.

Нестандартные методы работы с аппаратным обеспечением - модули системы защиты обращаются к аппаратуре ЭВМ, минуя процедуры операционной системы, и используют малоизвестные или недокументированные её возможности.

Алгоритмы мутации и Алгоритмы компрессии данных.

Алгоритмы мутации – создаются таблицы соответствия операндов – синонимов и замена их друг на друга при каждом запуске программы по определенной схеме или случайным образом, случайные изменения структуры программы.

mov op1,op2 = push op2

pop op1
jmp addr = push addr
ret
call addr = push m
jmp addr
m: . . .

Алгоритмы компрессии данных – программа упаковывается, а затем распаковывается по мере выполнения.
EXEPACK, zLib.

Алгоритмы шифрования данных

Алгоритмы шифрования данных – программа шифруется, а затем расшифровывается по мере выполнения.
(Полная – частичная расшифровка).

Защищаемый участок кода шифруется каким-либо алгоритмом

В программу добавляется модуль дешифрования, который в нужный момент дешифрует его и передаст ему управление.

Защищаемый участок будет воспринят дизассемблером неверно.

На выходе дизассемблера будет сформирован «мусор».

Данный метод при неправильном использовании достаточно уязвим ввиду того, что алгоритм расшифровки доступен взломщику, необходимо лишь найти его и произвести расшифровку.

Поэтому шифрование должно производиться на секретном ключе, который доступен только легальному пользователю и никому другому.
Методы затруднения дизассемблирования

Методы затруднения дизассемблирования – используются различные приемы, направленные на предотвращение дизассемблирования в пакетном режиме. (IDA Scripts)

Шифрование кода программы.

Сокрытие команд передачи управления.

Использование косвенной передачи управления.

Использование нестандартных способов передачи управления.

Осуществление короткого перехода вперёд, а между командой перехода и адресом перехода добавление «мусора».

Методы затруднения отладки – используют различные приемы, направленные на усложнение отладки программы. (Завешивание отладчика – обнаружение отладчика)

Противодействие отладке затрудняет злоумышленнику трассировку программы, а значит локализацию кода модуля защиты и его исследование.

Методы борьбы с отладчиками можно подразделить на следующие группы:

Использование триков (ловушек) для отладчиков, с помощью которых

возможно выявить наличие последнего в оперативной памяти, и прекратить работу программы, либо затруднить процесс отладки.

Выявление наличия отладчика в оперативной памяти и последующая его нейтрализация, используя различные «дыры», допущенные при реализации отладчиков, либо внедренные разработчиком отладчика сознательно.

Использование недокументированных команд и недокументированных возможностей процессора.

Эмуляция процессоров и операционных систем

Эмуляция процессоров и операционных систем – создается виртуальный процессор и/или операционная система (не обязательно реально существующие) и программа-переводчик из системы команд IBM в систему команд созданного процессора или ОС.

После такого перевода ПО может выполняться только при помощи эмулятора, что резко затрудняет исследование алгоритма ПО.

7.4. Водяные знаки

Чтобы компьютерный файл, представляющий собой объект авторского права, не мог быть изменен без ведома автора, чтобы он содержал всю необходимую информацию о правомерном использовании, применяются стеганографические вставки, или цифровые водяные знаки (ЦВЗ).

Если произведение подвергается какому-то изменению, то вместе с ним изменяется и видимый водяной знак. ЦВЗ представляют собой специальные метки, внедряемые в файл, в цифровое изображение или цифровой сигнал в целях контроля их правомочного использования.

ЦВЗ делятся на два типа — видимые и невидимые. ЦВЗ используются для защиты от копирования, сохранения авторских прав.

Невидимые водяные знаки считываются специальным устройством, которое может подтвердить либо опровергнуть корректность. ЦВЗ могут содержать различные данные: авторские права, идентификационный номер, управляющую информацию.

Наиболее удобными для защиты с помощью ЦВЗ являются неподвижные изображения, аудио и видео файлы.

Требования к ЦВЗ.

Видимые ЦВЗ довольно просто удалить или заменить.

Для этого могут быть использованы графические или текстовые редакторы.

Невидимые ЦВЗ представляют собой встраиваемые в компьютерные файлы вставки, не воспринимаемые человеческим глазом или ухом.

Поэтому ЦВЗ должны отвечать следующим требованиям:

незаметность;

робастность к обработке сигналов (робастность — способность системы к восстановлению после воздействия на неё внешних/внутренних искажений, в

том числе умышленных);

индивидуальность алгоритма нанесения (достигается с помощью стеганографического алгоритма с использованием ключа);

возможность для автора обнаружить несанкционированное использование файла;

невозможность удаления неуполномоченными лицами;

устойчивость к изменениям носителя-контейнера (к изменению его формата и размеров, к масштабированию, сжатию, повороту, фильтрации, введению спецэффектов, монтажу, аналоговым и цифровым преобразованиям).

ЦВЗ имеют небольшой объём, но для выполнения указанных выше требований, при их встраивании используются более сложные методы, чем для встраивания обычных заголовков или сообщений.

Такие задачи выполняют специальные стегосистемы.

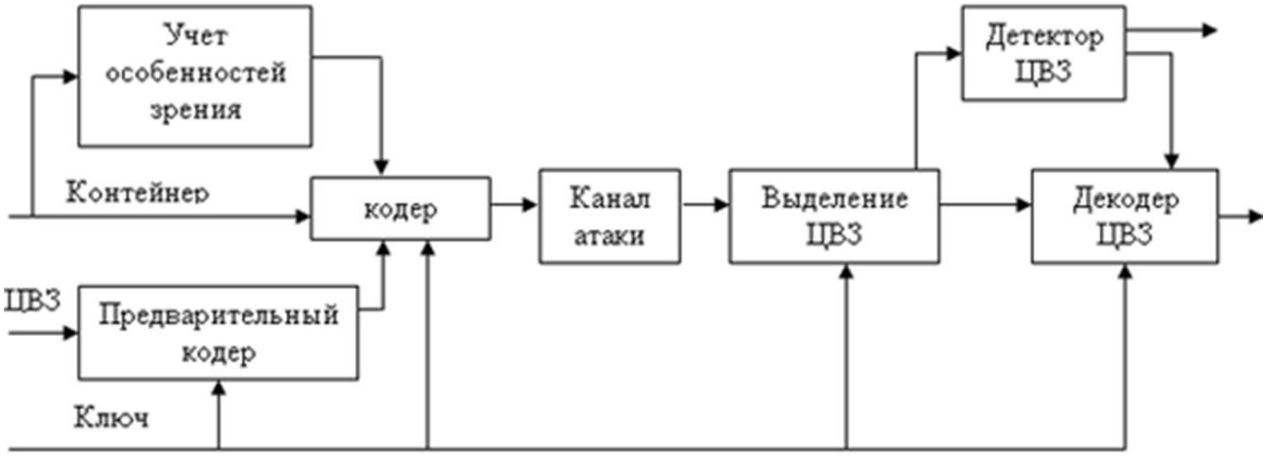

Рисунок 7.3. Схема типичной стегосистемы.

прекодер – устройство, предназначенное для преобразования скрываемого сообщения к виду, удобному для встраивания в сигнал-контейнер (контейнером называется информационная последовательность, в которой прячется сообщение);

стегокодер – устройство, предназначенное для осуществления вложения скрытого сообщения в другие данные с учетом их модели;

устройство выделения встроенного сообщения;

стегодетектор – устройство, предназначенное для определения наличия стегосообщения;

декодер – устройство, восстанавливающее скрытое сообщение.

Типы контейнеров.

Существенное влияние на надежность и устойчивость стегосистемы, а также возможность обнаружения факта передачи скрытого сообщения оказывает выбор контейнера.

Контейнеры можно подразделить на два типа:

непрерывные(потоковые);

фиксированной длинны;

Особенности потокового контейнера – невозможно определить его начало или конец, нет возможности узнать заранее, какими будут последующие шумовые биты, скрывающие биты выбираются с помощью специального генератора, задающего расстояние между последовательными битами в потоке.

Для получателя – возможность определить, когда начинается скрытое сообщение – сигналы синхронизации.

Типы контейнеров.

Особенности контейнеров фиксированной длины:

Отправитель заранее знает размер файла и может выбрать скрывающие биты в подходящей псевдослучайной последовательности. Иногда встраиваемое сообщение может не поместиться в файл-контейнер. Расстояния между скрывающими битами равномерно распределены между наиболее коротким и наиболее длинным заданными расстояниями, в то время как истинный случайный шум будет иметь экспоненциальное распределение длин интервала.

На практике чаще всего используются именно контейнеры фиксированной длины, как наиболее распространенные и доступные.

Схема данных в мультимедийных файлах

Встраивание информации в мультимедийные файлы можно представить следующим рисунком:

Рисунок 7.4. Встраивание информации в мультимедийные файлы

Стеганографические методы, применяемые для встраивания информации в видео, сжатое согласно стандарту MPEG требуют работы в реальном времени и должны обладать малым количеством вычислений. Поток видеоданных в MPEG имеет иерархическую структуру. Низший уровень состоит из блоков яркости и цветности. Таким образом, оперируя блоками яркости и цветности можно достичь встраивания ЦВЗ.

Методы встраивания информации.

Существует много методов встраивания информации в изображения:

– встраивание информации на уровне коэффициентов;

– встраивание информации на уровне битовой плоскости;

– встраивание информации за счет энергетической разности между коэффициентами.

Наиболее оптимальным является алгоритм встраивание информации на уровне коэффициентов. Этот метод требует только кодирование на уровне кодирование Хаффмана, кодирование длин серий и квантовании.

Алгоритм внедрения ЦВЗ в статическое изображение.

24-битное изображение формата JPEG или BMP имеет RGB-кодировку.

Встраивание выполняется в канал синего цвета, так как к синему цвету система человеческого зрения наименее чувствительна.

Количество цветов приблизительно 16 млн.,

Человеческий глаз улавливает в зависимости от 1 до 4 млн цветов.

Общую схему цветности можно схематически описать так:

R(красный) - человеческий глаз улавливает 7 бит из 8;

G(зеленый) - человеческий глаз улавливает 8 бит из 8;

B(синий) - человеческий глаз улавливает 4 бит из 8.

Соответственно к синему цвету человеческий глаз наименее восприимчив. Это преимущество позволяет встраивать информации в области синего цвета. Но этим свойством могут воспользоваться и оптимизаторы размера картинки. Можно обрезать неиспользуемые биты, а точнее невидимые для глаза человека.

Рассмотрим алгоритм передачи одного бита секретной информации.

1) Пусть si – встраиваемый бит I = {R,G,B} – контейнер, p = (x,y) - позиция, в которой выполняется вложение.

2) Секретный бит встраивается в канал синего цвета путем модификации яркости с помощью q - константы, определяющей энергию встраиваемого сигнала.

3) Ее величина зависит от предназначения схемы. Чем больше q, тем выше робастность вложения, но тем сильнее его заметность.

Алгоритм извлечения ЦВЗ

1) Извлечение бита получателем осуществляется без наличия исходного изображения.

2) Выполняется предсказание значения исходного, немодифицированного пикселя на основании значений его соседей.

3) Для получения оценки пикселя можно использовать значения 2 пикселей, расположенных справа и слева от оцениваемого в той же строке.

4) При использовании исходного файла значение встроенного бита определяется знаком разности значений для закодированного и незакодированного изображения.