

А.М. Голиков

**МЕТОДЫ ШИФРОВАНИЯ ИНФОРМАЦИИ В СЕТЯХ И
СИСТЕМАХ РАДИОСВЯЗИ**

**Сборник лабораторных работ по курсу: «Кодирование и
шифрование информации в системах связи»**

Томск

Министерство образования и науки РФ

Томский государственный университет систем управления
и радиоэлектроники

УТВЕРЖДАЮ
Заведующий кафедрой РТС

_____ Г.С.Шарыгин

МЕТОДЫ ШИФРОВАНИЯ ИНФОРМАЦИИ В СЕТЯХ И СИСТЕМАХ РАДИОСВЯЗИ

Сборник лабораторных работ по курсу: «Кодирование и шифрование
информации в системах связи»
специальности 21065-2.65 – Радиоэлектронные системы и комплексы передачи
информации

Разработчик
доцент кафедры РТС

_____ А.М.Голиков

2012

Голиков А.М. Методы шифрования информации в сетях и системах радиосвязи: Сборник лабораторных работ. – Томск: Томск. гос. ун-т систем управления и радиоэлектроники, 2012. – 333 с.

Сборник содержит описания лабораторных работ по курсу «Кодирование и шифрование информации в системах связи» специальности 21065-2.65 – Радиоэлектронные системы и комплексы передачи информации. Представлены описания аппаратно-программных комплексов и методики выполнения лабораторных работ. В разработке аппаратно-программных комплексов принимали участие студенты ТУСУР.

ОГЛАВЛЕНИЕ

Лабораторная работа 1. Исследование методов скремблирования информации в сетях и системах радиосвязи

1. Цель работы
2. Краткие теоретические сведения
3. Порядок выполнения работы
4. Рекомендуемая литература

Лабораторная работа 2. Исследование методов стеганографии информации в сетях и системах радиосвязи

1. Цель работы
2. Краткие теоретические сведения
3. Порядок выполнения работы
4. Рекомендуемая литература

Лабораторная работа 3. Исследование отечественных алгоритмов шифрования информации ГОСТ 28147-89

1. Цель работы
2. Краткие теоретические сведения
3. Порядок выполнения работы
4. Рекомендуемая литература

Лабораторная работа 4. Исследование международных алгоритмов шифрования информации AES

1. Цель работы
2. Краткие теоретические сведения
3. Порядок выполнения работы
4. Рекомендуемая литература

Лабораторная работа 5. . Исследование отечественных стандартов хэш-функции (ГОСТ Р 34.11-94) и электронной цифровой подписи (ЭЦП ГОСТ Р 34.10-2001)

1. Цель работы
2. Краткие теоретические сведения
3. Порядок выполнения работы
4. Рекомендуемая литература

Лабораторная работа 6. Исследование алгоритмов защиты данных с помощью PGP

1. Цель работы
2. Краткие теоретические сведения
3. Порядок выполнения работы
4. Рекомендуемая литература

Лабораторная работа 1. Исследование методов скремблирования информации в сетях и системах радиосвязи

1. Цель работы

Ознакомиться с методами цифровой обработки сигналов с использованием dsp-скремблеров, схемами получения зашифрованных сигналов и исследованием их свойств. Изучить принципы работы цифровых сигнальных процессоров, а также основы их программирования.

2. Краткие теоретические сведения

С развитием вычислительной техники и информационных технологий увеличивается сложность систем защиты компьютерной информации. Финансовый кризис обостряет конкуренцию в различных секторах рынка и выводит на первый план вопросы информационной безопасности даже для тех компаний, которые раньше, в более спокойных условиях, не придавали этим вопросам особого значения.

В данном обзоре предпринята попытка сопоставить базовые функции отечественных аппаратных устройств защиты информации (УЗИ) на основе сведений, предоставляемых фирмами-производителями, и выработать рекомендации по выбору устройства.

Преимущественный интерес к отечественным продуктам обусловлен тем, что применение зарубежных средств встречает значительные трудности. С одной стороны, их вывоз из страны происхождения зачастую вообще невозможен: так, в США, которые наряду с Россией остаются лидером мировой криптографии, наложен запрет на экспорт средств шифрования с “сильными” ключами. С другой стороны, легальное применение зарубежных криптографических средств в нашей стране жестко регламентируется, и в этих

условиях оправданным представляется сосредоточить внимание на отечественных продуктах.

Как правило, аппаратные средства уступают чисто программным решениям по ряду параметров (скорости шифрования, возможности применения в компьютерах различных конструктивных исполнений — например, ноутбуках), но превосходят их по главному показателю — стойкости. Кроме того, нельзя полностью гарантировать, что в применяемых операционных системах (ОС) и прикладных программах отсутствуют неописанные возможности, невыявленные ошибки или умышленно размещенные “закладки”. Потенциально возможностями их модификации обладают не только активные злоумышленники, но даже обычные пользователи компьютеров: имея навыки программирования и администрирования и доступ к компьютерной системе, они способны модифицировать эту систему, снижая уровень ее надежности.

Для обеспечения должной безопасности следует либо заниматься анализом и аттестацией существующих систем на уровне их исходных текстов, либо разрабатывать собственную, полностью контролируемую ОС и постоянно заботиться о поддержании ее целостности. Чаще всего пользователи выбирают третий путь, отвечающий их реальным возможностям: принимают используемую ОС за контролируемую и — в лучшем случае — стремятся обеспечить ее целостность имеющимися в наличии средствами (антивирусными пакетами и т.п.).

Однако проверку целостности одних программ при помощи других нельзя признать надежной, поскольку под сомнение ставится целостность самой программы проверки целостности. Для получения достоверных сведений необходима некая начальная “точка отсчета”, подлинность которой не вызывает сомнений.

В подобной роли и могут выступить рассматриваемые устройства, применяемые в составе систем защиты информации (СЗИ) и позволяющие обеспечить целостность таких систем аппаратной реализацией функций,

критически важных для поддержания надежности СЗИ. К числу таких функций, полностью или частично выполняемых аппаратными средствами, относятся следующие.

Запрет на модификацию процесса загрузки компьютера.

Идентификация и аутентификация пользователя до загрузки ОС.

Контроль целостности операционной системы и прикладного программного обеспечения (ПО).

Управление доступом пользователя к ресурсам компьютера.

Ведение журнала действий пользователя.

Реализация криптографических алгоритмов, гарантия их стойкости и неизменности.

Надежное хранение секретных криптографических элементов (ключей) вне оперативного запоминающего устройства (ОЗУ) компьютера.

Загрузка секретных элементов со специальных носителей: смарт-карт, идентификаторов Touch Memory (TM) и т.п. К примеру, смарт-карты по сравнению с традиционными носителями (дискетами) обладают большей стойкостью к механическим, электрическим, магнитным, климатическим воздействиям, более долговечны (срок хранения данных — до 10 лет) и удобны в пользовании, а микропроцессорные карты, кроме того, обладают еще и высокой степенью защищенности от несанкционированного использования.

Наличие аппаратного датчика случайных чисел. Аппаратный датчик, использующий физический процесс (например, обратный пробой специализированного диода), обеспечивает распределение случайных чисел, наиболее близкое к равновероятному (что недоступно программным датчикам). Таким образом, при формировании криптографических ключей с помощью аппаратного датчика обеспечивается максимальный уровень их надежности.

Полностью реализовать все перечисленные функции одним только аппаратным способом невозможно (а часто и не нужно), поэтому ряд функций может выполняться программно. Программное обеспечение может находиться в

постоянном запоминающем устройстве (ПЗУ) УЗИ, либо размещаться на обычных носителях (в этом случае защищать его целостность необходимо организационными методами — особым режимом хранения, ограничением числа лиц, имеющих доступ к этому ПО, и т.д.). В любом случае, ПО должно опираться на возможности, предоставляемые устройством защиты информации.

Скремблирование аналоговых сигналов

Преобразования с инверсией спектра и статическими перестановками спектральных компонент речевого сигнала

Схемотехническая реализация двух рассматриваемых вариантов заметно отличается, что и обуславливает их отдельное рассмотрение. Однако с точки зрения достигаемых результатов по защищенности сигнала в канале связи оба варианта аналогичны.

Процесс инверсии спектра сигнала при передаче и его восстановления при приеме иллюстрируется на рисунке 1.1.

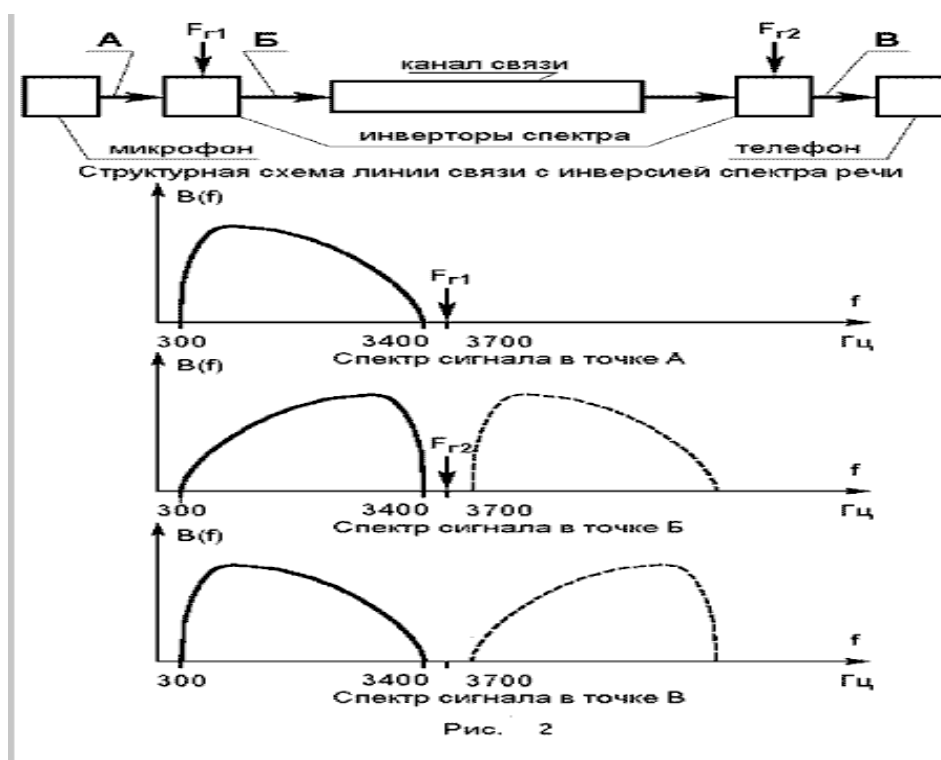


Рис. 1.1. Процесс инверсии спектра

Схема инвертора представляет собой балансный смеситель. При частоте гетеродина F_g , равной сумме граничных частот F_n и F_b преобразуемого сигнала (3700 Гц для стандартного телефонного канала с $F_n = 300$ Гц и $F_b = 3400$ Гц) нижняя полоса частот после смесителя воспроизводится в исходной полосе частот, т.е. в полосе канала в инверсном виде. При приеме производится повторная инверсия и исходный сигнал восстанавливается.

Качество восстановленной речи зависит от качества (на передающей и на приемной сторонах) смесителей, фильтров, ограничивающих спектр входного сигнала и выделяющих нижнюю полосу частот преобразованного сигнала, а также от коррекции на приемной стороне частотных искажений канала, влияние которых также сказывается инверсно: затухание канала в высокочастотной части спектра на приеме сказывается в низкочастотной части сигнала и наоборот.

При перехвате сигнал с инвертированным спектром может быть легко восстановлен любым аналогичным аппаратом (не обязательно однотипным), а при соответствующей тренировке — воспринят человеком непосредственно.

Для повышения стойкости защиты некоторые изготовители вводят переменную частоту гетеродина, устанавливаемую партнерами по договоренности в форме числового кода-пароля, вводимого в аппарат при переходе в защищенный режим.

Возможности такого дополнительного частотного сдвига, приводящего к несовпадению спектра передаваемого сигнала и номинальной частотной полосы канала связи и, соответственно, к ухудшению качества восстановленной речи, ограничены несколькими сотнями герц. Достижимый эффект весьма условен. Действительно, при прослушивании восстановленного сигнала, в случае неравенства частот гетеродинов на передаче и на приеме, в первый момент возникает ощущение неестественной и непонятной речи, которое, однако, почти не мешает воспринимать ее смысл после некоторой адаптации.

Процесс преобразования с фиксированными перестановками спектральных компонент речевого сигнала при передаче и его восстановления при приеме иллюстрируется на рисунке 1.2.

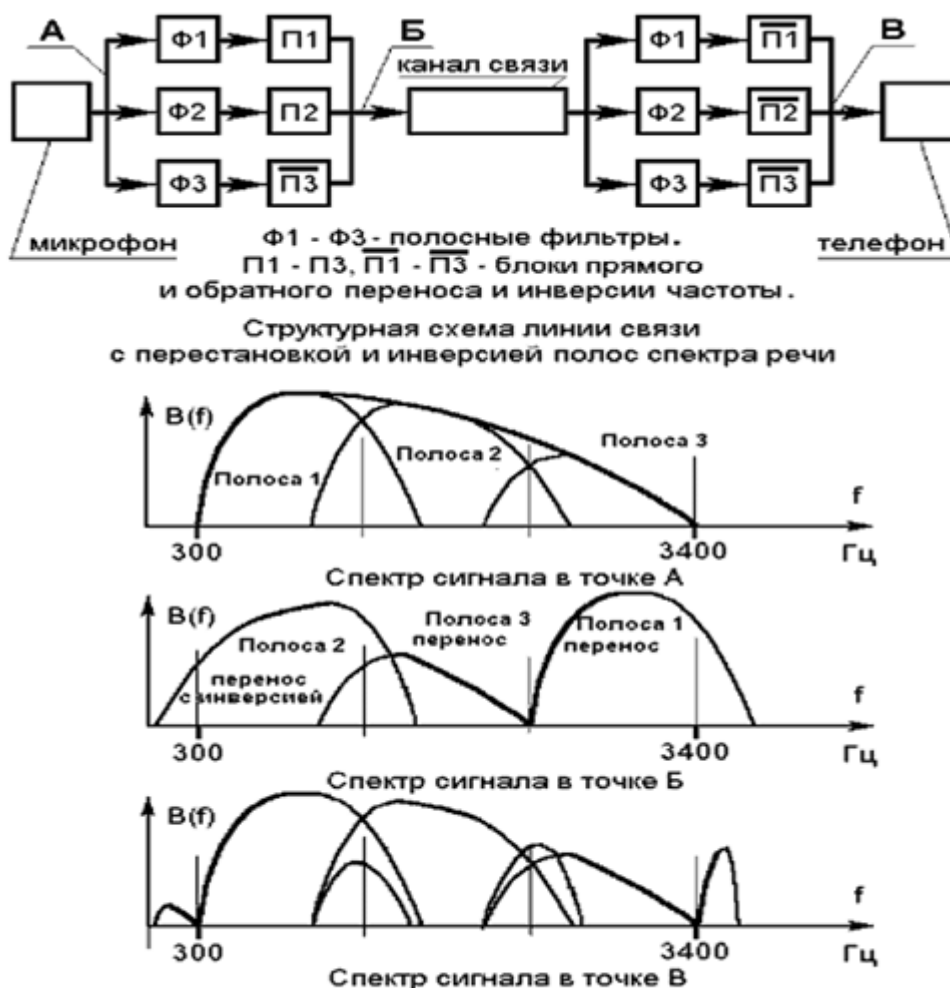


Рис. 1.2. Преобразование с фиксированными перестановками

При таком преобразовании разборчивость речевого сигнала нарушается в значительно большей степени, чем при простой инверсии. Следует, однако, учитывать, что выбор вариантов частотных перестановок весьма ограничен. Фильтры, выделяющие частотные полосы в исходном и в линейном сигнале, имеют конечную крутизну характеристики, в результате чего на заметном частотном интервале в окрестности границы частотных полос будет

происходить заметное невосстановимое смешение различных компонент сигнала. Полная полоса частот (300-3400 Гц) составляет 3,5 октавы. При формировании трех полос (по 1,2 октавы на каждую полосу) и при использовании фильтров 8-го порядка (нарастание затухания около 48 дБ/октаву) затухание в середине (!) соседней полосы составит не более 30 дБ, что предопределяет низкое качество восстановленной речи. Существенное увеличение порядка фильтров настолько усложняет аппаратуру, что она теряет преимущества перед другими вариантами преобразователей. В тоже время число возможных перестановок из трех полос — всего лишь 6, из четырех полос — 24, т.е. даже в условиях прямого перехвата, не говоря уже об анализе записи, подбор нужной подстановки не составит труда.

Наиболее существенным положительным качеством рассматриваемых преобразователей является их автономность, т.е. отсутствие необходимости во взаимной синхронизации передающего и приемного аппарата и, соответственно, отсутствие задержки связи на время проведения синхронизации и возможных срывов защищенного режима из-за качества канала, недостаточного для проведения синхронизации. Если удалось установить связь в открытом режиме после включения партнерами инверторов будет реализован и защищенный режим.

Положительными качествами такой аппаратуры также являются:

- дешевизна (цены инверторов спектра порядка 30 — 50 USD);
- возможность построения схем, не вносящих задержку сигнала;
- малая критичность к качеству используемого канала связи и предельная простота в управлении.

Аппаратура может включаться между телефонным аппаратом и линией в стандартный двухпроводной стык между телефонным аппаратом и микротелефонной трубкой, может использоваться в виде накладки на микротелефонную трубку с акустической передачей преобразованного сигнала. Переход в защищенный режим происходит по взаимной договоренности

партнеров после установления соединения. Переход происходит немедленно после нажатия соответствующей клавиши (или другого управляющего действия). Включение и выключение защищенного режима осуществляется каждым партнером самостоятельно, синхронизация действий не требуется.

При разговоре в линии прослушивается характерный сигнал, по структуре полностью повторяющий передаваемую речь. Восстановленный сигнал имеет высокое качество. В дешевых аппаратах с недостаточной фильтрацией возможно наличие свистящих тонов и изменение тембра голоса говорящего. Наличие посторонних шумов в помещении, из которого ведется передача, сказывается на качестве восстановленного сигнала так же, как в открытом режиме, на стойкость защитного преобразования почти не влияет.

Преобразования с временными перестановками (скремблированием) и временной инверсией элементов речевого сигнала со статическим законом перестановки

Принцип работы аппаратуры сходен с разрушением и последующим восстановлением мозаичной картины, что обусловило появление названия “аппаратура мозаичных преобразований”.

Данный класс аппаратуры требует наличия в своем составе блока запоминания сигнала с управляемым доступом по записи и считыванию, поэтому аналоговой такую аппаратуру можно назвать условно. Временная перестановка элементарных отрезков речевого сигнала и восстановление их последовательности на приеме занимают соответствующий интервал времени. Поэтому обязательным свойством такой аппаратуры является заметная задержка сигнала на приемной стороне. Процессы преобразования сигнала показаны на рисунке 1.3.

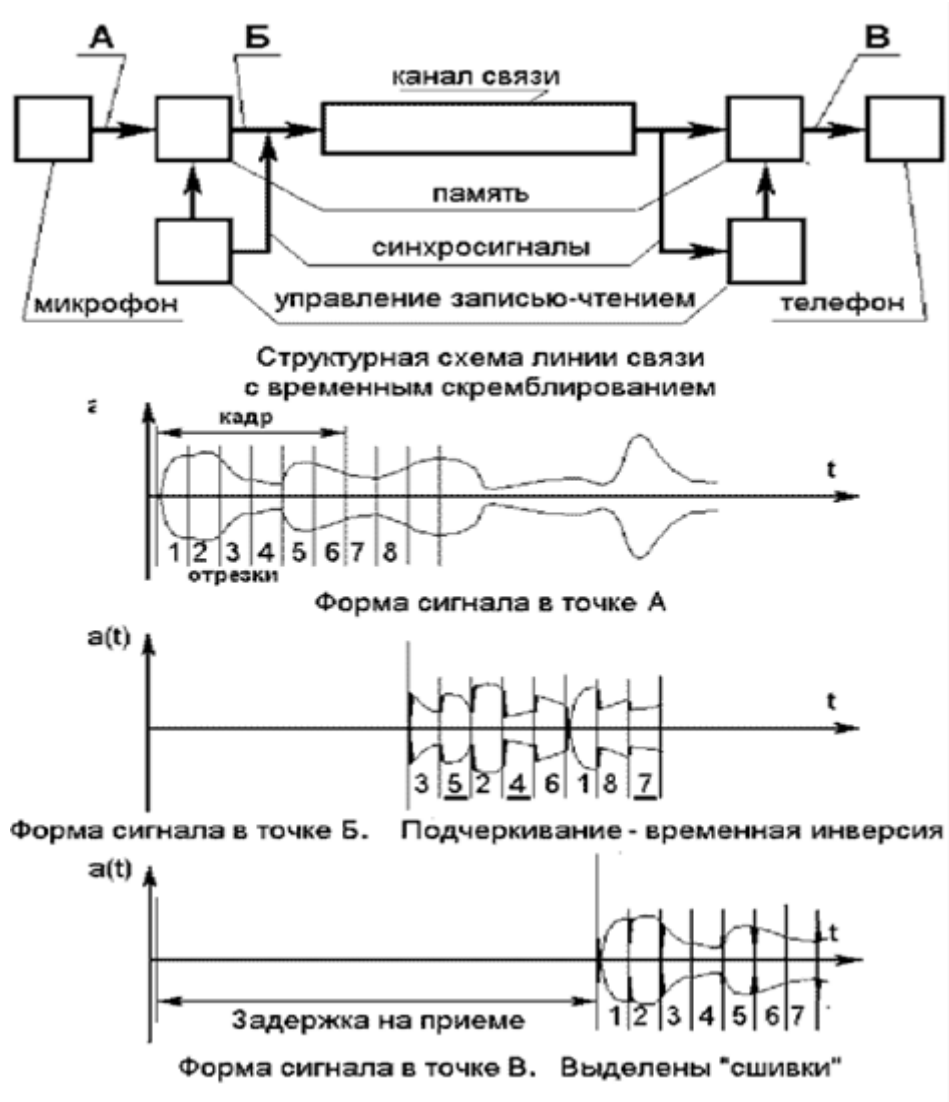


Рис. 1.3. Процесс преобразования сигнала

Чем меньше длительность элементарных отрезков, на которые разбивается исходный речевой сигнал и чем больше элементов участвуют в операции перестановки, тем сложнее процесс восстановления речи по перехваченному линейному сигналу.

Однако при передаче по каналу связи возникают краевые искажения элементарных отрезков. При восстановлении речи на приемной стороне это приводит к появлению “сшивок”, ухудшающих качество восстановленного сигнала. С учетом характеристик реальных телефонных каналов длительность

элементарных отрезков сигнала ограничена снизу на уровне 15 — 20 миллисекунд.

Увеличение числа перемешиваемых элементов мозаики — увеличение “глубины перестановки” — ограничено возрастанием задержки восстановленного сигнала на приеме. При диалоге заметные неудобства возникают при задержке более 0,3 сек, а при задержке более 1 сек диалог становится невозможным. Оба указанных фактора определяют глубину перестановки на уровне 16 — 64 элементарных отрезков речи.

Маскирующее воздействие на структуру сигналов в линии связи может быть достигнуто временной инверсией (воспроизведением в обратном направлении по отношению к записи) всех или отдельных отрезков. Такое преобразование неэффективно на коротких отрезках (с продолжительностью менее длительности одного элементарного звука речи). Применение длинных отрезков уменьшает возможность их перемешивания. Поэтому временная инверсия применяется исключительно как дополнительное преобразование в комбинации с временными перестановками. При этом наиболее эффективна временная инверсия всех отрезков.

Временные перестановки и временная инверсия при правильном выборе параметров перестановки исключают непосредственное прослушивание речи в канале связи, но при анализе записи или при оперативном анализе сигнала на месте перехвата статическая перестановка, повторяющаяся из кадра в кадр, легко выявляется по спектральным и амплитудным связям отрезков, в результате чего исходная речь может быть восстановлена с применением несложной аппаратуры (ПЭВМ с аудиоплатой).

В то же время по своему составу и сложности алгоритма аппаратура с фиксированными перестановками незначительно отличается от аппаратуры с переменными перестановками, управляемыми криптоблоком. Поэтому в настоящее время для цепей защиты информации применяются почти исключительно аппараты с переменными перестановками.

Преобразования с временными или частотными перестановками (скремблированием) с переменными перестановками под управлением криптоблока и комбинированные мозаичные преобразования

Применение переменных перестановок позволяет значительно затруднить восстановление исходной речи по перехвату сигнала в канале. При правильном выборе криптоалгоритма удачный подбор перестановки на одном интервале никак не способствует подбору перестановок на последующих интервалах. Кроме того, введение криптоалгоритма с индивидуальным ключом исключает возможность использования для перехвата однотипного аппарата.

Аппаратура строится, как правило, на базе сигнальных процессоров, имеет в своем составе АЦП, ЦАП, криптоблок управления перестановкой, систему ввода или формирования ключа. Обязательным этапом рабочего процесса является начальная синхронизация взаимодействующих аппаратов и их последующая подсинхронизация.

Как следствие, эта аппаратура заметно дороже аппаратуры частотной инверсии — 200 - 400 USD за единицу.

При переходе в защищенный режим по договоренности абонентов возникает интервал прерывания речевой связи, который занимает процесс синхронизации и установления взаимодействия криптоблоков. В ряде изделий в это же время абонент, используя клавиатуру телефонного аппарата или клавиатуру скремблера, или персональный узел памяти, должен ввести ключ. В результате переход в защищенный режим может занимать до 10 — 20 секунд. При этом надо учитывать, что при плохом качестве канала синхронизация и переход в защищенный режим могут не состояться, хотя связь в открытом режиме, пусть и при плохом качестве, поддерживается.

Наличие временной задержки при передаче сигнала при работе по двухпроводной линии неизбежно приводит к возникновению “эха” (это же характерно и для статических временных перестановок). В современной

аппаратуре связи отработаны весьма совершенные алгоритмы подавления эха, широко применяемые в скоростных модемах. Однако человеческое ухо реагирует на уровни эхо-сигналов, заведомо несущественные для модемов. Поэтому даже в наиболее удачных моделях скремблеров подавление эха до не замечаемого абонентом уровня достигается только при случайном удачном сочетании параметров линии связи.

Криптоблок, управляющий процессом перестановок, может использовать как симметричную, так и несимметричную (“с открытым ключом”) ключевую систему. Варианты с несимметричной системой предпочтительнее, так как упрощают эксплуатационный процесс и исключают вскрытие записи при хищении личного ключа. Однако и в этом случае применение личного пароля полезно, так как исключает вхождение в связь посторонних лиц.

Учитывая то вышеуказанное обстоятельство, что при самом совершенном криптоалгоритме передаваемая речь может быть восстановлена по перехвату линейного сигнала по остаточным признакам взаимного расположения элементарных отрезков, применение в скремблерах очень мощных криптоалгоритмов и ключевых кодов большой длины не оправдано. Вполне достаточной является длина ключевого кода порядка 9 десятичных (30 двоичных) знаков в симметричной ключевой системе и 30 десятичных (около 100 двоичных) — в несимметричной ключевой системе. При разговоре в линии прослушивается характерный “рваный” сигнал, в котором достаточно легко определяется структура передаваемой речи. Восстановленный сигнал имеет высокое качество, мало отличающееся от качества речи в открытом режиме на том же канале. Наличие посторонних шумов в помещении, из которого ведется передача, сказывается на качестве восстановленного сигнала так же, как в открытом режиме. Однако ритмические помехи, создающие “шкалу времени” параллельную преобразуемому сигналу, могут повлиять на стойкость защитного преобразования. Аппаратура может включаться между телефонным аппаратом и линией в стандартный двухпроводной стык, между телефонным аппаратом и

микротелефонной трубкой, может использоваться в виде накладки на микротелефонную трубку с акустической передачей преобразованного сигнала.

Таким образом, основными положительными качествами аппаратуры мозаичных преобразований — скремблеров — являются:

- относительно высокая стойкость защиты передаваемого речевого сигнала, исключающая его непосредственное прослушивание даже при наличии группы высокотренированных аудиторов и требующая для восстановления речи значительных затрат времени при использовании специализированных измерительно-вычислительных комплексов, применяемых государственными спецслужбами;
- относительно низкая стоимость;
- простота эксплуатации (для моделей, специально разработанных для непрофессионального пользователя).

К недостаткам данного класса аппаратуры следует отнести:

- задержку восстановленного сигнала на приемной стороне, требующую привыкания и затрудняющую диалог;
- наличие эха, зависящего от параметров коммутируемой линии связи;
- задержку связи на время прохождения процесса синхронизации аппаратов;
- возможность срыва синхронизации на плохих каналах.

По совокупности качеств этот класс аппаратуры представляется наиболее приемлемым для использования в корпоративных системах защищенного обмена речевой информацией оперативного характера, не требующей длительного периода секретности.

Скремблирование цифровых сигналов

Суть скремблирования заключается в побитном изменении проходящего через систему потока данных. Практически единственной операцией, используемой в скремблерах является XOR – "побитное исключающее ИЛИ".

Параллельно прохождению информационного потока в скремблере по определенному правилу генерируется поток бит – кодирующий поток. Как прямое, так и обратное шифрование осуществляется наложением по XOR кодирующей последовательности на исходную.

Генерация кодирующей последовательности бит производится циклически из небольшого начального объема информации – ключа по следующему алгоритму. Из текущего набора бит выбираются значения определенных разрядов и складываются по XOR между собой. Все разряды сдвигаются на 1 бит, а только что полученное значение ("0" или "1") помещается в освободившийся самый младший разряд. Значение, находившееся в самом старшем разряде до сдвига, добавляется в кодирующую последовательность, становясь очередным ее битом (см. рис.1.4).

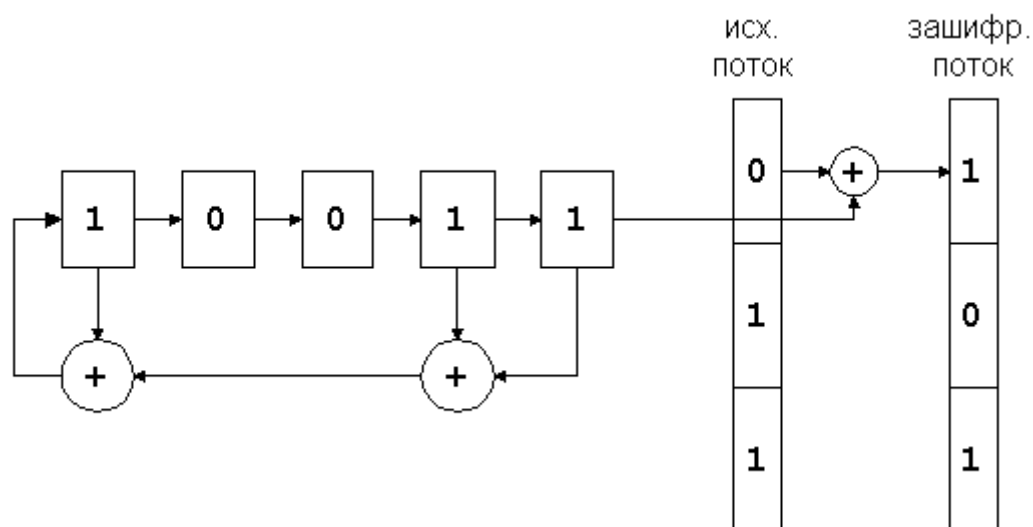


Рис. 1.4. Схема скремблирования

Из теории передачи данных криптография заимствовала для записи подобных схем двоичную систему записи. По ней изображенный на рисунке скремблер записывается комбинацией "10011₂" – единицы соответствуют разрядам, с которых снимаются биты для формирования обратной связи.

Как видим, устройство скремблера предельно просто. Его реализация возможна как на электронной, так и на электрической базе, что и обеспечило его широкое применение в полевых условиях. Более того, тот факт, что каждый бит выходной последовательности зависит только от одного входного бита, еще более упрочило положение скремблеров в защите потоковой передачи данных. Это связано с неизбежно возникающими в канале передаче помехами, которые могут исказить в этом случае только те биты, на которые они приходятся, а не связанную с ними группу байт, как это имеет место в блочных шифрах.

Декодирование заскремблированных последовательностей происходит по той же самой схеме, что и кодирование. Именно для этого в алгоритмах применяется результирующее кодирование по "исключающему ИЛИ" – схема, однозначно восстанавливаемая при раскодировании без каких-либо дополнительных вычислительных затрат. Произведем декодирование полученного фрагмента.

Как Вы можете догадаться, главная проблема шифров на основе скремблеров - синхронизация передающего (кодирующего) и принимающего (декодирующего) устройств. При пропуске или ошибочном вставлении хотя бы одного бита вся передаваемая информация необратимо теряется. Поэтому, в системах шифрования на основе скремблеров очень большое внимание уделяется методам синхронизации. На практике для этих целей обычно применяется комбинация двух методов: а) добавление в поток информации синхронизирующих битов, заранее известных приемной стороне, что позволяет ей при ненахождении такого бита активно начать поиск синхронизации с отправителем, и б) использование высокоточных генераторов временных импульсов, что позволяет в моменты потери синхронизации производить декодирование принимаемых битов информации "по памяти" без синхронизации.

Число бит, охваченных обратной связью, то есть разрядность устройства памяти для порождающих кодирующую последовательность бит называется разрядностью скремблера. Изображенный выше скремблер имеет разрядность 5.

В отношении параметров криптостойкости данная величина полностью идентична длине ключа блочных шифров, который будет проанализирован далее. На данном же этапе важно отметить, что чем больше разрядность скремблера, тем выше криптостойкость системы, основанной на его использовании.

При достаточно долгой работе скремблера неизбежно возникает его закливание. По выполнении определенного числа тактов в ячейках скремблера создается комбинация бит, которая в нем уже однажды оказывалась, и с этого момента кодирующая последовательность начнет циклически повторяться с фиксированным периодом. Данная проблема неустранима по своей природе, так как в N разрядах скремблера не может пребывать более 2^N комбинаций бит, и, следовательно, максимум, через $2^N - 1$ циклов повтор комбинации обязательно произойдет. Комбинация "все нули" сразу же исключается из цепочки графа состояний скремблера – она приводит скремблер к такому же положению "все нули". Это указывает еще и на то, что ключ "все нули" неприменим для скремблера. Каждый генерируемый при сдвиге бит зависит только от нескольких бит хранимой в данный момент скремблером комбинации. Поэтому после повторения некоторой ситуации, однажды уже встречавшейся в скремблере, все следующие за ней будут в точности повторять цепочку, уже прошедшую ранее в скремблере.

Возможны различные типы графов состояния скремблера. На рисунке 1.5 приведены примерные варианты для 3-разрядного скремблера. В случае "А" кроме всегда присутствующего цикла "000" >> "000" мы видим еще два цикла – с 3-мя состояниями и 4-мя. В случае "Б" мы видим цепочку, которая сходится к циклу из 3-х состояний и уже никогда оттуда не выходит. И наконец, в случае "В" все возможные состояния кроме нулевого, объединены в один замкнутый цикл. Очевидно, что именно в этом случае, когда все $2^N - 1$ состояний системы образуют цикл, период повторения выходных комбинаций максимален, а

корреляция между длиной цикла и начальным состоянием скремблера (ключом), которая привела бы к появлению более слабых ключей, отсутствует.

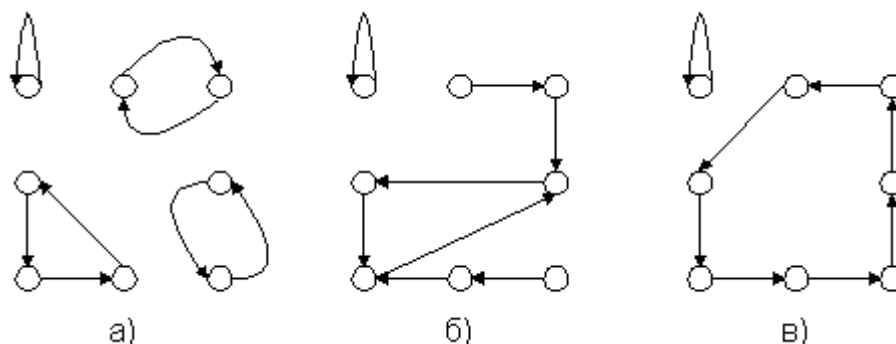


Рис. 1.5. Графы для трех разрядного скремблера

И вот здесь математика преподнесла прикладной науке, каковой является криптография, очередной подарок. Следствием одной из теорем доказывается (в терминах применительно к скремблированию), что для скремблера любой разрядности N всегда существует такой выбор охватываемых обратной связью разрядов, что генерируемая ими последовательность бит будет иметь период, равный $2^N - 1$ битам. Так, например, в 8-битном скремблере, при охвате 0-го, 1-го, 6-го и 7-го разрядов действительно за время генерации 255 бит последовательно проходят все числа от 1 до 255, не повторяясь ни разу.

Схемы с выбранными по данному закону обратными связями называются генераторами последовательностей наибольшей длины (ПНД), и именно они используются в скремблирующей аппаратуре. Из множества генераторов ПНД заданной разрядности во времена, когда они реализовывались на электрической или минимальной электронной базе выбирались те, у которых число разрядов, участвующих в создании очередного бита, было минимальным. Обычно генератора ПНД удавалось достичь за 3 или 4 связи. Сама же разрядность скремблеров превышала 30 бит, что давало возможность передавать до 2^{40} бит =

100 Мбайт информации без опасения начала повторения кодирующей последовательности.

ПНД неразрывно связаны с математической теорией неприводимых полиномов. Оказывается, достаточно чтобы полином степени N не был представим по модулю 2 в виде произведения никаких других полиномов, для того, чтобы скремблер, построенный на его основе, создавал ПНД. Например, единственным неприводимым полиномом степени 3 является x^3+x+1 , в двоичном виде он записывается как 1011_2 (единицы соответствуют присутствующим разрядам). Скремблеры на основе неприводимых полиномов образуются отбрасыванием самого старшего разряда (он всегда присутствует, а следовательно, несет информацию только о степени полинома), так на основе указанного полинома, мы можем создать скремблер 011_2 с периодом зацикливания $7(=2^3-1)$. Естественно, что на практике применяются полиномы значительно более высоких порядков. А таблицы неприводимых полиномов любых порядков можно всегда найти в специализированных математических справочниках.

Существенным недостатком скремблирующих алгоритмов является их нестойкость к фальсификации. Подробнее данная проблема рассмотрена на следующей лекции, применительно к созданию целых криптосистем.

Генераторы псевдослучайной последовательности.

Как уже говорилось выше, скремблеры и дескремблеры цифрового сигнала строятся на основе генераторов псевдослучайных последовательностей битов. Генераторы чаще всего выполняются с использованием M -разрядных сдвиговых регистров RG с цепями обратной связи (рис. 1.6).

Показанные устройства различаются периодом генерируемых последовательностей битов. В генераторе по схеме на рис. 1.6, регистр RG исходно установлен в некоторое ненулевое состояние (цепь начальной установки не показана). Под действием фронтов синхросигнала CLK хранимый в регистре код непрерывно циркулирует в нем и одновременно видоизменяется благодаря

преобразованию битов логическим элементом «Исключающее ИЛИ» (XOR). Генерируемая последовательность битов снимается с выхода этого элемента или с выхода любого разряда регистра. Направление сдвига данных в регистре показано стрелкой. В полном цикле работы генератора в регистре однократно формируются все возможные М-разрядные коды за исключением нулевого. Циклы следуют один за другим без пауз.

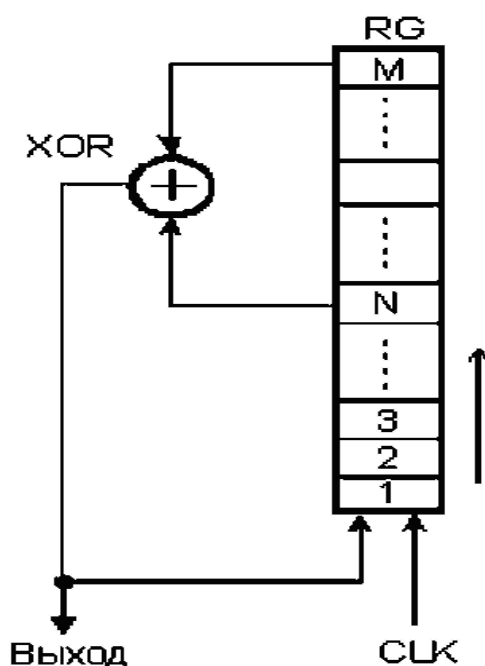


Рис. 1.6. Генератор псевдослучайной последовательности

В общем случае при использовании М-разрядного регистра цепь обратной связи подключается к разрядам с номерами М и N ($M > N$). Для того чтобы на выходе генератора формировалась псевдослучайная последовательность битов с периодом повторения, равным $2^M - 1$, следует выбирать точки подключения цепи обратной связи в соответствии с:

$M=3$ 4 5 6 7 9 10 11 15 17 18 20 21 22 23 25 28 29 31 33 35 36 39 ...

$N=2$ 3 3 5 6 5 7 9 14 14 11 17 19 21 18 22 25 27 28 20 33 25 35

Отметим, что не для любой разрядности регистра М возможно построение генератора с периодом $2^M - 1$ по такой простой схеме.

Псевдослучайная последовательность битов с периодом повторения, равным 2^M-1 , обладает следующими свойствами.

- В полном цикле (2^M-1 тактов) число лог. 1, формируемых на выходе генератора, на единицу больше, чем число лог. 0. Добавочная лог. 1 появляется за счет исключения состояния, при котором в регистре присутствовал бы нулевой код. Это можно интерпретировать так, что вероятности появления логического 0 и 1 на выходе генератора практически одинаковы.

Вероятности появления логической единицы и логического нуля в каждой последующей позиции потока битов одинаковы и не зависят от предыстории. Применительно к телекоммуникационным системам скремблирование повышает надежность синхронизации устройств, подключенных к линии связи, и уменьшает уровень помех, излучаемых на соседние линии многожильного кабеля. Есть и иная область применения скремблеров — защита передаваемой информации от несанкционированного доступа.

В полном цикле (2^M-1 тактов) половина серий из последовательных лог. 1 имеет длину 1, одна четвертая серий — длину 2, одна восьмая — длину 3 и т. д. Такими же свойствами обладают и серии из лог. 0 с учетом пропущенного лог. 0. Это говорит о том, что вероятности появления «орлов» и «решек» не зависят от исходов предыдущих «подбрасываний». Поэтому вероятность того, что серия из последовательных лог. 1 или 0 закончится при следующем подбрасывании, равна $1/2$.

Если последовательность полного цикла (2^M-1 тактов) сравнивать с этой же последовательностью, но циклически сдвинутой на любое число тактов W (W не является нулем или числом, кратным 2^M-1), то число несовпадений будет на единицу больше, чем число совпадений.

Усовершенствованный вариант генератора (рис. 1.7) формирует псевдослучайную последовательность битов с периодом повторения, равным 2^M . К нему также применима таблица, приведенная на рис. 1, в. В регистре RG в определенном порядке формируются все возможные коды, включая нулевой.

Генератор дополнительно содержит элемент «ИЛИ-НЕ», инвертор и мультиплексор MS. Сигнал Z на выходе элемента «ИЛИ-НЕ» задает направление передачи данных через мультиплексор. При $Z = 0$ на выход мультиплексора транслируется сигнал с выхода элемента «Исключающее ИЛИ», а при $Z = 1$ — сигнал с выхода инвертора.

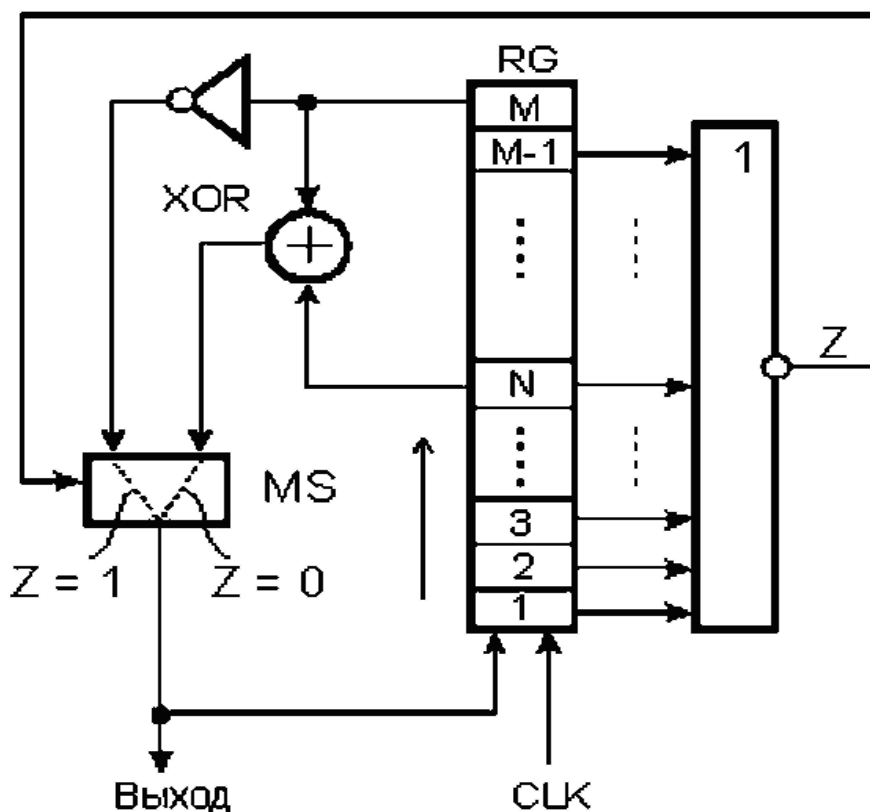


Рис. 1.7. Усовершенствованный вариант генератора псевдослучайной последовательности

До тех пор, пока на входах элемента «ИЛИ-НЕ» присутствует хотя бы одна лог. 1, на его выходе будет сигнал $Z = 0$. В этом случае мультиплексор MS в каждом такте передает в освободившийся (нижний) разряд сдвигового регистра бит с выхода элемента «Исключающее ИЛИ» так же, как и в схеме, показанной на рис. 1.6.

В некотором такте i в регистре фиксируется код, содержащий единственную лог. 1, размещенную в разряде $M-1$. Так как в разрядах M и N присутствуют лог. 0, то на выходе элемента «Исключающее ИЛИ» сформирован сигнал лог. 0, который к началу такта $i+1$ поступает на вход регистра. В начале такта $i+1$ лог. 1 перемещается из разряда $M-1$ в разряд M , на входах элемента «ИЛИ-НЕ» формируется нулевой код. Сигнал $Z = 1$ переводит мультиплексор MS в состояние, при котором на вход нижнего разряда сдвигового регистра поступает бит с выхода инвертора. В данном случае этот бит равен 0, поэтому в такте $i+2$ в регистре фиксируется нулевой код.

К началу такта $i+3$ на вход сдвигового регистра с выхода инвертора поступает лог. 1, поэтому по фронту синхросигнала CLK в регистре фиксируется код, содержащий лог. 0 во всех разрядах, кроме первого. Сигнал Z вновь принимает нулевое значение, мультиплексор переключается в состояние передачи сигнала с выхода элемента «Исключающее ИЛИ» и т. д. Таким образом регистр проходит через все состояния, включая нулевое. Возможны и иные варианты построения генераторов с числом состояний, равным 2^M .

Система с неизолированными генераторами.

В системе, показанной на рис. 1.8, скремблер и дескремблер содержат фрагменты рассмотренного ранее генератора (рис. 1.6) псевдослучайных последовательностей битов. В скремблере цепь обратной связи генератора на основе сдвигового регистра $RG1$ дополнительно содержит элемент «Исключающее ИЛИ» $XOR2$. В дескремблере применен аналогичный генератор на основе сдвигового регистра $RG2$ с разомкнутой цепью обратной связи.

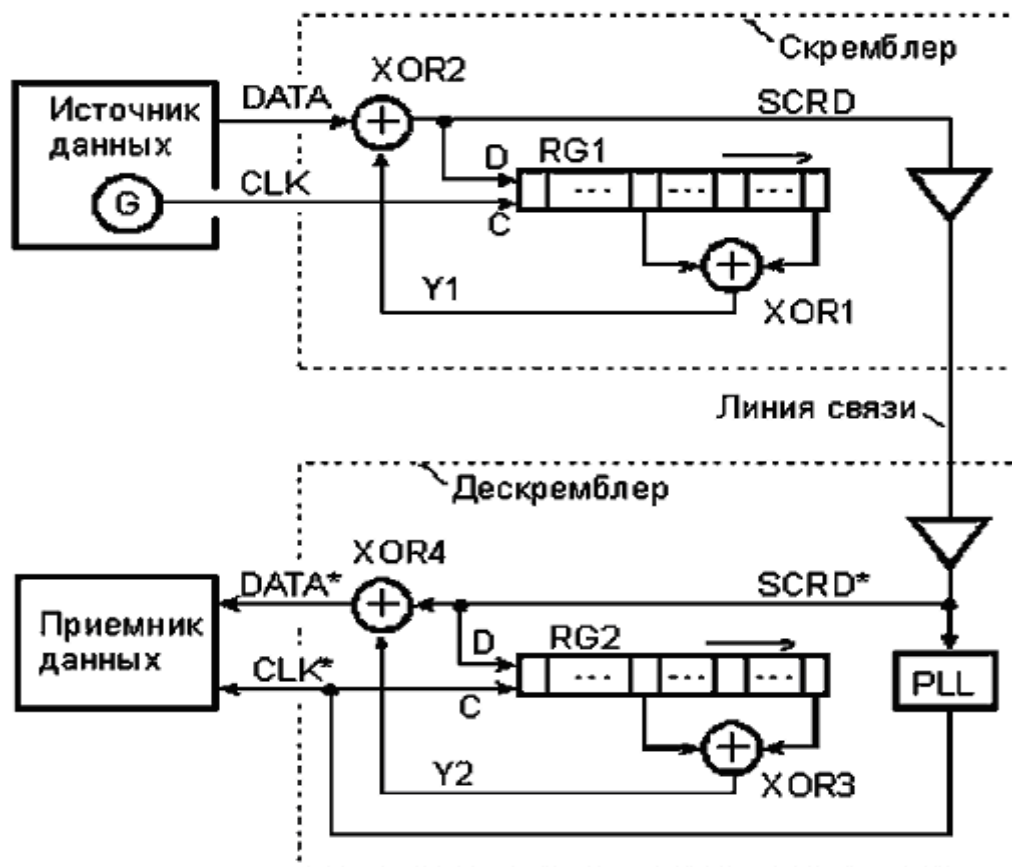


Рис. 1.8. Скремблер и дескремблер с неизолированным генератором

На этом рисунке показана система «скремблер — дескремблер» с неизолированными генераторами псевдослучайных последовательностей битов.

Все процессы, протекающие в системе, синхронизируются от тактового генератора G, размещенного в источнике данных (возможно также его размещение в скремблере). Тактовый генератор формирует сигнал CLK — непрерывную последовательность тактовых импульсов со скважностью, равной двум. В каждом такте по фронту сигнала CLK на вход скремблера подается новый бит передаваемых данных DATA, а код в его сдвиговом регистре RG1 продвигается на один разряд вправо, причем в этот же момент в освободившийся разряд заносится старый бит данных, просуммированный по модулю два со старым битом Y с выхода элемента XOR1.

Строго говоря, на границах между битовыми интервалами на выходе элемента XOR2 могут сформироваться короткие ложные импульсы в результате

неодновременного формирования новых сигналов на его входах (сигнал Y1 приходит чуть позже сигнала DATA). Для устранения ложных импульсов можно ввести в цепь сигнала SCRD D-триггер, синхронизируемый спадом сигнала CLK (триггер на рисунке не показан). Короткими ложными импульсами пока пренебрегаем для упрощения изложения основных идей построения систем «скремблер — дескремблер».

Если источник данных посылает в скремблер длинную последовательность сигналов лог. 0 ($DATA \equiv 0$), то элемент XOR2 можно рассматривать как повторитель сигнала Y1. Тогда регистр RG1 фактически оказывается замкнутым в кольцо и генерирует точно такую же псевдослучайную последовательность битов, как и в рассмотренной ранее схеме генератора, приведенной на рис. 1.6. Отметим, что в этой ситуации при неблагоприятном стечении обстоятельств есть опасность потери работоспособности скремблера, если в регистре RG1 к началу передачи последовательности сигналов лог. 0 зафиксирован нулевой код (подробнее этот случай рассмотрим ниже). Если от источника данных поступает произвольная битовая последовательность, то она взаимодействует с последовательностью битов с выхода элемента XOR1. В результате формируется новая (скремблированная) последовательность битов данных SCRD, по структуре близкая к случайной. Эта последовательность, в свою очередь, продвигается по регистру RG1, формирует поток битов Y1 на выходе элемента XOR1 и т. д.

Скремблированная последовательность битов SCRD проходит через передающий усилитель и по линии связи поступает в дескремблер, где проходит через приемный усилитель. Линия связи может быть выполнена, например, в виде витой пары проводов многожильного кабеля городской телефонной сети. С помощью генератора PLL (Phase Locked Loop) с фазовой автоподстройкой частоты из входного сигнала SCRD* выделяется тактовый сигнал CLK*, который передается на синхронизирующие входы регистра RG2 и приемника данных.

Генератор PLL с фазовой автоподстройкой частоты может быть построен по одной из известных схем (например, [6]). Он предназначен для формирования высокостабильного синхросигнала CLK* на основе непрерывного слежения за входным сигналом SCRD*. В данном случае спад сигнала CLK* привязан к моментам изменения сигнала SCRD* ($0 \rightarrow 1$ или $1 \rightarrow 0$), так что фронт сигнала CLK* формируется в середине битового интервала сигнала SCRD*, что соответствует его установившемуся значению. Сдвиг данных в регистре RG2 и прием очередного бита SCRD* в его освободившийся разряд происходят по фронту сигнала CLK*. Дескремблированные данные DATA* поступают в приемник данных и фиксируются в нем по фронтам сигнала CLK*. Благодаря достаточной инерционности генератора PLL сигнал CLK* практически нечувствителен к «дрожанию фазы» сигнала SCRD* и иным его кратковременным искажениям, вызванным помехами в линии связи.

Потоки данных DATA и DATA* совпадают с точностью до задержки передачи. Действительно, в установившемся режиме в сдвиговых регистрах RG1 и RG2 присутствуют одинаковые коды, так как на входы D этих регистров поданы одни и те же данные $SCRD = SCRD^*$ (с учетом задержки передачи), а тактовая частота одна и та же. Поэтому $Y2 = Y1$ и с учетом этого $DATA^* = SCRD^* \odot Y2 = SCRD \odot Y2 = (DATA \odot Y1) \odot Y2 = DATA \odot Y1 \odot Y1 = DATA \odot 0 = DATA$.

Рассмотренный способ скремблирования-дескремблирования данных не требует применения какой-либо специальной процедуры начальной кодовой синхронизации, после которой коды в обоих регистрах становятся одинаковыми и, следовательно, начинает выполняться условие $Y2 = Y1$. Синхронизация достигается автоматически после заполнения регистров одинаковыми данными. Это, пожалуй, единственное преимущество данного варианта перед классическим устройством с изолированными генераторами (рис. 3).

К сожалению, есть и существенные недостатки. О первом из них уже вскользь упоминалось — это плохая устойчивость по отношению к некоторым

неблагоприятным кодовым ситуациям, которые могут возникнуть как при нормальной работе системы, так и в результате злого умысла. Этот недостаток и меры его устранения будут подробно рассмотрены далее.

Второй недостаток состоит в размножении ошибок. При появлении одиночной ошибки в линии связи идентичность содержимого регистров RG1 и RG2 временно нарушается, но затем автоматически восстанавливается, как только правильные данные вновь заполнят регистр RG2. Однако в процессе продвижения ошибочного бита по сдвиговому регистру RG2, а именно в периоды его попадания сначала на один, а затем на другой вход элемента XOR3 сигнал Y2 дважды принимает неправильное значение. Это приводит к размножению одиночной ошибки — она впервые появляется в сигнале DATA* в момент поступления из линии и затем возникает еще два раза при последующем искажении сигнала Y2.

Система с изолированными генераторами

В системе, показанной на рис. 1.9, применены изолированные от линии связи генераторы псевдослучайных битовых последовательностей. Преимущества этой системы перед предыдущей заключаются в том, что ошибки, поступающие из линии, не размножаются, и как будет показано ниже, такая система более устойчива по отношению к неблагоприятным последовательностям битов, которые формируются в силу случайных стечений обстоятельств, либо в результате преднамеренных действий пользователя.

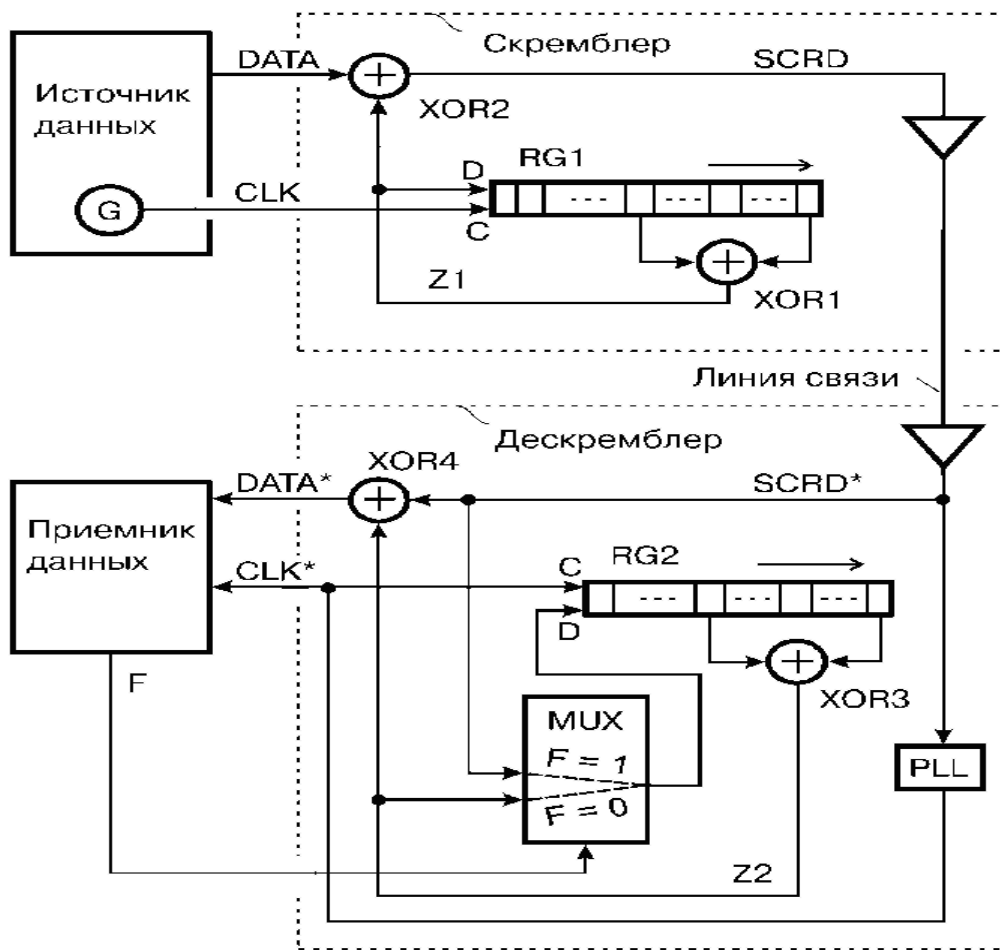


Рис. 1.9. Скремблер и дескремблер с изолированным генератором

Недостаток этой системы — сложность установления кодовой синхронизации. Отметим, что в конце статьи предложено решение, лишенное этого недостатка — оно предусматривает автоматическое установление и поддержание синхронной работы изолированных генераторов псевдослучайных битовых последовательностей.

Начальная кодовая синхронизация системы с изолированными генераторами псевдослучайных последовательностей битов (рис.1.9) осуществляется с использованием аппаратных средств дескремблера и программных средств источника и приемника данных. К аппаратным средствам относятся мультиплексор MUX и программно-управляемый выход приемника данных, на котором формируется управляющий сигнал F. При нормальной работе системы приемник данных постоянно поддерживает на выходе сигнал $F = 0$. На выход мультиплексора транслируется сигнал Z2 с выхода элемента

«Исключающее ИЛИ» XOR3. Генератор псевдослучайной битовой последовательности на основе регистра RG2 изолирован от внешних воздействий со стороны линии связи.

Предположим, что в исходном состоянии дескремблер не синхронизирован со скремблером. Такая ситуация может возникнуть, например, после включения напряжения питания аппаратуры приемной стороны, после ошибки в работе генератора PLL дескремблера из-за воздействия помех на линию связи или по иным причинам. При отсутствии кодовой синхронизации между скремблером и дескремблером содержимое регистров RG1 и RG не совпадает, поток принимаемых данных DATA* ошибочен и не совпадает с потоком передаваемых данных DATA.

При обнаружении устойчивого «хаоса» в данных DATA* (когда в потоке нет обусловленного протоколом обмена разделения на информационные кадры и т. п.) приемник формирует сигнал $F = 1$. Вследствие этого мультиплексор начинает транслировать на вход D регистра RG2 сигнал скремблированных данных SCRD*, как в ранее рассмотренной системе.

Протокол обмена предусматривает пересылку данных в виде последовательности кадров. Группы обычных кадров перемежаются со служебными кадрами. Например, после группы из 1000 обычных кадров следует один служебный. Он, в частности, содержит синхронизирующую последовательность из некоторого числа (например, 256) нулевых битов. При выдаче этих битов ($DATA = 0$) в скремблер элемент XOR2 выполняет функцию повторителя сигнала Z1 с выхода элемента XOR1. Поэтому в данном случае скремблированный сигнал SCRD представляет собой фрагмент «истинной» псевдослучайной битовой последовательности в том смысле, что она не смешана с потоком произвольных данных DATA и порождается только генератором скремблера.

Эта последовательность загружается в регистр RG2 и проходит через него, так как $F = 1$. После того как содержимое регистров RG1 и RG2 оказывается

одинаковым, сигнал Z2 начинает повторять сигнал Z1. Кодовая синхронизация достигнута. На вход приемника данных подается непрерывная последовательность лог. 0, так как $DATA^* = DATA \equiv 0$. После уверенного обнаружения достаточно длинной (например, содержащей 220 бит) последовательности лог. 0 приемник данных формирует сигнал $F = 0$ и тем самым возвращает генератор псевдослучайной последовательности битов дескремблера в режим изолированной работы. Теперь кодовая синхронизация не только достигнута, но и «сохранена» благодаря логической изоляции регистра RG2 от линии связи. После окончания передачи служебного (синхронизирующего) кадра источник данных приступает к передаче группы из 1000 обычных кадров согласно принятому в системе протоколу обмена.

Таким образом, в рассмотренной системе для поддержания синхронной работы сдвиговых регистров скремблера и дескремблера (в случае нарушения синхронизации или при первоначальном включении приемной части системы) необходимо периодически прерывать передачу полезных данных и передавать по линии связи служебные информационные кадры, содержащие достаточно длинные цепочки синхронизирующих битов ($DATA \equiv 0$). В результате уменьшается эффективная скорость передачи данных по линии, усложняется протокол обмена. Кроме того, с увеличением интервалов между служебными кадрами (что способствует более эффективной передаче пользовательских данных) увеличивается время ожидания этих кадров дескремблером в случае потери кодовой синхронизации. В течение времени ожидания передача полезных данных невозможна.

Повышение устойчивости синхронизации.

Следует рассмотреть вопросы улучшения синхронизации и устойчивости скремблеров по отношению к нежелательным последовательностям битов. При передаче данных по линии связи применяют различные способы кодирования. В частности, широкое распространение получило NRZ-кодирование и его

модификации. Для передачи нулевых и единичных битов выделяются одинаковые интервалы времени — «битовые интервалы». В каждом битовом интервале в зависимости от значения передаваемого бита (лог. 1 или 0) между проводами витой пары (двухпроводной линии) присутствует положительное или отрицательное напряжение. Применительно к оптоволоконным линиям в каждом битовом интервале по оптическому волокну передается или не передается световой поток. Такое кодирование неприменимо в случаях, когда по линии могут передаваться очень длинные цепочки из одинаковых битов. Тогда состояние линии будет оставаться неизменным на протяжении длительного интервала времени, и синхронизация приемника с передатчиком может нарушиться. Действительно, приемник надежно восстанавливает синхросигнал только тогда, когда паузы между изменениями сигнала в линии не слишком велики. Изменение сигнала в линии после незначительной паузы позволяет всякий раз корректировать «ход часов» приемника. С увеличением паузы надежность «службы времени» падает. Например, после передачи серии из 10000 нулей приемник, вероятнее всего, не сможет с уверенностью определить, находится ли последующая единица на позиции 9999, 10000 или 10001. То же относится и к передаче длинных цепочек лог. 1. Другими словами, при передаче достаточно большой последовательности нулей или единиц приемник теряет синхронизацию с передатчиком.

На практике данные группируют в кадры постоянной или переменной длины. Каждый кадр содержит некоторую служебную информацию, например, флаговый код начала кадра, причем эта информация заведомо не является последовательностью одинаковых битов. Это облегчает поддержание синхронизации, так как возможная однородность пользовательских данных периодически нарушается заведомо неоднородными служебными данными. Тем не менее, для повышения надежности синхронизации желательно исключить из потока пользовательских данных длинные последовательности нулевых или единичных битов. Это и делается с помощью скремблирования — це-почки

нулевых или единичных битов (и не только они) преобразуются в псевдослучайные битовые последовательности, в которых вероятность изменения уровня сигнала в каждом последующем битовом интервале (по отношению к текущему уровню) равна $1/2$

В подтверждение сказанного об устойчивости синхронизации рассмотрим цепь выделения синхросигнала и данных из сигнала в линии.

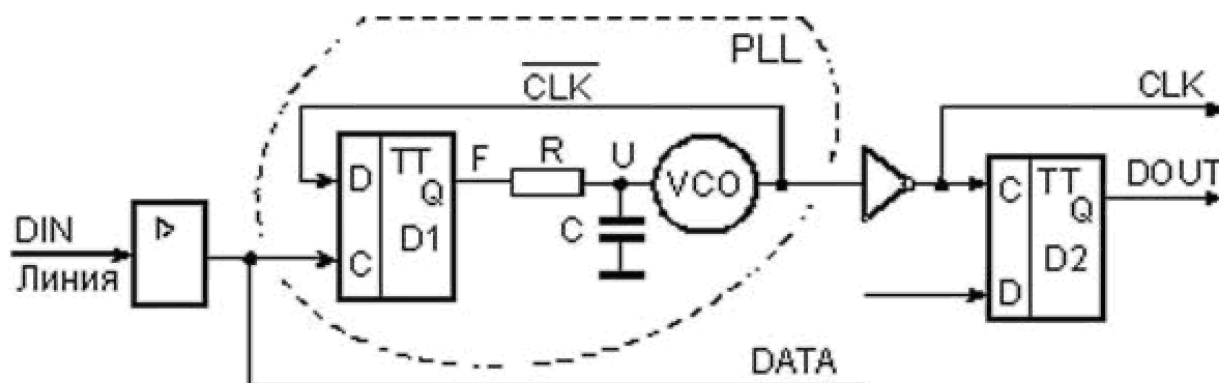


Рис. 1.10. Цепь выделения синхросигнала

Узел содержит входной усилитель, генератор PLL с фазовой автоподстройкой частоты, инвертор и триггер D2. Генератор PLL включает в себя триггер D1, фильтр низких частот (RC-цепь) и генератор VCO (Voltage Controlled Oscillator), управляемый напряжением U. Триггеры D1 и D2 принимают данные с входов D по фронту сигнала на входе C. Генератор VCO формирует выходной синхросигнал CL. Частота этого сигнала в незначительных пределах может изменяться в зависимости от напряжения U на его управляющем входе. С повышением этого напряжения частота уменьшается, и наоборот.

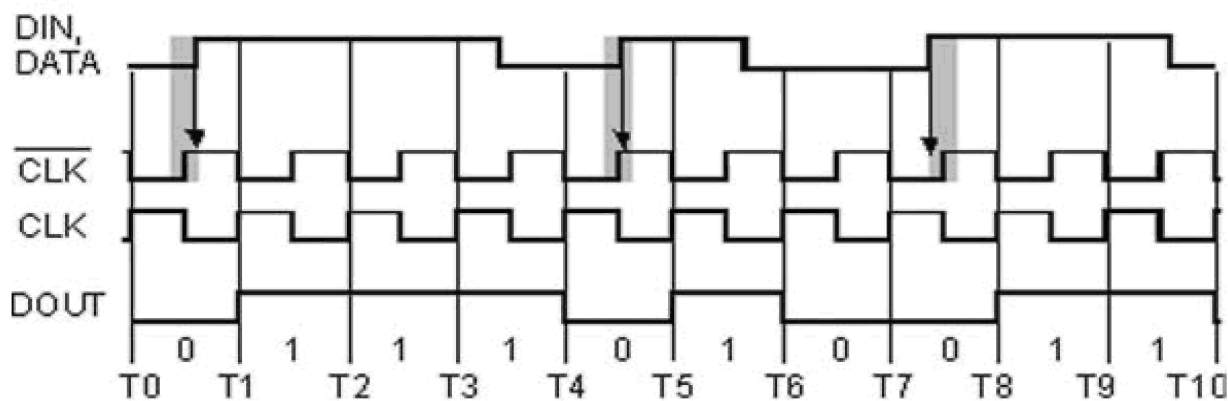


Рис. 1.11. Формирование синхросигнала

Данные DATA представлены кодом NRZ. Границы битовых интервалов примерно соответствуют моментам формирования спадов сигнала CLK, а середины этих интервалов — моментам T0—T10.

Обратите внимание на нестандартный режим работы триггера D1. Обычно на вход С синхронизации D-триггера подается синхроимпульс, а на вход D — данные. При этом согласно техническим условиям на триггер к моменту формирования фронта синхроимпульса сигнал на входе D данных должен принять установившееся значение в течение некоторого «времени предустановки» и сохранять это значение в течение «времени удержания», которое отсчитывается от того же фронта. Здесь же реализован некий противоположный режим. В качестве данных на D-вход триггера поступает синхросигнал, а в качестве синхросигнала данные. При этом в идеальном случае фронт сигнала на входе С должен совпадать с фронтом сигнала на входе D.

Как показано на рис. 1.11 затемненными прямоугольниками, существуют области неопределенности моментов формирования фронтов сигнала данных. Неопределенность обусловлена так называемым джиттером (дрожанием фронтов). Джиттеру в равной мере подвержены и спады входного сигнала, но они нас в данном случае не интересуют.

Предположим, что генератор VCO уже вошел в синхронизацию с сигналом, поступающим из линии. В этом случае фронты сигнала DATA с равной

вероятностью совмещены во времени с нулевыми или единичными уровнями инвертированного сигнала CLK. На рисунке вертикальные стрелки показывают, что два фронта сигнала DATA совмещены с единичным уровнем инвертированного сигнала CLK, а третий — с нулевым уровнем. При равновероятном приеме в триггер D1 лог. 0 и 1 сигнал F на его выходе постоянно изменяется, но в среднем пребывает в состоянии лог. 0 столько же времени, сколько и в состоянии лог. 1. При этом напряжение U на выходе интегрирующей RC-цепи примерно равно половине напряжения, соответствующего уровню лог. 1 на выходе триггера D1.

Предположим теперь, что вертикальные стрелки в большинстве своем указывают на нулевые состояния синхросигнала с выхода генератора VCO. Это означает, что следует слегка увеличить частоту синхросигнала, что приведет к его незначительному фазовому опережению, т. е. к некоторому «сжатию влево» его временной диаграммы. Это и произойдет благодаря тому, что в данной ситуации сигнал управления F будет преимущественно нулевым, напряжение U снизится, частота сигнала CLK незначительно увеличится.

В противоположной ситуации временная диаграмма синхросигнала исходно чуть смещена влево относительно показанной на рисунке. Тогда стрелки должны преимущественно попадать на единичные состояния синхросигнала. Это означает, что синхросигнал вырабатывается с опережением, и его следует задержать. Средством задержки служит незначительное снижение частоты. Оно достигается благодаря тому, что в данной ситуации управляющий сигнал F преимущественно равен единице, напряжение U повышается, частота незначительно снижается.

В результате малых постоянных колебаний около равновесного состояния осуществляется статистически наиболее правильная привязка синхросигнала CLK к сигналу DATA. Такая привязка обеспечивает стабильность данных на входе D триггера D2 в момент прохождения фронта сигнала CLK. Таким

образом, на выходе схемы формируется синхросигнал и данные, выделенные из входного сигнала DIN.

Если сигнал DIN в течение длительного времени остается неизменным, то управление генератором VCO теряется, так как постоянно действующий на него через интегрирующую цепь нулевой или единичный сигнал F вызывает неуклонное повышение или снижение частоты синхросигнала. Скремблирование передаваемого по линии сигнала предотвращает его длительную фиксацию, т. е. повышает надежность синхронизации.

3. Порядок выполнения работы

Описание лабораторной установки

Лабораторная установка позволяет осуществить физический анализ сигналов, лабораторная установка включает в себя аппаратную и программную часть, аппаратная часть состоит из (отладочная плата DSP и персонального компьютера соединенного с ней по шине USB) программная часть включает в себя (программный код, среда проектирования visual dsp). Структурная схема приведена на рисунке 1.12.

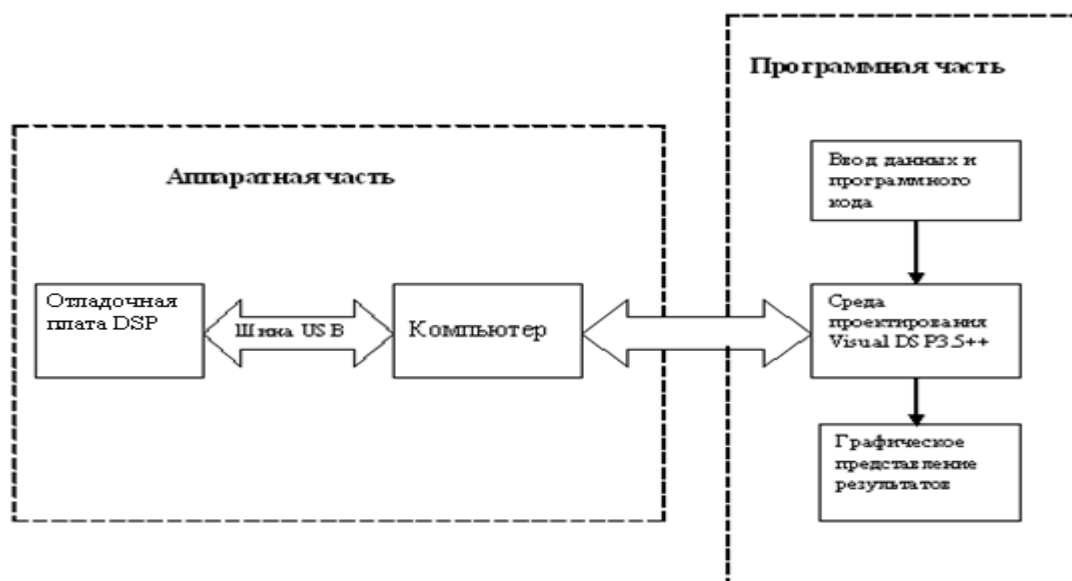


Рис. 1.12. Структурная схема лабораторного макета

Порядок выполнения лабораторной работы

1. Уяснить функциональное назначение отдельных узлов макета, и принципы шифрования данных
2. Собрать лабораторную установку для исследования методов Цифровой обработки сигналов
3. Включить лабораторную установку.
4. Выполнить исследование открытых сигналов, псевдослучайной последовательности, шифр сигналов, формируемых в лабораторной установки.

Примечание. Предполагается изменять по указанию преподавателя параметры:

Скорость ввода информационной последовательности

Начальное состояние регистров (ключ шифрования)

Длину сдвигового регистра и расположение отводов.

Изображения получаемых сигналов при изменении указанных параметров рекомендуется помещать на графиках один под другим при неизменном масштабе по оси времени.

Исследование проводить в следующем порядке:

1. Запустить VisualDSP++ Environment (на рисунке представлен интерфейс программы).
2. В закладке Project выбрать New.
3. Нажать CTRL+N и написать исходный текст.
4. Сохранить с расширением asm.
5. Подключить этот файл в проект, Project->Add to Project->File.
6. Для создания файла архитектуры процессора необходимо выполнить 2-4 пункты, но сохранить с расширением ldf.
7. Для использования готового файла архитектуры процессора необходимо

выполнить

пункт 5 и найти его.

8. Для подключения файлов с расширением dat (например, содержащих коэффициенты

фильтра), необходимо нажать закладки Project->Add to Project->New Folder.

9. Подключить в папку файлов данных необходимые файлы, однократным нажатием

левой кнопкой мышки на созданной папке, и щелчком правой кнопкой мышки

выбрать Add File(s) to folder.

10. Отредактировать текст исходной программы и откомпилировать нажатием клавиши F7.

11. При появлении ошибок в Output window, посмотреть номера строки, отредактировать и снова выполнить пункт 10.

12. Пошагово отлаживать клавишей F11

13. Исследовать характеристики входного сигнала;

14. Исследовать псевдослучайной последовательности при изменении начального

состояния регистров, Длину сдвигового регистра и расположение отводов

15. Исследовать характеристики шифр сигнала, для различных параметров измененных в п2.

16. Определить криптостойкость шифр последовательности для 3 значений длины

сдвигового регистра.

17. Сделать необходимые выводы по каждому из пунктов исследования.

4. Рекомендуемая литература

1.General DSP Training and Workshops: <http://www.analog.com/industry/dsp/training>

2. DSP Designer's Reference (DSP Solutions) CD-ROM, Analog Devices, 1999.
4. Марпл-мл. С. Л. Цифровой спектральный анализ и его приложения. - М.: Мир, 1990, -234с.
5. Гольденберг Л.М., Матюшкин Б.Д., Поляк М.Н.. Цифровая обработка сигналов. - М: Радио и связь, 1985, - 340 с.
6. Эммануил С. Айфичер, Барри У. Джервис. Цифровая обработка сигналов: практический подход, 2-е издание.- М.: Изд-во Вильямс, 2004. – 992 с.

Лабораторная работа 2. Исследование методов стеганографии информации в сетях и системах радиосвязи

1. Цель работы

Объектом исследования является программный комплекс, предназначенный для проведения стегоанализа графических файлов формата JPEG в области исследования передаваемой информации графического характера. В рассматривается проблема вскрытия стегоканала в файлах графического формата JPEG. Сегодня данный формат наиболее часто используется в глобальных и локальных вычислительных сетях для передачи графической информации, так как обладает возможностью существенно уменьшать объем файла по сравнению с другими форматами за счет сжатия данных при сохранении приемлемого качества изображения. Разработка программного комплекса выполнена в среде Builder C.

2. Краткие теоретические сведения

Введение

Попытки скрыть информацию предпринимались во все времена на протяжении истории человечества. Уже в древнем мире выделилось два основных направления решения этой задачи, существующие и по сегодняшний день: криптография и стеганография. Стеганография, наряду с криптографией и криптоанализом, являясь разделом криптологии, описывает методы сокрытия факта передачи сообщения. Целью криптографии является сокрытие содержимого сообщений за счет их шифрования. В отличие от этого при стеганографии скрывается сам факт существования тайного сообщения.

Развитие средств вычислительной техники в последнее десятилетие дало новый толчок для развития компьютерной стеганографии, появилось много новых областей ее применения. Теперь сообщения встраивают в цифровые данные, как правило, имеющие аналоговую природу. Это – речь, аудиозаписи, изображения,

видео. Известны также предложения по встраиванию информации в текстовые файлы и в исполняемые файлы программ. Наибольшую популярность приобрели методы встраивания данных в изображения.

В последние десятилетия широкое распространение получили методы цифровой стеганографии. Наиболее удобными для использования контейнерами являются графические файлы, поскольку в них имеется большой объем психовизуальной избыточности. Особую популярность приобрели программы, скрывающие информацию в файле формата BMP с 24-разрядным кодированием цвета (значение цветовой составляющей точки лежит в интервале от 0 до 255), используя метод младших значащих разрядов (LSB). Обнаружить скрытую передачу данных достаточно тяжело, анализируя только предположительно заполненные контейнеры. Решением данной проблемы является программный стегоанализ, то есть применение программных средств, которые позволяют с некоторой вероятностью определить наличие информации в контейнере или ее отсутствие.

Таким образом, можно обозначить направление работы, цель которого является программная реализация алгоритмов обнаружения факта передачи скрытой информации в канале передачи данных.

Стеганография. Понятия и определения

Необходимость скрыть какую-либо информацию от чужих глаз возникла очень и очень давно. Спрятать информацию можно разными способами, например, зашифровать ее. Правда, в этом случае противник знает, что вы передаете некоторое секретное сообщение, но не может его прочитать (криптография). А ведь иногда достаточно и самого факта передачи для получения информации о каком-то событии, особенно если рассматривать и сопоставлять все факты вместе - на этом основана разведка по материалам из открытых источников. Так

вот другой способ состоит в том, чтобы скрыть не только сообщение, но и сам факт его передачи, при этом секретная информация может содержаться во вполне безобидной фразе, например:

"КОМПАНИЯ "ЛОМО" ИСПОЛЬЗУЕТ ЕДКИЙ НАТРИЙ, ТЯЖЕЛЫЕ ГРУЗИЛА, ОСТРОГУ ТРЕХЗУБУЮ, ОБВЕТШАЛЫЙ ВАТНИК".

Обратите внимание на первые буквы, они складываются в предложение: "Клиент готов". Этот пример хотя и тривиален, позволяет проиллюстрировать способ скрытия информации, называемый стеганографией (в переводе с греческого - "покрытое письмо").

Стеганография известна еще со времен Геродота. В Древней Греции послания писались острыми палочками на дощечках, покрытых воском. В одной из историй Демерат хотел послать в Спарту сообщение об угрозе нападения Ксерксов. Тогда он соскоблил воск с дощечки, написал послание непосредственно на дереве, затем вновь покрыл ее воском.

В результате доска выглядела неиспользованной и без проблем прошла досмотр центурионов

К стеганографии также относится написание текстов невидимыми чернилами, проявляющимися при нагревании. По мере совершенствования техники обнаружения тайнописи появились и специальные химические соединения. Но это было до компьютеров.

В настоящее время, когда объемы различной информации все растут, соответственно растет доля сведений, которые необходимо держать в тайне от посторонних глаз. Применение компьютеров позволило усовершенствовать известные идеи скрытия информации и дало возможность так прятать текст и любые другие данные, что их дешифровка без знания ключей и паролей стала практически невозможной.

Конечно же, наряду с шифрованием, стеганография является одним из основных методов сохранения конфиденциальной информации. В процессе исследования стеганографии в рамках компьютерной безопасности, становится

очевидным, что она, по существу, не является чем-то новым, так как возникла еще во времена древнего Рима. К примеру, в древнем Риме и Греции тексты обычно писались на воске, которым заливали каменные таблички. Если отправитель информации хотел скрыть сообщение (например, в целях военной тайны), он использовал стеганографию: воск соскребался, и сообщение либо вставлялось, либо писалось прямо на табличке, затем воск снова заливался поверх послания. Таким образом, скрывалось не только значение сообщения, но и вообще его существование.

Стеганография (также известная как "steg" или "stego") – это «искусство написания цифр или букв, которые не понятны никому, кроме человека, у которого есть ключ - cryptography». В компьютерный мир стеганография вошла как метод сокрытия определенного сообщения в другом (которое есть заведомо большим), таким образом, что невозможно увидеть присутствие или смысл скрытого сообщения. В современном мире – это цифровая стратегия сокрытия файла в мультимедийном формате, например: картинка, звуковой файл(wav, mp3) или даже видеофайл.

Как и большинство утилит по безопасности, стеганография может использоваться для разных целей. Законные цели включают водяные знаки на изображениях в целях защиты прав собственности. Цифровые подписи (также известны как fingerprinting (отпечатки), указывают в основном на объекты, защищенные законом об авторском праве) являются обычными для стеганографии, так как содержатся в файлах, являясь частью их, и потому сложно обнаружимы обыкновенным человеком. Стеганография может также использоваться для замены одностороннего значения хэша (когда берется длина вводимой переменной и создается статическая длина строки на выходе, для того, чтобы подтвердить, что водимая переменная не была изменена). Также стеганография может быть использована для добавления заметок на онлайн-изображения (как стикеры для бумаг). И, наконец, стеганография может использоваться для сохранения ценной информации, в целях защиты данных от

возможного саботажа, кражи или несанкционированного просмотра. К сожалению, стеганография может быть также использована для незаконных целей. Например, при попытке украсть информацию, существует возможность скрыть ее в другом файле, или файлах и послать ее в виде невинного письма или файла. И как говорилось ранее, она может использоваться в целях террористов как средство скрытого общения. Конечно же, применение стеганографии может быть как законным, так и незаконным.

1.1 Стегоанализ понятия и определения

Дадим определения основным терминам, используемым в дипломной работе.

Стеганографический анализ или **стегоанализ** – раздел криптологии о методах обнаружения стеганографического канала (вскрытия факта передачи сообщения), а также о способах извлечения, разрушения и подмены скрытого сообщения при известном или неизвестном алгоритме и недоступных параметрах (ключах) стегосистемы.

Стегоаналитик или **атакующий** (противник) – специалист в области стегоанализа.

Атака – попытка стегоаналитика обнаружить стегоканал, извлечь, изменить или разрушить скрытое сообщение.

Стеганографическая система или **стегосистема** – совокупность средств и методов, которые используются для формирования скрытого канала передачи информации.

Контейнер – любая информация, предназначенная для сокрытия конфиденциальных сообщений.

Пустой контейнер – контейнер без встроенного сообщения.

Заполненный контейнер или **стега** – контейнер, содержащий встроенное сообщение.

Встроенное (скрытое) сообщение – сообщение, встраиваемое в контейнер. В статье используется термин «закладка» («информационная закладка»), который полностью аналогичен термину «встроенное сообщение».

Стеганографический канал или **стегоканал** – канал передачи стего.

Стегоключ или просто **ключ** – секретный ключ, необходимый для сокрытия и извлечения информации.

Из определения стегоанализа вытекают решаемые задачи. Обнаружение стеганографического канала является наиболее трудно реализуемой задачей. Только после ее решения возможно выполнение других задач стегоанализа. Следующей по сложности необходимо назвать задачу подмены скрытого сообщения. Для ее реализации должен быть известен как алгоритм сокрытия, так и секретный ключ. Для задачи извлечения скрытого сообщения достаточно знать алгоритм сокрытия. Последней и наиболее легко реализуемой задачей является разрушение скрытого сообщения. Для ее решения не надо знать ни алгоритма сокрытия, ни стегоключа.

1.2. Алгоритмы определения закладки в контейнере

Принципиально невозможно описать все множество незаполненных контейнеров. Посильными задачами представляются только выявление особенностей заполненных контейнеров и задача степени соответствия параметров исследуемого графического файла параметрам изображения с вложенной стеганографической закладкой. Для более точного обнаружения необходимо использовать несколько методов, анализирующих разные особенности контейнеров. Рассмотрим алгоритмы, которые были запрограммированы и протестированы. Они позволяют проводить стегоанализ файлов формата BMP с 24-разрядным кодированием цвета.

Первые три метода рассматривают множество переходов (различий между двумя точками в изображении).

Четвертый метод позволяет извлечь сообщение, соответствующее скрытому, но для этого мы должны знать алгоритм сокрытия сообщения и стегоключ. Он основан на поиске в извлеченном сообщении сигнатур различных форматов файлов, из чего делается вывод о формате сообщения. Остальные два метода являются графическими, они позволяют исследовать содержимое графического файла.

Назовем первый алгоритм критерием несогласованности переходов, второй – критерием частот переходов (названия определяются принципом работы алгоритмов), третий – критерием серий (это название заимствованно из математической статистики), а четвертый – методом сигнатур. Графические методы – выделения несоответствия и минимального множества точек (по принципу их работы).

2. Теоретические основы стегоанализа графических файлов

2.1 Особенности стеганографии графических файлов

Давайте рассмотрим, как работает программа S-Tools, а потом сделаем выводы. Суть примененного здесь метода стеганографии (а по-русски будет "скрытопись") базируется на том, что оцифрованные данные (изображение или звук) принципиально не могут быть точны. Всегда существует погрешность дискретизации, равная значению младшего бита отсчета. Так как эти значения малы, то органы чувств человека неспособны уловить изменение сигнала на такую величину, поэтому вполне возможно сохранить данные, "размазав" их по младшим, наименее значимым битам оцифрованных данных.

Пусть имеется число 180, в двоичном коде оно выглядит так: 10110100. Спрячем его в последовательности из восьми байт, приведенной в первой колонке таблицы. Для этого заменим в двоичном представлении чисел последовательности (вторая колонка), младшие биты (выделены красным) битами нашего числа. Получим третью колонку таблицы, десятичное представление чисел которой запишем в четвертой колонке.

Исходные значения (десятичные)	Двоичное представление	Последовательность после замены	Десятичные значения после замены
135	10000111	10000111	135
121	01111001	01111000	120
120	01111000	01111001	121
107	01101011	01101011	107
143	10001111	10001110	142
98	01100010	01100011	99
103	01100111	01100110	102
102	01100110	01100110	102

Как видно из полученного результата, изменение значений последовательности не превышает единицы. Если взять 24 - битное кодирование цветов (для хранения каждого элемента используется три байта), то объем информации, который можно "запрятать" в картинку: $(640 \times 480 \times 3) / 8 = 115200$ байт. Правда, объем самого изображения в формате BMP велик (921600 байт), что, по мнению автора программы, само по себе может привлечь внимание противника. Но, очевидно, сжимать изображение каким-либо методом,

вносящим потери, нельзя, так как информация будет безвозвратно утеряна - возможно, лишь запаковать его архиватором. Такая же схема применяется и для звука. Для изображений формата GIF (число цветов в которых принципиально не может превышать 256) программа предлагает либо преобразовать его в 24-битное, либо уменьшить число цветов до 32. В последнем случае "размазывание" данных будет сделано по трем младшим битам каждого байта, но, качество 32-цветной картинки явно неудовлетворительное. Чтобы немного повысить степень серьезности этой затеи, избавив ее от указанных выше недостатков, предлагается: спрятать данные в файл очень распространенного формата JPEG. Но JPEG ведь вносит потери, которые так "зашифруют" всю информацию, что восстановить ее не получится даже на квантовом компьютере - теоретическом изделии, обладающем огромными возможностями параллельных вычислений. Однако выход найден - он кроется в механизме сжатия по алгоритму JPEG, состоящему из двух стадий. Первая стадия - дискретное косинусное преобразование, после которого и возникают потери данных. Вторая стадия - сжатие по методу Хаффмана - потерь не вносит. Вот между этими стадиями и происходит вставка секретных данных.

2.2 Формат графических файлов JPEG

Этапы работы алгоритма JPEG

1) *Плавное преобразование цветового пространства: [R G B] -> [Y Cb Cr]*

(R, G, B - 8-битовые величины без знака)

$$\begin{aligned}
 |Y| &= |0.299 R + 0.587 G + 0.114 B| \\
 |Cb| &= |-0.1687 R - 0.3313 G + 0.5 B| + |128| \\
 |Cr| &= |0.5 R - 0.4187 G - 0.0813 B| + |128|
 \end{aligned}$$

Новая величина $Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$ названа яркостью. Это - величина, использованная монохромными мониторами, чтобы представить цвет RGB. Физиологически, передает интенсивность цвета RGB воспринятого глазом. Вы видите, что формула для Y, подобно средневзвешенному значению с

разным весом для каждого спектрального компонента: глаз наиболее чувствителен на Зеленый цвет, затем следует Красный компонент и в последнюю очередь - Синий.

Величины $C_b = -0.1687 * R - 0.3313 * G + 0.5 * B + 128$

и $C_r = 0.5 * R - 0.4187 * G - 0.0813 * B + 128$

названы цветовыми величинами и представляют 2 координаты в системе, которая измеряет оттенок и насыщение цвета ([Приблизленно], эти величины указывают количество синего и красного в этом цвете).

Эти две координаты кратко названы цветоразностью.

Преобразование $[Y, C_b, C_r]$ в $[R, G, B]$ (обратно предыдущему преобразованию) RGB-цвет может быть вычислен непосредственно из YCbCr (8-битовые величины без знака) следующим образом:

$R = Y + 1.402 * (C_r - 128)$

$G = Y - 0.34414 * (C_b - 128) - 0.71414 * (C_r - 128)$

$B = Y + 1.772 * (C_b - 128)$

Примечание, связывающее Y, C_b, C_r в человеческой визуальной системе:

глаз, особенно сетчатка, имеет как визуальные анализаторы два типа ячеек: ячейки для ночного видения, воспринимающие только оттенки серого (от ярко-белого до темно-черного) и ячейки дневного видения, которые воспринимают цветовой оттенок. Первые ячейки, дающие цвет RGB, обнаруживают уровень яркости, подобный величине Y . Другие ячейки, ответственные за восприятие цветового оттенка, - определяют величину, связанную с цветоразностью.

2) Дискретизация

JPEG Стандарт принимает во внимание то, что глаз более чувствителен к яркости цвета, чем к оттенку этого цвета. (Черно-белые ячейки вида имеют больше влияния, чем ячейки дневного видения)

Так, для большинства JPG, яркость взята для каждого пикселя, тогда как цветоразность – как средняя величина для блока 2x2 пикселей. Однако это не обязательно, но при этом можно достичь хороших результатов сжатия, с

незначительным убытком в визуальном восприятии нового обработанного изображения.

Примечание: JPEG стандарт определяет, что для каждого компонента образа (подобно, например Y) должно быть определено 2 коэффициента дискретизации: один для горизонтальной дискретизации и один для вертикальной дискретизации. Эти коэффициенты дискретизации определяются в файле JPG как относительно максимального коэффициента дискретизации (дополнительно об этом позже).

3) Сдвиг Уровня

Все 8-битовые величины без знака (Y, Cb, Cr) в изображении - "смещенные по уровню": они преобразовываются в 8-битовое знаковое представление вычитанием 128 из их величины.

4) 8x8 Дискретное Косинусоидальное Преобразование (DCT)

Изображение делится на блоки 8x8 пикселей, затем для каждого блока 8x8 применяется DCT-трансформация. Заметьте, что если размер X исходного образа не делится на 8, шифратор должен сделать его делимым, дополняя остальные правые столбцы (пока X не станет кратным 8). Аналогично, если размер Y не делимо на 8, шифратор должен дополнить строки.

Блоки 8x8 обрабатываются слева направо и сверху вниз.

Примечание: Поскольку каждый пиксель в блоке 8x8 имеет 3 компонента (Y, Cb, Cr), DCT приложен отдельно в трех блоках 8x8:

Первый блок 8x8 является блоком, который содержит яркость пикселей в исходном блоке 8x8;

Второй блок 8x8 является блоком, который содержит величины Cb;

И, аналогично, третий блок 8x8 содержит величины Cr.

Цель DCT-трансформации в том, что вместо обработки исходных изображений, Вы работаете с пространством частот изменения яркости и оттенка. Эти частоты очень связаны с уровнем детализации изображения. Высокие частоты соответствуют высокому уровню детализации.

DCT-трансформация очень похожа на 2-мерное преобразование Фурье, которое получает из временного интервала (исходный блок 8x8) частотный интервал (новые коэффициенты 8x8=64, которые представляют амплитуды проанализированного частотного пространства)

Математическое определение прямого DCT (FDCT) и обратного DCT (IDCT):

FDCT:

$$F(u, v) = \frac{c(u, v)}{4} * \sum_{x=0}^7 \sum_{y=0}^7 f(x, y) * \cos\left(\frac{2x+1}{16} * u * \pi\right) * \cos\left(\frac{2y+1}{16} * v * \pi\right)$$

$u, v = 0, 1, \dots, 7$

$c(u, v) = 1/2$, когда $u=v=0$;

$c(u, v) = 1$ – в остальных случаях.

IDCT:

$$f(x, y) = \frac{1}{4} * \sum_{u=0}^7 \sum_{v=0}^7 c(u, v) * F(u, v) * \cos\left(\frac{2x+1}{16} * u * \pi\right) * \cos\left(\frac{2y+1}{16} * v * \pi\right)$$

$x, y = 0, 1 \dots 7$

5) Зигзагообразная перестановка 64 DCT коэффициентов

Так, после того, как выполнено DCT-преобразование над блоком величин 8x8, есть новый блок 8x8. Затем, этот блок 8x8 просматривается по зигзагу подобно этому:

(Числа в блоке 8x8 указывают порядок, в котором мы просматриваем 2-мерную матрицу 8x8)

0, 1, 5, 6, 14, 15, 27, 28,
 2, 4, 7, 13, 16, 26, 29, 42,
 3, 8, 12, 17, 25, 30, 41, 43,
 9, 11, 18, 24, 31, 40, 44, 53,
 10, 19, 23, 32, 39, 45, 52, 54,
 20, 22, 33, 38, 46, 51, 55, 60,

21,34,37,47,50,56,59,61,
35,36,48,49,57,58,62,63

Как видно, сначала - верхний левый угол (0,0), затем величина в (0,1), затем (1,0), затем (2,0), (1,1), (0,2), (0,3), (1,2), (2,1), (3,0) и т.п.

После прохождения по зигзагу матрицу 8x8, получается теперь вектор с 64 коэффициентами (0..63) Смысл этого зигзагообразного вектора – в том, что мы просматриваем коэффициенты 8x8 DCT в порядке повышения пространственных частот. Так, мы получаем вектор, отсортированный критериями пространственной частоты: первая величина на векторе (индекс 0) соответствует самой низкой частоте в изображении – она обозначается термином DC. С увеличением индекса на векторе, мы получаем величины соответствующие высшим частотам (величина с индексом 63 соответствует амплитуде самой высокой частоте в блоке 8x8). Остальная часть коэффициентов DCT обозначается AC.

б) Квантование

На этом этапе, у нас есть отсортированный вектор с 64 величинами, соответствующими амплитудам 64 пространственных частот в блоке 8x8.

Эти 64 величины квантованы: Каждая величина делится на число, определенное для вектора с 64 величинами - таблицу квантования, затем округляется до ближайшего целого.

для (i = 0; i<=63; i++)

вектор[i] = (округлить) (вектор[i] / таблица_квантования[i] + 0.5)

Вот пример таблицы квантования для яркости(Y) данной в приложении JPEG стандарта. (Дается в форме блока 8x8; полученного из 64 векторных величин, зигзагообразным преобразованием)

16 11 10 16 24 40 51 61

12 12 14 19 26 58 60 55

14 13 16 24 40 57 69 56

14 17 22 29 51 87 80 62
18 22 37 56 68 109 103 77
24 35 55 64 81 104 113 92
49 64 78 87 103 121 120 101
72 92 95 98 112 100 103 99

Эта таблица опирается на "психовизуальный порог", это "используется с хорошими результатами для изображений с 8-битовой яркостью и оттенками". Большинство существующих шифраторов просто копируют этот пример, но величины не оптимизируются, (шифратор может использовать ЛЮБУЮ ДРУГУЮ таблицу квантования) таблица определяется в JPEG файле с DQT (Определение Таблицы Квантования) маркером. Обычно присутствует одна таблица для Y, и другие для оттенка (Cb и Cr).

Процесс квантования играет ключевую роль в JPEG сжатии. Это - процесс, который удаляет высокие частоты, представленные в исходном изображении - впоследствии высокую детализацию. Мы делаем это из-за того, что глаз более чувствителен к низким частотам, чем к высоким, так что мы можем удалить, с очень небольшим визуальным убытком, высокие частоты. Это сделано посредством деления величин в высоких индексах на векторе (амплитуды высоких частот) на большие величины, чем величины, на которыми разделены амплитуды более низких частот. Больше величины в таблице квантования - больше потери (впоследствии визуальные потери) введенные этим процессом, и меньше - лучше визуальное качество.

Другой важный факт - в большинстве изображений цвет изменяется медленно от одного пикселя к другому, так что большинство образов будут иметь небольшое количество высокой детализации -> небольшая сумма (небольшие амплитуды) высоких пространственных частот - но у них есть много информации об изображении, содержащейся на низких пространственных частотах.

Впоследствии на новом квантованном векторе, на высоких пространственных частотах, мы будем иметь много последовательных нулей.

7) *RunLength кодирование нулей (RLC)*

Теперь у нас есть квантованный вектор с длинной последовательностью нулей. Мы можем использовать это, кодируя последовательные нули. ВАЖНО: Вы увидите позже почему, но здесь мы пропускаем кодировку первого коэффициента вектора (коэффициент DC), который закодирован по-другому. (Я представлю его кодирование позже в этом документе) Рассмотрим исходный 64 вектор как 63 вектор (это - 64 вектор без первого коэффициента)

Допустим, мы имеем 57,45,0,0,0,0,23,0,-30,-16,0,0,1,0,0,0,0,0, только 0,...,0

Здесь - как RLC JPEG сжатие сделано для этого примера:

(0,57); (0,45); (4,23); (1,-30); (0,-16); (2,1); EOB

Как видно, мы кодируем для каждой величины, отличающейся от 0 количество последовательных ПРЕДШЕСТВУЮЩИХ нулей перед величиной, затем мы добавляем величину. Другое примечание: EOB - короткая форма для Конца Блока, это - специальная закодированная величина (маркер). Если мы достигли в позиции на векторе, от которого мы имеем до конца только нули вектора, мы выделим эту позицию с EOB и завершим сжатие RLC квантованного вектора.

[Заметьте, что если квантованный вектор не оканчивается нулями (имеет последний элемент не 0), мы не будем иметь маркер EOB.]

(0,57); (0,45); (4,23); (1,-30); (0,-16); (2,1); (0,0)

Допустим, где-нибудь на квантованном векторе мы имеем:

57, восемнадцать нулей, 3, 0,0 ,0,0 2, тридцать-три нуля, 895, EOB

Кодирование Хаффмана JPG делает ограничение, по которому число предшествующих нулей должно кодироваться как 4-битовая величина - не может превысить 15.

Так, предшествующий пример должен быть закодирован как:

(0,57); (15,0) (2,3); (4,2); (15,0) (15,0) (1,895), (0,0)

(15,0) - специальная кодированная величина, которая указывает, что там следует за 16 последовательными нулями. Примечание: 16 нулей не 15 нулей.

8) Конечный шаг - кодирование Хаффмана

Сначала ВАЖНОЕ примечание: Вместо хранения фактической величины, JPEG стандарт определяет, что мы храним минимальный размер в битах, в котором мы можем держать эту величину (это названо категория этой величины) и затем битно кодированное представление этой величины подобно этому:

Величины	Категория	Биты для величины
0	0	-
-1,1	1	0,1
-3,-2,2,3	2	00,01,10,11
-7,...,-4,4,...,7	3	000,001,010,011,100,101,110,111
-15,...,-8,8,...,15	4	0000,...,0111,1000,...,1111
-31,...,-16,16,...,31	5	00000,...,01111,10000,...,11111
-63,...,-32,32,...,63	6	.
-127,...,-64,64,...,127	7	.
-255,...,-128,128,...,255	8	.
-511,...,-256,256,...,511	9	.
-1023,...,-512,512,...,1023	10	.
-2047,...,-1024,1024,...,2047	11	.
-4095,...,-2048,2048,...,4095	12	.
-8191,...,-4096,4096,...,8191	13	.
-16383,...,-8192,8192,...,16383	14	.
-32767,...,-16384,16384,...,32767	15	.

Впоследствии для предшествующего примера:

(0,57); (0,45); (4,23); (1,-30); (0,-8); (2,1); (0,0)

давайте закодируем ТОЛЬКО правую величину этих пар, кроме пар, которые являются специальными маркерами подобно (0,0) или (если мы должны иметь) (15,0)

57 - в категории 6 и это – битно кодированный 111001, так что мы закодируем это как (6,111001)

45, аналогично, будет закодирован как (6,101101)

23 -> (5,10111)

-30 -> (5,00001)

-8 -> (4,0111)

1 -> (1,1)

И теперь, мы напишем снова строку пар:

(0,6), 111001; (0,6), 101101; (4,5), 10111; (1,5), 00001; (0,4), 0111; (2,1), 1; (0,0)

Пары двух величин, заключенные в скобки, могут быть представлены в байте, так как фактически каждая из 2 величин может быть представлена в 4-битном кусочке (счетчик предшествующих нулей - всегда меньше, чем 15 и также как и категория [числа, закодированные в файле JPG - в области -32767..32767]). В этом байте, старший кусочек представляет число предшествующих нулей, а младший кусочек - категорию новой величины, отличной от 0.

КОНЕЧНЫЙ шаг кодировки состоит в кодировании Хаффмана этого байта, и затем записи в файле JPG, как поток из битов, кода Хаффмана этого байта, сопровождающийся битовым представлением этого числа.

Например, для байта 6 (эквивалент (0,6)) у нас есть код Хаффмана = 111000;

для байта 69 = (4,5) (например) у нас есть 111111110011001

21 = (1,5) — 11111110110

4 = (0,4) — 1011

33 = (2,1) — 11011

0 = EOB= (0,0) — 1010

Конечный поток битов записанных в файле JPG на диск для предшествующего примера 63 коэффициентов (запомните, что мы пропустили первый коэффициент) -

111000 111001 111000 101101 111111110011001 10111 11111110110 00001
1011 0111 11011 1 1010

Кодирование коэффициента DC

DC является коэффициентом на квантованном векторе соответствующем самой низкой частоте в образе (это - 0 частота), и (перед квантованием) - математически = (сумма 8x8 составляющих)/8. (Это - подобно средней величине для этого блока образцов изображения). Сказано, что он содержит много энергии, присутствующей в исходном изображении блока 8x8. (Обычно, это большие величины). Авторы JPEG стандарта обращали внимание, что есть очень близкое соотношение между коэффициентом DC последовательных блоков, так что они решили кодировать в файле JPG различие между DC последовательными блоками 8x8 (Примечание: последовательные блоки 8x8 одинаковых компонентов изображения, подобно последовательным блокам 8x8 для Y, или последовательные блоки для Cb, или для Cr)

$Diff = DC_i - DC_{i-1}$; Так DC текущего блока (DC_i) равен: $DC_i = DC_{i-1} + Diff$

Декодирование JPG начинается с 0 - рассматривается первым коэффициентом $DC = 0$; $DC_0 = 0$

А затем добавляется к текущей величине величину, декодированную из JPG (Diff величину)

Так, в файле JPG, первый коэффициент, равный коэффициенту DC является действительно отличительным, и это – Хаффман-кодирование отличается от кодирования коэффициентов AC.

Diff предшествует, как Вы видели, представлению категории и, это – бит-кодированное представление. В файле JPG будет Хаффманом закодирована только величина категории, подобно этому:

$Diff = (\text{Категория, битно кодированное представление})$

Затем Diff будет закодирован как (код_Хаффмана (категория), битно кодированное представление)

Например, если Diff равняется -511, тогда Diff соответствует

(9, 000000000)

Допустим, что 9 имеет код Хаффмана = 1111110 (В файле JPG, есть 2 таблицы Хаффмана для каждого компонента изображения: для DC и для AC)

В файле JPG, биты соответствующие коэффициенту DC будут:

1111110 000000000

И, относившийся к этому примеру DC и в предшествующем примере AC, для этого вектора с 64 коэффициентами, КОНЕЧНЫЙ ПОТОК БИТОВ, записанных в файле JPG, будет:

1111110 000000000 111000 111001 111000 101101 1111111110011001 10111
11111110110 00001 1011 0111 11011 1 1010

(В файле JPG, сначала закодировано DC, затем AC)

2.3 Анализ существующих методов стегоанализа

2.3.1 Угрозы безопасности стегосистем

Пассивный взломщик может лишь обнаружить факт наличия стегоканала и (возможно) читать сообщения. Сможет ли он прочесть сообщение после его обнаружения, зависит от стойкости системы шифрования, и этот вопрос, как правило, не рассматривается в стеганографии. Если у имеется возможность выявить факт наличия скрытого канала передачи сообщений, то стегосистема обычно считается нестойкой. Хотя существуют и другие точки зрения на стойкость стегосистем. Осуществление обнаружения стегоканала является наиболее трудоемкой задачей, а защита от обнаружения считается основной задачей стеганографии, по определению. Диапазон действий активного нарушителя значительно шире. Скрытое сообщение может быть им удалено или разрушено. В большинстве случаев это противоречит интересам сторон

(например, по юридическим мотивам). Другое дело – удаление или разрушение цифрового водяного знака, которые могут рассматриваться как основные угрозы в этой области

Действия злоумышленного нарушителя наиболее опасны. Он способен не только разрушать, но и создавать ложные стего. История противостояния разведки и контрразведки знает немало примеров, когда реализация этой угрозы приводило к катастрофическим последствиям. Эта угроза актуальна и по отношению к системам ЦВЗ. Обладая способностью создавать водяные знаки, нарушитель может создавать копии защищаемого контента, создавать ложные оригиналы и т.д. Во многих случаях нарушитель может создавать ложные стего без знания ключа.

Для осуществления той или иной угрозы нарушитель применяет атаки. Наиболее простая атака – субъективная. Субъект внимательно рассматривает изображение (слушает аудиозапись), пытаясь определить “на глаз”, имеется ли в нем скрытое сообщение. Ясно, что подобная атака может быть проведена лишь против совершенно незащищенных стегосистем. Тем не менее, она, наверное, наиболее распространена на практике, по крайней мере, на начальном этапе вскрытия стегосистемы. Первичный анализ также может включать в себя следующие мероприятия:

- Первичная сортировка стего по внешним признакам.
- Выделение стего с известным алгоритмом встраивания.
- Определение использованных стегоалгоритмов.
- Проверка достаточности объема материала для стегоанализа.
- Проверка возможности проведения анализа по частным случаям.
- Аналитическая разработка стегоматериалов. Разработка методов вскрытия стегосистемы.
- Выделение стего с известными алгоритмами встраивания, но неизвестными ключами и т.д.

Из криптоанализа нам известны следующие разновидности атак на зашифрованные сообщения:

- атака с использованием только шифртекста;
- атака с использованием открытого текста;
- атака с использованием выбранного открытого текста;
- адаптивная атака с использованием открытого текста;
- атака с использованием выбранного шифртекста.

По аналогии с криптоанализом в стегоанализе можно выделить следующие типы атак.

- Атака на основе известного заполненного контейнера. В этом случае у нарушителя есть одно или несколько стего. В последнем случае предполагается, что встраивание скрытой информации осуществлялось субъектом одним и тем же способом. Задача субъекта может состоять в обнаружении факта наличия стегоканала (основная), а также в его извлечении или определении ключа. Зная ключ, нарушитель получит возможность анализа других стегосообщений.
- Атака на основе известного встроенного сообщения. Этот тип атаки в большей степени характерен для систем защиты интеллектуальной собственности, когда в качестве водяного знака используется известный логотип фирмы. Задачей анализа является получение ключа. Если соответствующий скрытому сообщению заполненный контейнер неизвестен, то задача крайне трудно решается.
- Атака на основе выбранного скрытого сообщения. В этом случае имеется возможность предлагать для передачи свои сообщения и анализировать получающиеся стего.
- Адаптивная атака на основе выбранного скрытого сообщения. Эта атака является частным случаем предыдущей. В данном случае имеется возможность выбирать сообщения для навязывания субъекту адаптивно, в

зависимости от результатов анализа предыдущих стего.

- Атака на основе выбранного заполненного контейнера. Этот тип атаки больше характерен для систем ЦВЗ. Стегоаналитик имеет детектор стего в виде «черного ящика» и несколько стего. Анализируя детектируемые скрытые сообщения, нарушитель пытается вскрыть ключ.
- Может иметься возможность применить еще три атаки, не имеющие прямых аналогий в криптоанализе.
- Атака на основе известного пустого контейнера. Если он известен, то путем сравнения его с предполагаемым стего он всегда может установить факт наличия стегоканала. Несмотря на тривиальность этого случая, в ряде работ приводится его информационно-теоретическое обоснование. Гораздо интереснее сценарий, когда контейнер известен приблизительно, с некоторой погрешностью (как это может иметь место при добавлении к нему шума).
- Атака на основе выбранного пустого контейнера. Например, предложенный контейнер может иметь большие однородные области (однотонные изображения), и тогда будет трудно обеспечить секретность внедрения.
- Атака на основе известной математической модели контейнера или его части. При этом атакующий пытается определить отличие подозрительного сообщения от известной ему модели. Например допустим, что биты внутри отсчета изображения коррелированы. Тогда отсутствие такой корреляции может служить сигналом об имеющемся скрытом сообщении. Задача внедряющего сообщение заключается в том, чтобы не нарушить статистики контейнера. Внедряющий и атакующий могут располагать различными моделями сигналов, тогда в информационно-скрывающем противоборстве победит имеющий лучшую модель.

2.3.2 Визуальная атака на стегосистемы

Рассмотрим принцип построения визуальной атаки, позволяющей выявить факт наличия скрываемого сообщения, вложенного в изображение-контейнер [14]. Пусть стегосистема построена таким образом, что НЗБ элементов изображения заменяются на биты скрываемого сообщения. Например, в системе EzStego младший бит цветовой компоненты каждого пиксела, начиная от начала изображения, последовательно заменяется соответствующим битом скрываемого сообщения. В других стегосистемах биты внедряемого сообщения замещают младшие биты яркостной компоненты каждого пиксела изображения. Ранее считалось, что НЗБ яркостной или цветовой компонент пикселов изображения, равно как и младшие биты отсчетов речевых или аудиосигналов независимы между собой, а также независимы от остальных битов элементов рассматриваемых контейнеров. Однако на самом деле это не так. Младшие биты не являются чисто случайными. Между младшими битами соседних элементов естественных контейнеров имеются существенные корреляционные связи. Также выявлены зависимости между НЗБ и остальными битами элементов естественных контейнеров.

На рис. 2.1 показано изображение мельницы, слева рисунок представляет пустой контейнер, справа в каждый НЗБ цветовой компоненты пикселов последовательно бит за битом вложено скрываемое сообщение. Различие между контейнером и стего визуально не проявляется. Но если изображение сформировать только из НЗБ пикселов стего, то можно легко увидеть следы вложения. На рис. 2.1 слева показано изображение, состоящее из НЗБ пустого контейнера. Видно, что характер изображения существенно не изменился. Справа представлено изображение из младших битов наполовину заполненного скрываемым сообщением контейнера. Видно, что верхняя часть изображения, куда внедрено сообщение, представляет собой случайный сигнал. В рассматриваемой стегосистеме скрываемое сообщение до встраивания зашифровывается, поэтому каждый его бит практически равновероятен и

независим от соседних битов, что позволяет легко визуально выявить факт его встраивания, сопоставляя изображения из младших битов стего и пустых естественных контейнеров, соответственно. В некоторых стегосистемах сообщения до встраивания сжимаются. Это целесообразно как для уменьшения размера скрытно внедряемой информации, так и для затруднения его чтения посторонними лицами. Архиваторы данных преобразуют сжимаемое сообщение в последовательность битов, достаточно близкую к случайной. Чем выше степень сжатия, тем ближе последовательность на выходе архиватора к случайной, и тем проще обнаружить факт существования стегоканала при визуальной атаке. Однако даже если скрываемое сообщение до встраивания не шифруется и не сжимается, то его вероятностные характеристики не совпадают с вероятностными характеристиками НЗБ используемых контейнеров, что опять таки можно выявить. Заметим, что отправитель сообщения может подобрать контейнер с законом распределения, совпадающим с законом распределения конкретного встраиваемого сообщения. В этом случае визуальная атака, как и статистические атаки, неэффективна. Но трудности подбора требуемого контейнера могут сделать такую стегосистему непрактичной.

В известной программе Steganos встраивание сообщения любой длины осуществляется во все НЗБ пикселей контейнера, поэтому выявляется визуальной атакой.

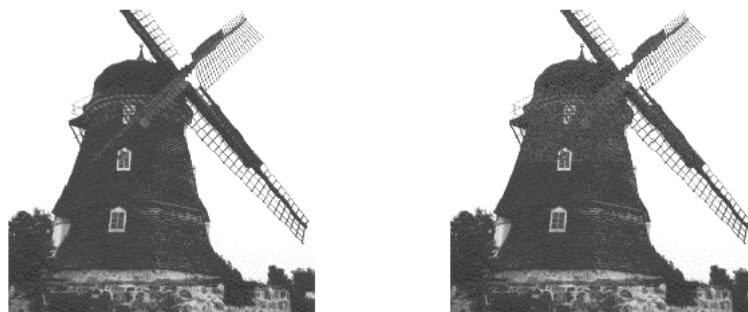


Рис. 2.1. Изображение мельницы, слева – пустой контейнер, справа - с вложенным сообщением



Рис. 2.2. Визуальная атака на EzStego, слева – изображение из НЗБ пустого контейнера, справа – изображение из НЗБ наполовину заполненного

Визуальная атака целиком основана на способности зрительной системы человека анализировать зрительные образы и выявлять существенные различия в сопоставляемых изображениях. Визуальная атака эффективна при полном заполнении контейнера, но по мере уменьшения степени его заполнения глазу человека все труднее заметить следы вложения среди сохраненных элементов контейнера.

В ряде стеганографических систем элементы скрываемого сообщения вкладываются в младшие биты коэффициентов преобразования Фурье контейнера-изображения. Например, 8×8 пикселей $f(x, y)$ блока изображения сначала преобразовываются в 64 коэффициента дискретного косинусного преобразования (ДКП) по правилу

$$F(u, v) = \frac{1}{4} C(u) C(v) \left[\sum_{x=0}^7 \sum_{y=0}^7 f(x, y) \cdot \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16} \right],$$

где $C(u)$ и $\tilde{N}(v) = \frac{1}{\sqrt{2}}$ когда u и v равны нулю и $C(u), C(v) = 1$ в других случаях.

Полученные коэффициенты квантуются с округлением до целого:

$$F^Q(u, v) = \text{Integer_Round} \left(\frac{F(u, v)}{Q(u, v)} \right),$$

где $Q(u, v)$ есть таблица квантования из 64 элементов.

Наименьшие значащие биты квантования ДКП коэффициентов, за исключением $F^Q(u, v) = 0$ и $F^Q(u, v) = 1$, в стегосистеме являются избыточными битами и вместо них внедряются биты скрываемого сообщения.

Против таких методов скрытия визуальная атака малопригодна, так как изменение любого коэффициента преобразования приводит к изменению множества пикселей изображения. Например, в программе JSteg преобразование выполняется над матрицей 16×16 пикселей контейнера. Следовательно, вложение скрываемого сообщения в младшие биты коэффициентов преобразования приведет к сравнительно небольшим изменениям каждого из 256 пикселей, что визуально незаметно.

Поэтому рассмотрим второй класс практических стегоатак с целью обнаружения скрытого канала передачи информации, основанный на анализе различий между статистическими характеристиками естественных контейнеров и сформированных из них стего.

2.3.3. Статистические атаки на стегосистемы с изображениями-контейнерами

Одним из наиболее перспективных подходов для выявления факта существования скрытого канала передачи информации является подход, представляющий введение в файл скрываемой информации как нарушение статистических закономерностей естественных контейнеров. При данном подходе анализируются статистические характеристики исследуемой последовательности и устанавливается, похожи ли они на характеристики естественных контейнеров (если да, то скрытой передачи информации нет), или они похожи на характеристики стего (если да, то выявлен факт существования

скрытого канала передачи информации). Этот класс стегоатак является вероятностным, то есть они не дают однозначного ответа, а формируют оценки типа “данная исследуемая последовательность с вероятностью 90% содержит скрываемое сообщение”. Вероятностный характер статистических методов стегоанализа не является существенным недостатком, так как на практике эти методы часто выдают оценки вероятности существования стегоканала, отличающиеся от единицы или нуля на бесконечно малые величины.

Класс статистических методов стегоанализа использует множество статистических характеристик, таких как оценка энтропии, коэффициенты корреляции, вероятности появления и зависимости между элементами последовательностей, условные распределения, различимость распределений по критерию Согласия и многие другие. Самые простые тесты оценивают корреляционные зависимости элементов контейнеров, в которые могут внедряться скрываемые сообщения. Для выявления следов канала скрытой передачи информации можно оценить величину энтропию элементов контейнеров. Стего, содержащие вложение скрываемых данных, имеют большую энтропию, чем пустые естественные контейнеры.

Рассмотрим атаку на основе анализа статистики. В программе EzStego младший бит цветовой компоненты каждого пиксела контейнера-изображения заменяется битом скрываемого сообщения. Исследуем закономерности в вероятностях появления значений цветовой компоненты в естественных контейнерах и сформированных программой EzStego стего. При замене младшего бита цветовой компоненты очередного пиксела контейнера на очередной бит предварительно зашифрованного или сжатого сообщения номер цвета пиксела стего или равен номеру цвета пиксела контейнера, или изменяется на единицу. Для поиска следов вложения предложен метод анализа закономерностей в вероятностях появления соседних номеров цвета пикселей. Номер цвета, двоичное представление которого заканчивается нулевым битом, назовем левым (L), а соседний с ним номер цвета, двоичное представление

которого заканчивается единичным битом - правым (R). Пусть цветовая гамма исходного контейнера включает 8 цветов. Следовательно, при встраивании сообщения в НЗБ цветовой компоненты пикселей необходимо исследовать статистические характеристики в 4 парах номеров цвета. На рис. 2.3 слева показана одна из типичных гистограмм вероятностей появления левых и правых номеров цвета в естественных контейнерах. Справа показана гистограмма вероятностей появления левых и правых номеров цвета в стего, сформированного из этого контейнера программой EzStego. Видно, что вероятности появления левых и правых номеров цвета в естественных контейнерах существенно различаются между собой во всех парах, а в стего эти вероятности выровнялись. Это является явным демаскирующим признаком наличия скрываемой информации. Заметим, что среднее значение вероятностей для каждой пары в стего не изменилось по сравнению с контейнером (показано на рис. 2.3 пунктирной линией).

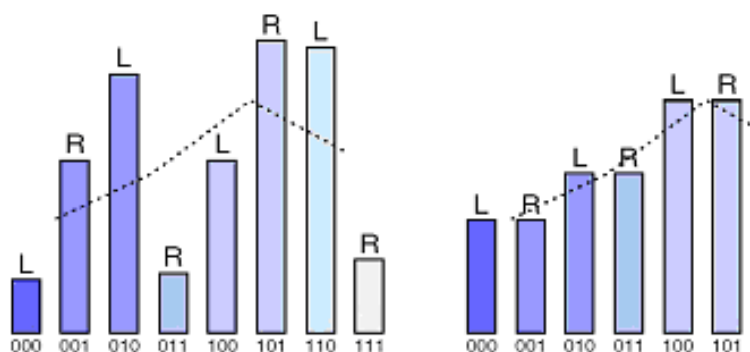


Рис. 2.3 Гистограмма частот появления левых и правых номеров цвета, слева – до встраивания, справа – после.

При замещении битами внедряемого сообщения младших битов яркостной компоненты пикселей контейнера-изображения проявляются аналогичные статистические различия.

Степень различия между вероятностными распределениями элементов естественных контейнеров и полученных из них стего может быть использована для оценки вероятности существования стегоканала. Данную вероятность удобно определить с использованием критерия согласия. По критерию согласия

сравнивается, насколько распределение исследуемой последовательности близко к характерному для стегограмм распределению. В исследуемой последовательности подсчитывается сколько раз n_i ее элемент x_i принял рассматриваемые значения, где всего k элементов. Например, в гистограмме левых и правых номеров цвета в левой части рис.2.4 номер цвета 000 появился 2 раза ($n_0^* = 2$), а номер 001 – 5 раз ($n_1^* = 5$). При встраивании очередных битов скрываемого сообщения в НЗБ этой пары номер цвета 000 должен появляться в среднем n_0 раз

$$n_0 = \frac{n_0^* + n_1^*}{2} .$$

Зная общее число n появления всех элементов исследуемой последовательности, легко подсчитать ожидаемую вероятность появления этих элементов в стего по правилу: $p_i = n_i / n$. Соответственно, для исследуемой последовательности вероятности равны: $p_i^* = n_i^* / n$.

Величина критерии согласия для сравниваемых распределения исследуемой последовательности и ожидаемого распределения стего равна

$$\chi^2 = \sum_{i=1}^v \frac{(n_i - np_i)^2}{np_i} ,$$

где v есть число степеней свободы. Число степеней свободы равно числу k минус число независимых условий, наложенных на вероятности p_i^* . Наложим одно условие вида

$$\sum_{i=1}^k p_i^* = 1 .$$

Вероятность p того, что два распределения одинаковы, определяется как

$$p = 1 - \int_0^{\chi^2} \frac{t^{(v-2)/2} e^{-t/2}}{2^{v/2} \Gamma(v/2)} dt ,$$

где Γ есть гамма-функция Эйлера.

Чем больше значение p , тем выше вероятность встраивания скрываемой информации в исследуемую последовательность.

Рассмотрим использование критерия согласия для отыскания следов стегоканала, образованного с использованием программы EzStego. Пусть в контейнерное изображение "Мельница", показанное в левой части рис. 1, в НЗБ спектральных коэффициентов изображения, начиная с его верхнего края до его середины, последовательно внедрено 3600 байт скрываемого сообщения. На рис. 2.4 показана вероятность встраивания скрываемой информации в зависимости от размера исследуемой последовательности. Начало графика получено при анализе первого фрагмента стего, составляющего одну сотую часть всего стего. Значение p составило 0,8826. Затем к анализируемому фрагменту была добавлена еще одна сотая часть стего, и так далее. На втором шаге вероятность составила 0,9808 и далее при анализе стего не опускалась ниже 0,77. При переходе к анализу нижней части изображения, не содержащей скрываемой информации, величина p скачком уменьшилась до нуля.

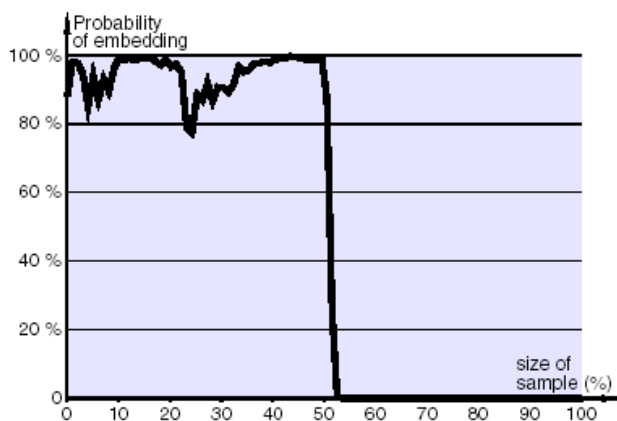


Рис. 2.4. Вероятность встраивания по критерию согласия при анализе EzStego.

В программе Steganos встраиваемое двоичное сообщение любой длины дополняется до длины контейнера (до числа пикселей изображения). Поэтому критерий Согласия при встраивании сколь угодно малого сообщения с

использованием Steganos дает вероятность существования стегоканала, практически не отличающуюся от единицы.

В программе S-Tools встраиваемое сообщение равномерно распределяется по всему контейнеру. При полностью заполненном контейнере по критерию Согласия уверенно выявляются следы вложения посторонней информации с пренебрежимо малой вероятностью ошибки (менее 10^{-16}), но при заполненном контейнере на треть и менее следы стегоканала не выявляются.

Как и в EzStego, в программе JSteg скрываемое сообщение последовательно встраивается в коэффициенты преобразования контейнера. На рис. 2.5 показана вероятность встраивания по критерию Согласия при анализе стего, сформированной с использованием JSteg. Видно, что статистическая атака успешно обнаруживает следы скрываемой информации в первой части исследуемой последовательности, содержащей скрываемое сообщение, и не дает ложной тревоги во второй ее части, являющейся пустым контейнером.

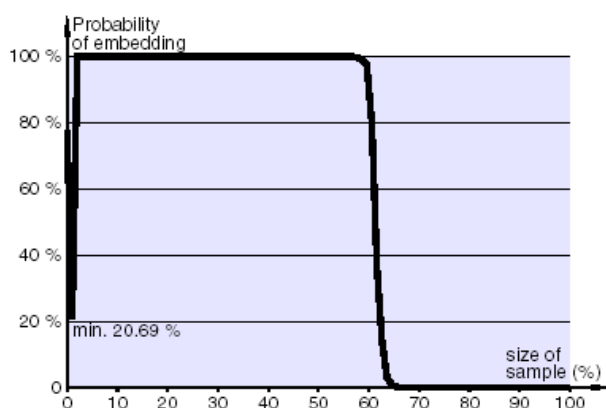


Рис. 2.5 Вероятность встраивания по критерию согласия при анализе JSteg

Для сжатия изображений очень часто используется алгоритм JPEG. На рис. 2.6 показано, что вероятность ложного срабатывания по критерию Согласия при анализе пустых контейнеров, сжатых алгоритмом JPEG, не превышает пренебрежимо малой величины 0,407%.

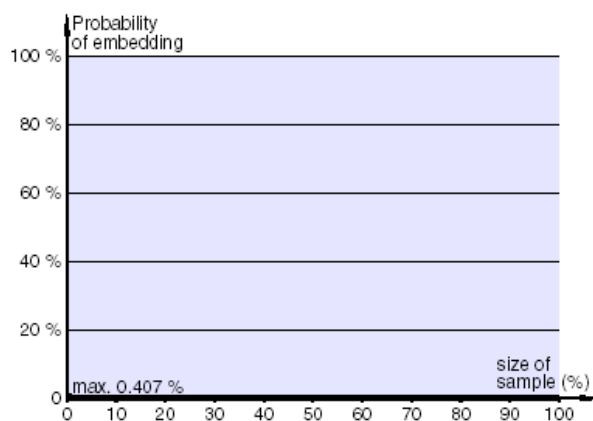


Рис. 2.6 Вероятность ложного срабатывания по критерию Согласия при сжатии по JPEG пустого контейнера

2.3.4. Цель атак на стегосистему

Главной целью любой атаки на стегосистему является, прежде всего, обнаружение стеганографического канала. Максимально достижимым результатом при реализации атаки – получение полной информации о стеганографической системе.

Атака с известным контейнером это самая слабая из всех возможных атак. Имея в распоряжении исходный контейнер, после его сравнения со стегоконтейнером, можно сделать вывод о существовании стегоканала. Подобную атаку может произвести пассивный противник.

Атака с выбором контейнера.

В этом случае противник сам выбирает тип контейнера (наиболее удобный с точки зрения последующего анализа) и создает условия, при которых передающий может воспользоваться для передачи скрытого сообщения только этим контейнером.

Атака с известным стего – это более сильная атака, поскольку располагая стего, но не зная исходного контейнера, противник не может однозначно установить факт существования стегоканала. Если задача в этом случае сводится к анализу стегоконтейнера и определению, является ли он стего или пустым. Подобную атаку может осуществить активный или злонамеренный противник.

Атака с выбором стего. Такая атака предполагает, что противнику известно некоторое множество стего и, возможно, реакция на некоторые из них. Тогда задача противника состоит в “навязывании” ему при определённых условиях какого-то конкретного стего и анализе полученного ответа. Подобная атака может быть осуществлена только при наличии канала с повторением. Её может реализовать активный или злоумышленный противник.

Атака с известным скрытым сообщением. Возможны, по крайней мере два варианта рассматриваемой атаки.

- известно стего, соответствующее скрытому сообщению. В этом случае задача состоит в определении секретного ключа.

- соответствующий скрытому сообщению стего неизвестен. В этом случае задача является нетривиальной и может не иметь решения.

Атака с выбором скрытого сообщения – это самый сильный вид атаки, которую может произвести только злонамеренный противник. Задача сводится к следующему: противник “подбрасывает” известное ему скрытое сообщение, получив стего с этим сообщением, анализирует его с целью установить секретный ключ. Кроме того, существует ряд атак, направленных на разрушение встроенного сообщения.

Атаки, направленные на удаление встроенного сообщения. Зашумление контейнера. Внесение в заполненный контейнер дополнительного шума неизбежно влечёт за собой потерю сообщения. С другой стороны, операция фильтрации также приводит к уничтожению встроенного сообщения. В этом случае используют либо низкочастотный и высокочастотный фильтры по отдельности, либо оба фильтра совместно.

Изменение гистограммы – растяжение или выработка гистограммы, которые иногда используются для компенсации недостаточного освещения.

2.4. Стеганограф JSTEG

2.4.1. Алгоритм работы

Программа JSTEG является одним из стеганографов, встраивающих информацию в файлы графического формата JPEG. Как и все подобные стеганографы, JSTEG осуществляет модификацию последних незначащих бит коэффициентов дискретно - косинусного преобразования битами скрываемой информации.

2.4.2. Стегоанализ стеганографа JSTEG

Первым публично доступным стеганографом для JPEG файлов был стеганограф JSteg. Его алгоритм встраивания информации заключался в последовательной замене младших бит ДКП коэффициентов битами скрываемой информации. Этот алгоритм не предполагает особой скрытности; поэтому любой, кто знаком с системой стеганографии, может вскрыть сообщение, спрятанное программой JSteg.

Отмечается, что стеганографические системы, которые изменяют младшие биты последовательно, вызывают искажения, обнаруживаемые с помощью стегоанализа. Они заметили, что для данного изображения встраивание данных, имеющих высокую энтропию (обычно вызванную крипто кодированием) приводит к изменению гистограммы частот цветов определённым образом.

Итак, процедура встраивания изменяет младшие биты цветов изображения. Цвета адресуются с помощью индексов таблицы цветов.

То же самое относится и к формату JPEG. Но вместо измерения частот цветов, следует обратить внимание на изменения частоты ДКП коэффициентов. На рис 2.7 показаны гистограммы до и после встраивания информации в JPEG файл.

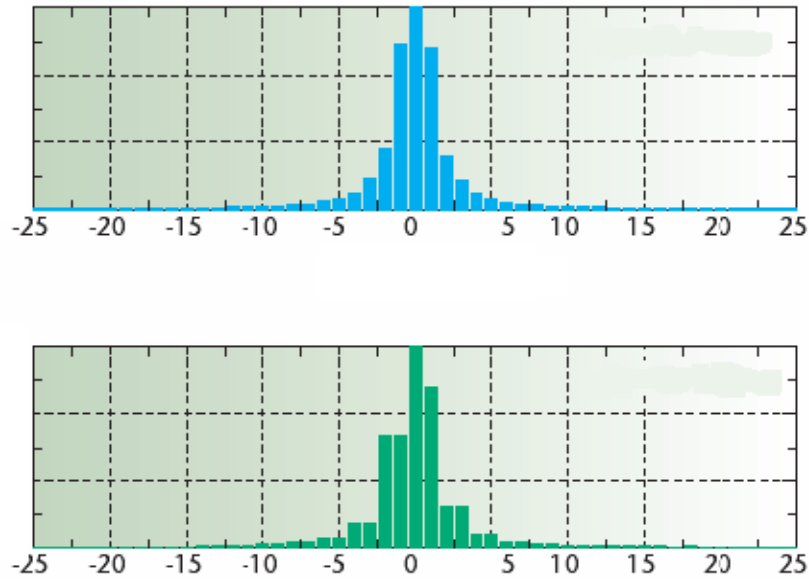


Рис. 2.7. Гистограммы до и после встраивания информации в JPEG файл

По сравнению с исходным пустым контейнером, для стего можно заметить снижение частоты изменения между двумя соседними ДКП коэффициентами – первым и вторым. Также можно заметить аналогичное снижение частоты между вторым и третьим коэффициентами.

Применим критерия согласия для определения, соответствует ли рассматриваемое частотное распределение y_i распределению y_i^* - искажённого контейнера. Хотя мы не знаем первоначального изображения, нам известно, что сумма соседних ДКП коэффициентов остаётся неизменной, что позволяет нам подсчитать ожидаемое распределение y_i^* из стего контейнера. Если n_i это гистограмма ДКП коэффициентов, то можно подсчитать среднее арифметическое

$$y_i^* = \frac{n_{2i} + n_{2i+1}}{2};$$

для определения ожидаемого распределения и сравнить его с рассматриваемым распределением

$$y_i = n_{2i};$$

Значение критерия согласия для разницы между распределениями определяется как

$$\chi^2 = \sum_{i=1}^{v+1} \frac{(y_i - y_i^*)}{y_i^*};$$

где v – количество степеней свободы, которое на единицу меньше, чем количество различных категорий гистограммы. Необходимо сложить соседние значения ожидаемого распределения и рассматриваемого распределения, чтобы убедиться, что категории количественно равны. Сложение двух соседних категорий уменьшает степень свободы на единицу.

$$p = 1 - \int_0^{\chi^2} \frac{t^{(v-2)/2} e^{-t/2}}{2^{v/2} \Gamma(v/2)};$$

$\Gamma(\dots)$ – гамма функция Эйлера.

Вероятность закладки определяется вычислением p для фрагмента ДКП коэффициентов. Фрагменты начинаются с начала изображения; для каждого измерения размер фрагмента увеличивается.

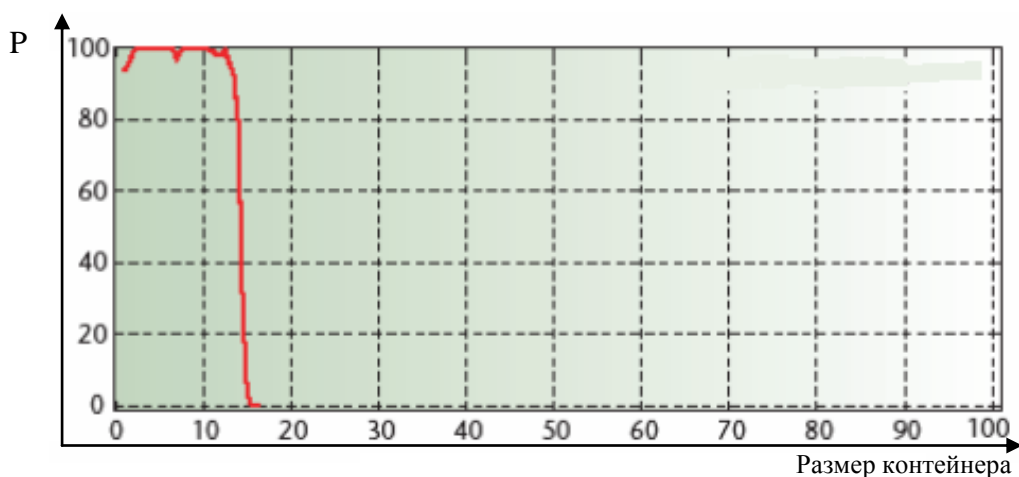


Рис. 2.8. Вероятность встраивания закладки в стего изображении

На рисунке 2.8 показана вероятность встраивания закладки в стего изображении, созданном программой JSteg. Высокая вероятность в начале изображения показывает наличие скрытого сообщения. Момент, с которого вероятность падает, указывает на завершение сообщения.

3. Порядок выполнения работы

Разработка комплекса программ стегоанализа графических файлов формата JPEG

3.1. Обоснование требований к разрабатываемому комплексу программ

В соответствии со спецификой решаемых задач, комплекс программ стегоанализа графических файлов формата JPEG должен соответствовать следующим требованиям:

- загружать для совместного анализа анализируемое и вспомогательное изображения;
- проводить различные статистические анализы коэффициентов дискретно-косинусного преобразования (ДКП) и отображать (в том числе и графически) соответствующие результаты, такие как:
 - количество искаженных значений ДКП;
 - распределение стегозакладки по всему объёму контейнера в линейном представлении;
 - распределение стегозакладки по всему объёму контейнера в двумерном представлении;
 - поиск всех уникальных значений ДКП – коэффициентов, их частоты и относительной вероятности;
 - поиск всплесков (тест Иванова);
 - построение гистограмм распределения ДКП – коэффициентов;
 - статистический тест на определение позиций стего в контейнере;
- представлять собой некую оболочку, выполняющую следующие функции:

- подключение отдельных модулей в виде DLL библиотек, реализующих различные методы стегоанализа;
- сканирование файловой системы на наличие файлов формата JPEG и анализ каждого файла с применением имеющихся методов;
- вывод результата стегоанализа для каждого файла в виде вероятности наличия закладки в нём.

3.2 Теоретические основы стегоанализа файлов графического формата JPEG

Основной задачей стегоанализа является определение факта наличия скрытого сообщения в предположительном контейнере (речи, видео, изображении). Решить эту задачу возможно путем изучения статистических свойств сигнала. Например, распределение младших битов сигналов имеет, как правило, шумовой характер (ошибки квантования). Они несут наименьшее количество информации о сигнале и могут использоваться для внедрения скрытого сообщения. При этом, возможно, изменится их статистика, что и послужит для атакующего признаком наличия скрытого канала.

Для незаметного встраивания данных стегокодер должен решить три задачи: выделить подмножество бит, модификация которых мало влияет на качество (незначимые биты), выбрать из этого подмножества нужное количество бит в соответствии с размером скрытого сообщения и выполнить их изменение. Если статистические свойства контейнера не изменились, то внедрение информации можно считать успешным. Так как распределение незначимых бит зачастую близко к белому шуму, встраиваемые данные должны иметь тот же характер. Это достигается за счет предварительного шифрования сообщения либо его сжатия.

Стегоаналитик на основе изучения сигнала всегда может выделить подмножество незначимых бит. Далее он должен проверить соответствие статистики анализируемого сигнала предполагаемому. При этом если аналитик

располагает лучшей моделью данных, чем стеганограф, вложение будет обнаружено. Поэтому, по-настоящему хорошие модели сигналов различного характера, вероятно, держатся в секрете, и вы не встретите их в открытых публикациях. Можно лишь дать рекомендации общего характера. При построении модели надо учитывать:

- неоднородность последовательностей отсчетов;
- зависимость между битами в отсчетах (корреляцию);
- зависимость между отсчетами;
- неравновероятность условных распределений в последовательности отсчетов;
- статистику длин серий (последовательностей из одинаковых бит).

Соответствие реально наблюдаемой статистики сигнала ожидаемой обычно проверяется при помощи критерия согласия. Возможны и более сложные тесты, аналогичные применяющимся при тестировании криптографически безопасных программных датчиков случайных чисел. Как показано в одной из работ на примере звуковых файлов, критерий согласия позволяет обнаружить модификацию всего лишь 10% незначащих битов. Там же показана эффективность для стегоанализа и еще более простого критерия

$$\theta = \frac{m_{00} - m_{01}}{2} - \frac{m_{11} - m_{10}}{2}$$
, где m_{ij} - количество переходов из значения бита i в значение j . Применение теста длин серий основано на следующем факте: в случайной последовательности серии большой длины (>15) встречаются значительно реже, чем в незначащих битах реальных сигналов. Поэтому, встраивание случайного сигнала может быть замечено после применения этого теста.

Таким образом, противодействие статистическому стегоанализу должна заключаться в построении математических моделей сигналов-контейнеров, поиску на их основе «разрешенных» для модификации областей и внедрению в них скрытой информации, чья статистика неотличима от статистики контейнера.

В большинстве методов скрытия данных в изображениях используется та или иная декомпозиция изображения - контейнера. Среди всех линейных

ортогональных преобразований наибольшую популярность в стеганографии получили вейвлет-преобразование и ДКП, что отчасти объясняется их успешным применением при сжатии изображений. Кроме того, желательно применять для скрытия данных то же преобразование изображения, как и то, которому оно подвергнется при возможном дальнейшем сжатии. В стандарте JPEG используется ДКП, а в JPEG2000 – вейвлет-преобразование.

3.3 Исследование возможностей обнаружения закладок, формируемых стеганографом JSteg

Первым публично доступным стеганографом для JPEG файлов был стеганограф JSteg. Его алгоритм встраивания информации заключался в последовательной замене младших бит ДКП коэффициентов битами скрываемой информации. Этот алгоритм не предполагает особой скрытности; поэтому любой, кто знаком с системой стеганографии, может вскрыть сообщение, спрятанное программой JSteg.

Отмечают, что стеганографические системы, которые изменяют младшие биты последовательно, вызывают искажения, обнаруживаемые с помощью стегоанализа. Замечено, что для данного изображения встраивание данных, имеющих высокую энтропию (обычно вызванную крипто кодированием) приводит к изменению гистограммы частот цветов определённым образом.

Процедура встраивания изменяет младшие биты цветов изображения. Цвета адресуются с помощью индексов таблицы цветов.

То же самое относится и к формату JPEG. Но вместо измерения частот цветов, следует обратить внимание на изменения частоты ДКП коэффициентов.

3.4 Описание программного комплекса “Стегоанализатор – 2006”

Программный комплекс “Стегоанализатор - 2006” предназначен для проведения различных статистических тестов и сравнения их результатов

относительно пустого и заполненного контейнера. В разработке данного комплекса применялся уже ранее написанный, который раскладывает данный файл графического формата JPEG и распаковывает его на составляющие: заголовки, таблицы квантования, коэффициенты дискретного косинусного преобразования и др. Это программа усовершенствована добавлением в неё функции сохранения множества коэффициентов ДКП в отдельных файлах для последующей работы с ними.

Итак, как мы уже выяснили, все стеганографы, работающие с JPEG форматом закладывают информацию в коэффициенты ДКП. Различие состоит лишь в том, что каждый из них делает это по своему собственному алгоритму. Основная цель этого алгоритма – свести к минимуму отличие стего от стандартного распределения. Ввиду того, что алгоритм JPEG является алгоритмом с потерями, то внедрение происходит непосредственно после всех процедур округления и удаления избыточных данных.

Итак, программный комплекс «Стегоанализатор – 2006» работает с множеством коэффициентов ДКП, пустого контейнера и этого же контейнера, но с закладкой. Рассмотрим работу комплекса на примере известного и распространённого стеганографа JSteg.

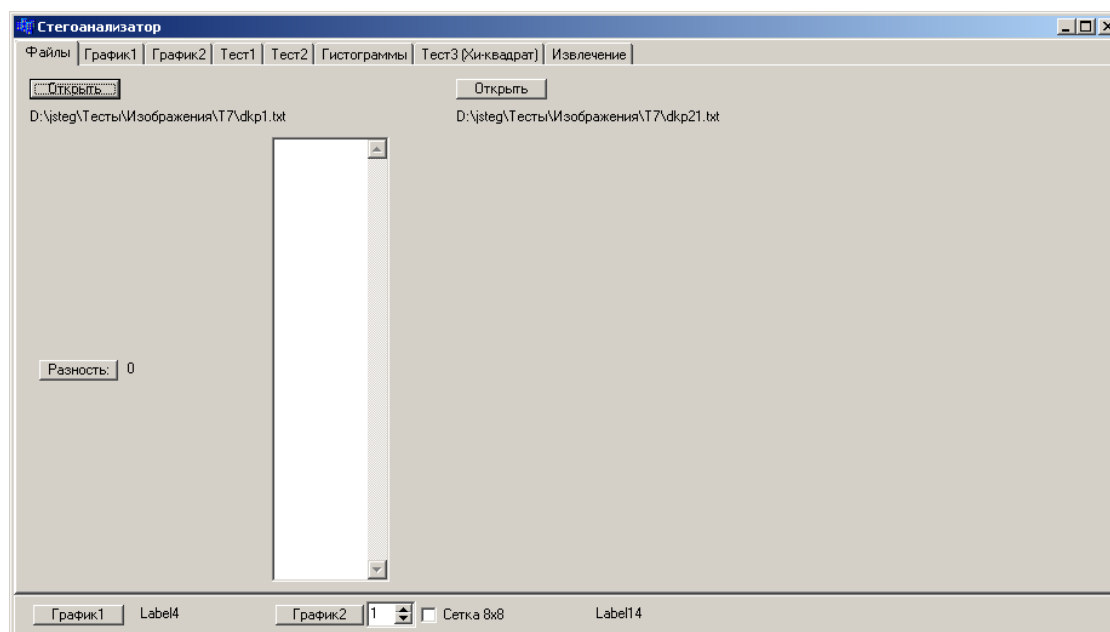


Рис 3.1. Заглавное окно программного комплекса «Стегоанализатор - 2006»

Нажатием кнопки “Разность” вызывается функция, которая поэлементно сравнивает оба множества коэффициентов a , b и для каждой пары выводит их разность $a - b$.

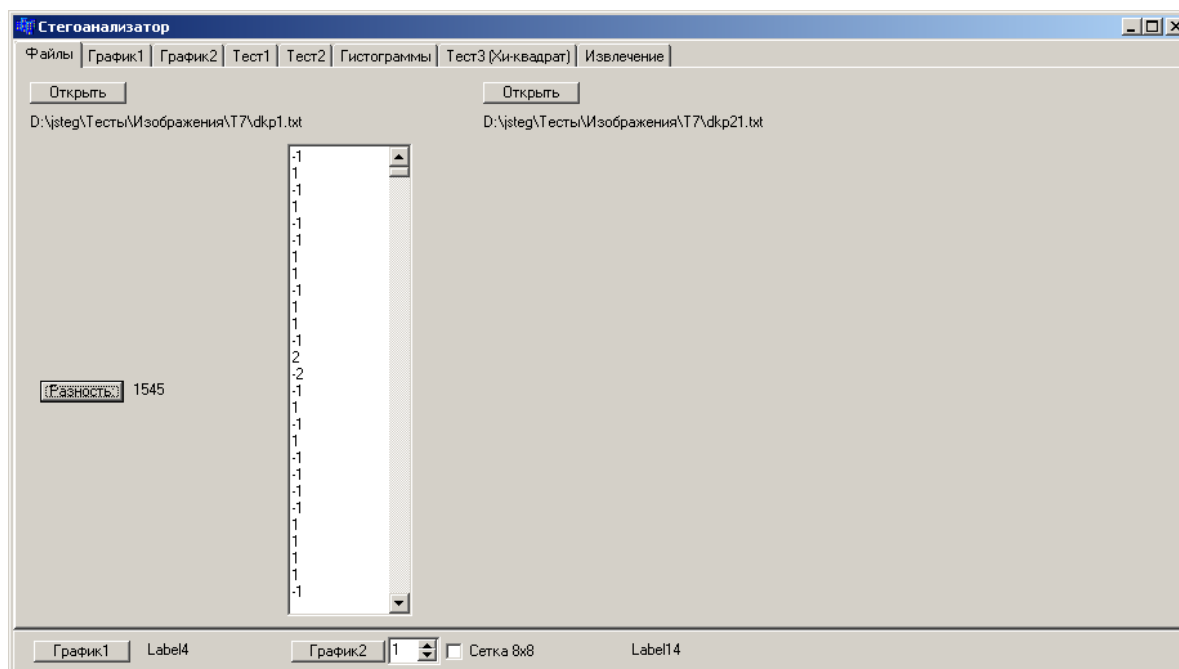


Рис 3.2. Заглавное окно программного комплекса «Стегоанализатор - 2006», расчет разности.

Из результатов видно, что количество различных значений равно 1545, а максимальная абсолютная разность соответствующих коэффициентов ДКП равна 2. Однако таких пар всего 2. Подавляющее же большинство разностей равны 1 или -1, что свидетельствует о том, что в процессе скрытия используются младшие незначащие биты коэффициентов ДКП.

На следующем рисунке 3.3:

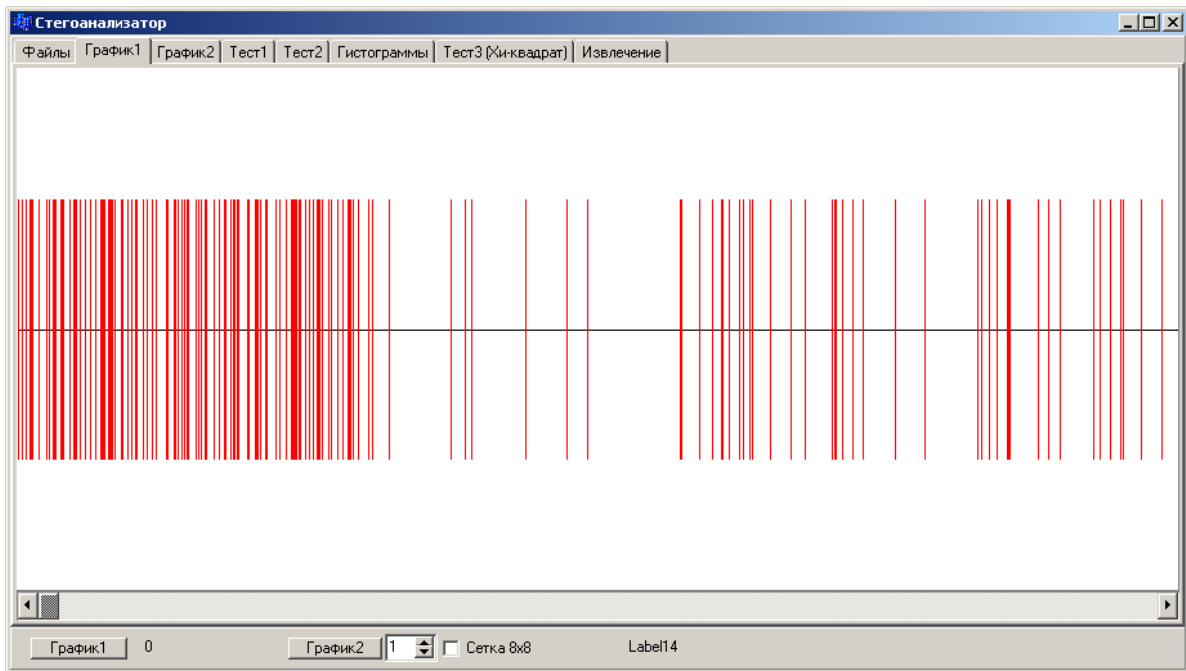


Рис. 3.3 Картина линейного распределения закладки по объёму контейнера показано линейное одномерное распределение закладки по объёму всего файла. На рисунке по горизонтали слева направо каждый пиксель соответствует очередному коэффициенту ДКП. Красными вертикальными отрезками отмечены позиции, в которых соответствующие коэффициенты не равны. Данный тест направлен на анализ алгоритм закладки с точки зрения позиций встраивания информации и может привести к следующим выводам:

1. закладка встраивается последовательно сначала файла;
2. закладка равномерно распределяется по всему объёму файла;
3. закладка дополняется (если это необходимо) нужным количеством бесполезной информации, а затем распределяется по всему контейнеру.

Следующий тест, представленный на рисунке ниже

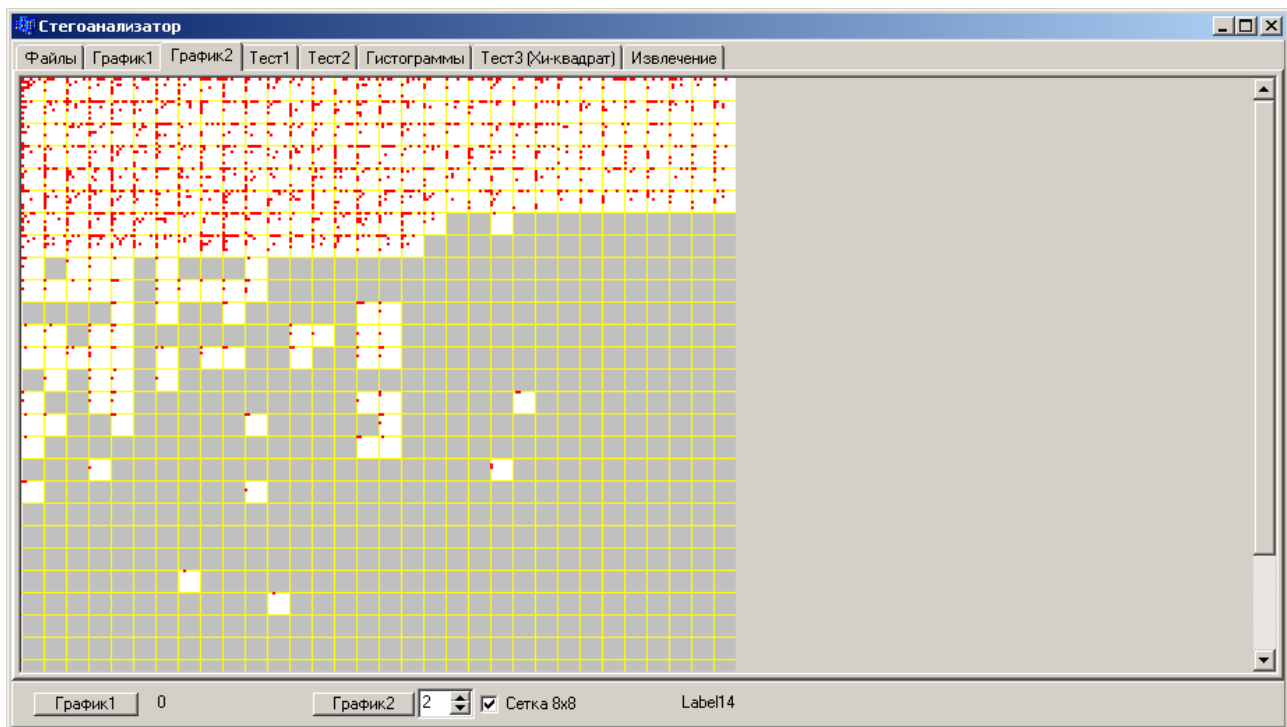


Рис. 3.4. Двумерная картина распределения закладки по объёму контейнера является как бы расширением предыдущего теста в том смысле, что по нему можно также сделать выводы о характере распределения закладки по объёму всего файла, а также прийти ещё к нескольким интересным заключениям. На рисунке изображён образ анализируемого изображения. Жёлтыми линиями вся область поделена на блоки 8x8 точек (каждая точка соответствует конкретному ДКП коэффициенту). Красными точками обозначены места, в которых соответствующие коэффициенты различны. Серым цветом закрашены те блоки, в которых все коэффициенты соответственно равны. Итак, на данном рисунке уже отчётливо виден характер распределения закладки по объёму всего файла. И можно сделать вывод, что стеганограф, с помощью которого был создан анализируемый файл, осуществляет закладку последовательно от начала файла, пока не закончится вся информация. Если брать и анализировать конкретные значения коэффициентов ДКП, сравнивая их с результатом данного теста, то можно прийти к очень важному выводу: данный стеганограф искажает только те коэффициенты, значение которых не равно 0 и 1. И действительно, обратите внимание множество

кранных квадратиков внутри каждого блока 8x8 сосредоточено ближе к левому верхнему углу, поскольку по ходу движения к правому нижнему углу количество нулей резко возрастает – это главная особенность сжатия по алгоритму JPEG, используемая на шаге кодирования последовательности нулей. Следующая процедура носит скорее информативный характер:

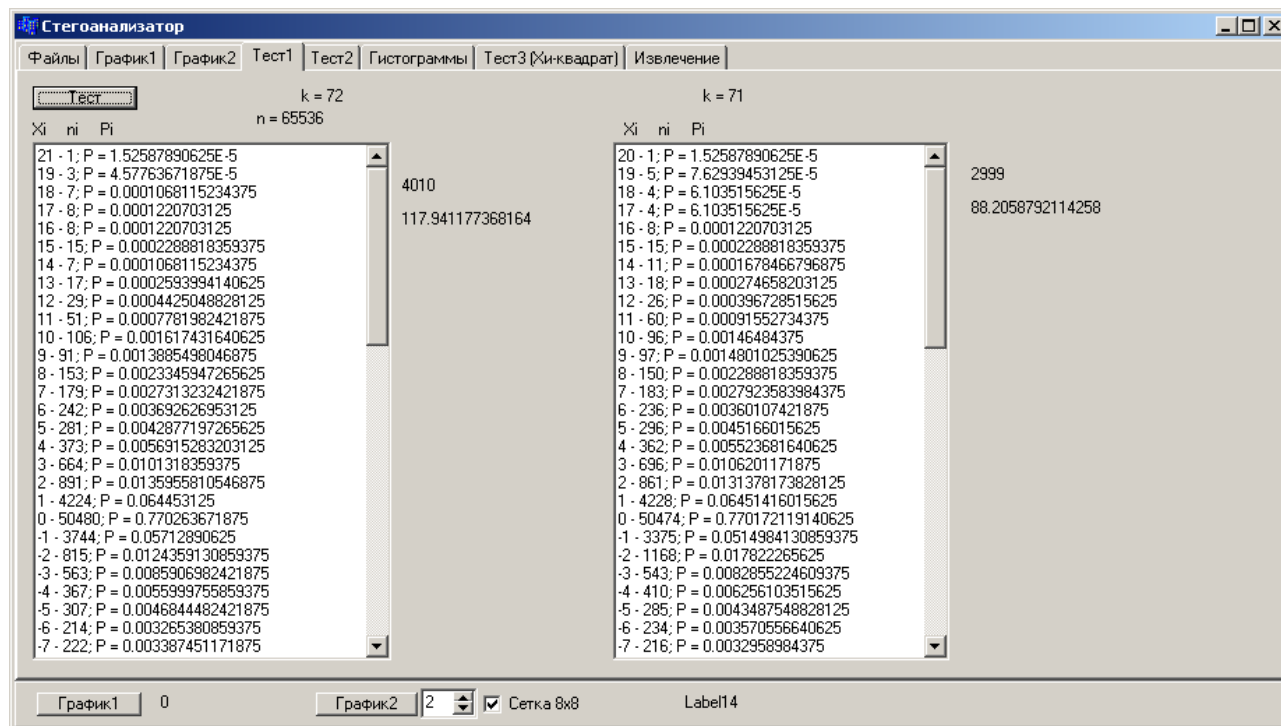


Рис. 3.5. Статистическая картина множества коэффициентов ДКП

В левом поле находится информация по файлу без закладки, а в правом – по файлу с закладкой. В каждом поле выведенные числа делятся на три столбца. В первом столбце содержится множество всех уникальных коэффициентов ДКП. Во втором столбце – их количество во всём множестве коэффициентов ДКП. Количество нулей (0 - 50480;), как исследовало ожидать, существенно преобладает над количествами других элементов. И чем больше абсолютное значение очередного коэффициента, тем меньше его количество во всём множестве. В третьем столбце указана вероятность, с которой очередной коэффициент встречается во всём множестве.

Следующий тест – тест всплесков:

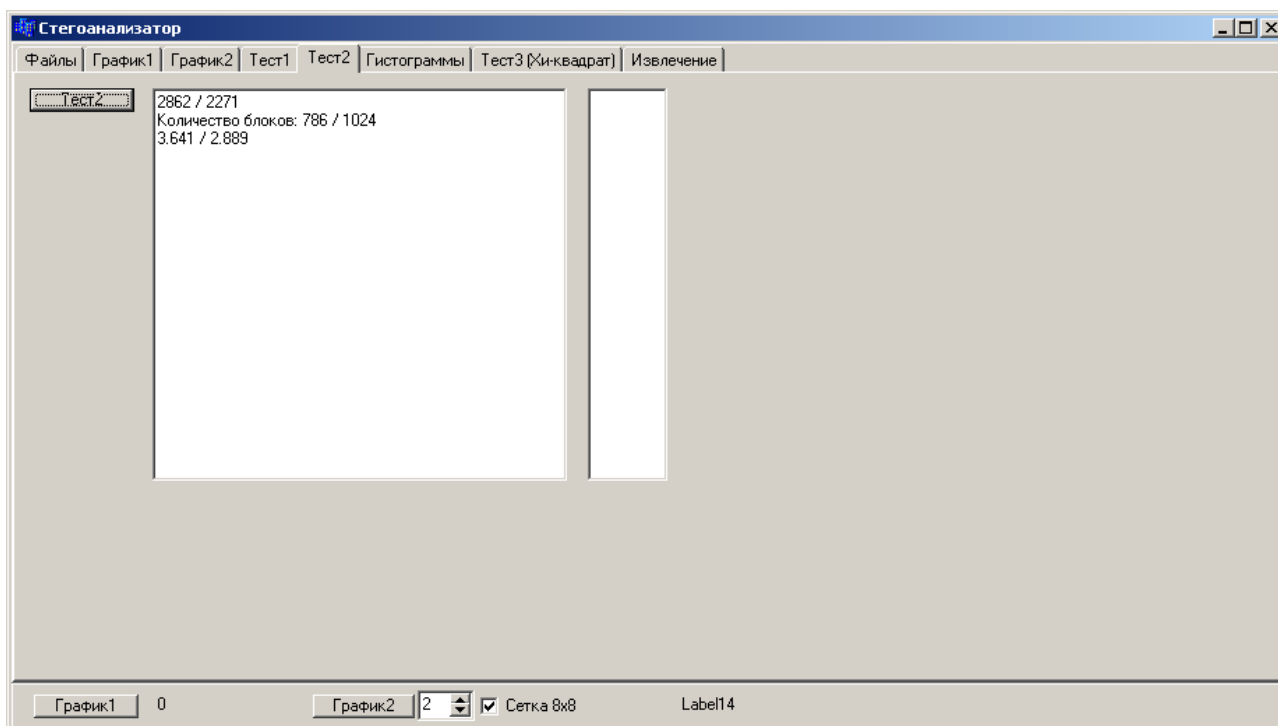


Рис. 3.6. Тест всплесков

он основан на статистических особенностях распределения нулевых и единичных коэффициентов ДКП пустого заполненного контейнера. Данный тест работает следующим образом: изображения сканируются по блокам 8x8. Каждый блок преобразуется в одномерный массив по алгоритму зигзага. И, наконец, среди элементов получившегося массива подсчитывается количество так называемых всплесков. Всплесками будем называть подряд идущие элементы: 0,1,0; 0,-1,0; 0,2,0; 0,-2,0; -1,1.

Результатом теста является общее количество всплесков относительно количества блоков 8x8, содержащих хотя бы один такой всплеск.

Проанализировав некоторое множество различных пар контейнер – стего, варьируя исходным изображением либо закладываемой информацией, можно выявить некоторый порог отличия пустых контейнеров от стего. В последствии на основе этого порога можно проводить стегоанализ любых файлов.

На следующем рисунке изображена гистограмма коэффициентов ДКП:

Количество коэффициентов

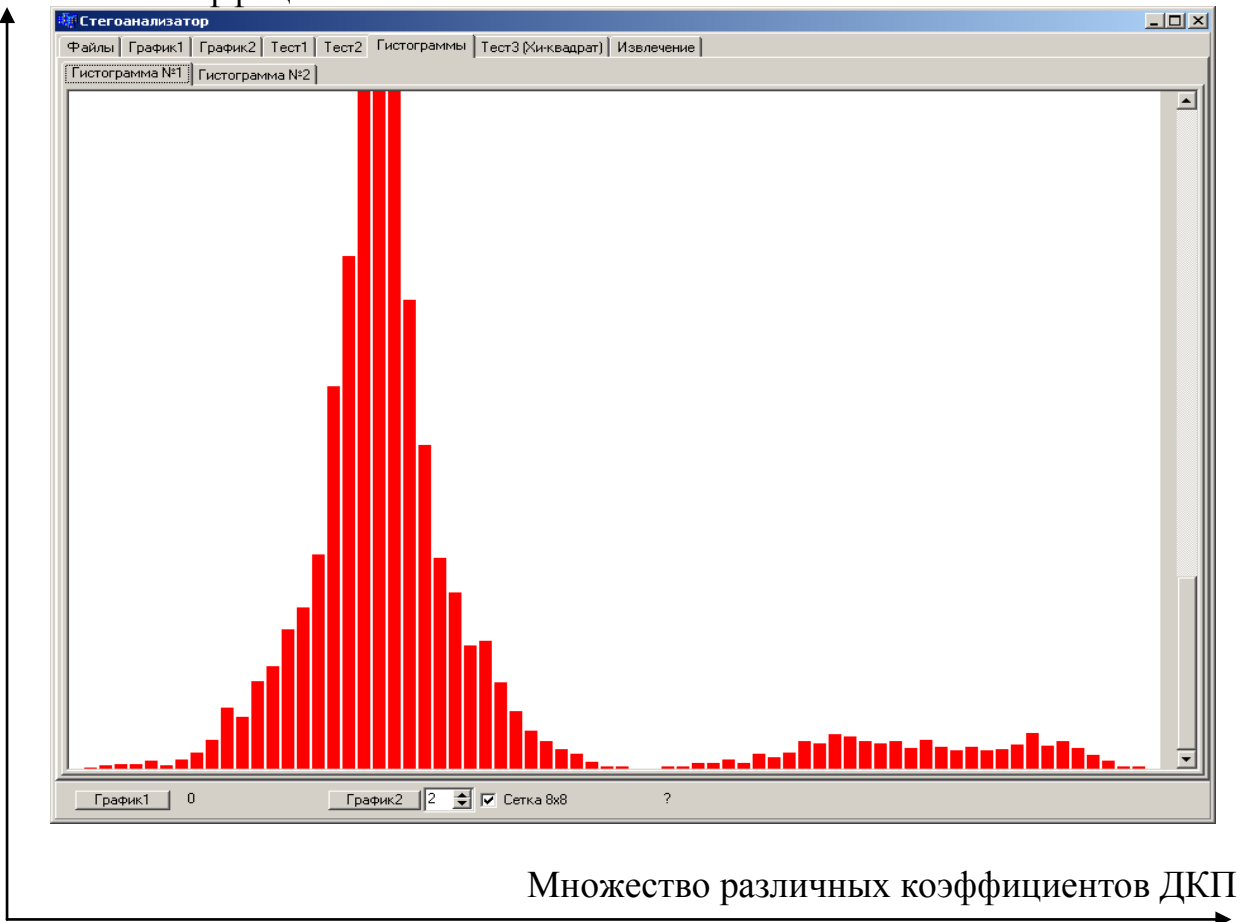


Рис. 3.7. Гистограмма распределения коэффициентов ДКП

Столбцы соответствуют различным коэффициентам ДКП. Высота столбика соответствует количеству этих коэффициентов во всём множестве. Самый высокий столбик соответствует коэффициенту 0.

На следующем рисунке реализован тест согласия.

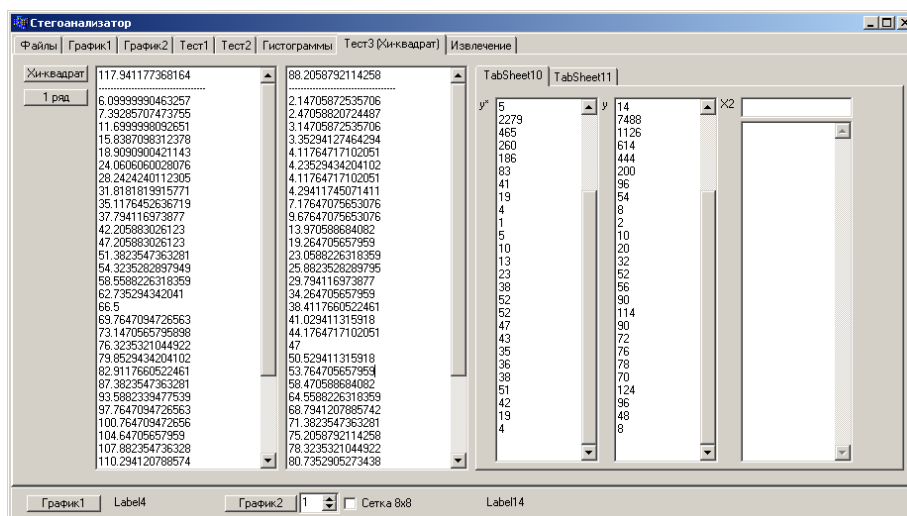


Рис. 3.8. Тест согласия

Как и в предыдущих тестах, первый столбик соответствует контейнеру без закладки, а второй – с закладкой.

На основе двух основных известных нам фактов, а именно:

- рассматриваемый стеганограф осуществляет закладку с начала контейнера подряд во все коэффициенты (кроме 0 и 1);
- после закладки наблюдается определённое изменение статистической картины между пустым и заполненным файлом.

построим алгоритм тестирования следующим образом. Будем сканировать изображение по сегментам. Сначала первую строчку, затем первые две, затем первые три и так далее. И, в конце концов, всё изображение. На каждом шагу будем считать среднюю разность соответствующих коэффициентов ДКП.

На графиках видны распределения получившихся значений:



а)



Рис. 3.9. Распределение заполненного и пустого контейнера

На рисунке 3.9а, сверху график возрастает равномерно, практически под одним и тем же углом, что соответствует равномерному распределению пустого контейнера. А на рисунке 3.9б, снизу как будто, тот же график, но с изменённой первой четвертью. Именно в этой первой четверти контейнера и была встроена закладка. Видно, что график очень пологий, что соответствует низкой разнице между соседними коэффициентами ДКП.

Это есть явный и главный демаскирующий признак рассматриваемого стеганографа, и может использоваться в качестве основного критерия при стегоанализе файлов.

Итак, задача стегоанализатора сведётся к последовательному сканированию контейнера и поиску этой точки перегиба. Если этой точки нет, то считается, что контейнер пуст, в противном случае в контейнере присутствует стегозакладка. И, чем больше абсолютная разница углов до и после переломной точки, тем больше вероятность наличия закладки в данном контейнере.

3.5 Описание программного комплекса “Стегоанализатор – JPEG Anti JSTEG”

Итак, из предыдущего раздела выяснилось, что существует множество различных тестов и проверок, которые с разной степенью эффективности обнаруживает тот или иной стеганограф. Первой задачей стегоаналитика является сопоставление стеганограф – множество методов и определение их эффективности. А конечным результатом работы должен быть комплекс, реализующий конкретные методы стегоанализа.

Как говорилось в предыдущем пункте, явным и основным демаскирующим признаком стеганографа JSteg является наличие некой переломной точки – границей между закладкой и пустым контейнером. Программный комплекс “Стегоанализатор JPEG Anti JSTEG” реализует данный метод. На рисунке ниже показано заглавное окно программы. Рабочая область разделена на две области: слева – навигатор файловой системы, в которой пользователь выбирает рабочий каталог, справа – список всех файлов формата JPEG, которые находятся в выбранном каталоге. Нажатием кнопки “Пуск” активизируется процесс сканирования всех файлов с помощью метода переломной точки.

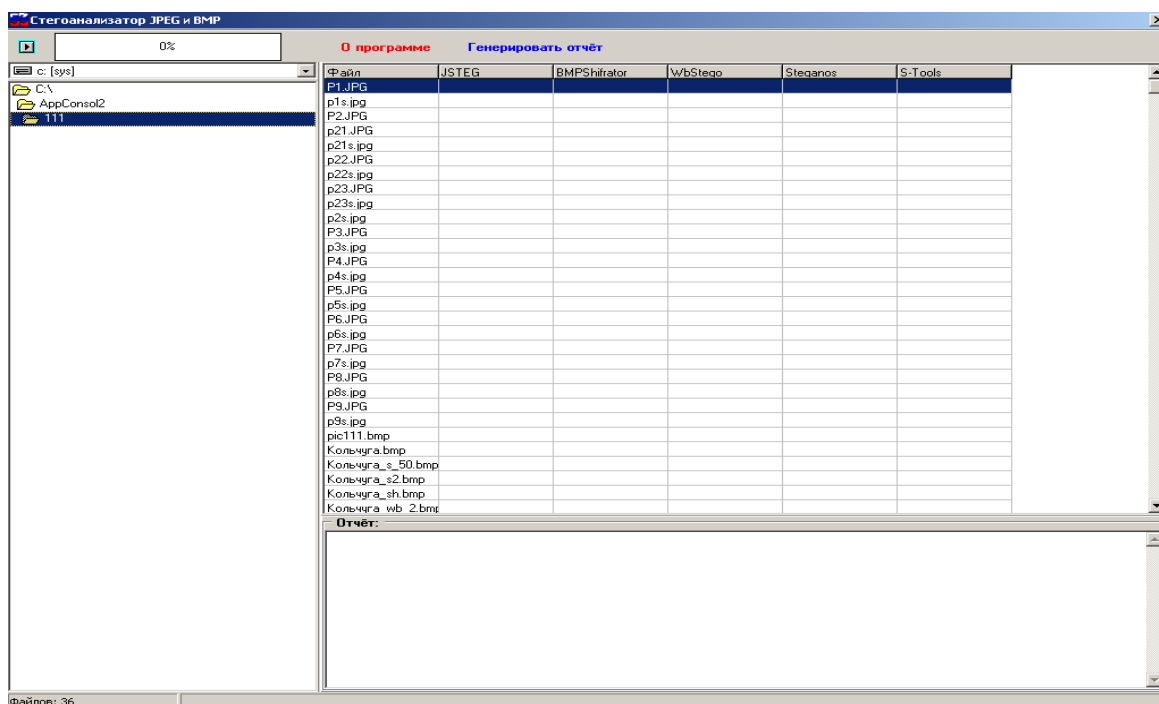


Рис. 3.10 Интерфейс программы Стегоанализатор JPEG

Правая часть окна представлена в виде таблицы с двумя столбцами. В первом столбце указано имя файла, а во втором – результат тестирования этого файла. Если во втором столбце после сканирования осталось пустое значение, то данный файл не содержит закладки, а если в нём появилось какое то число а, то значит, что данный файл с вероятностью а содержит закладку, созданную стеганографом JSteg.

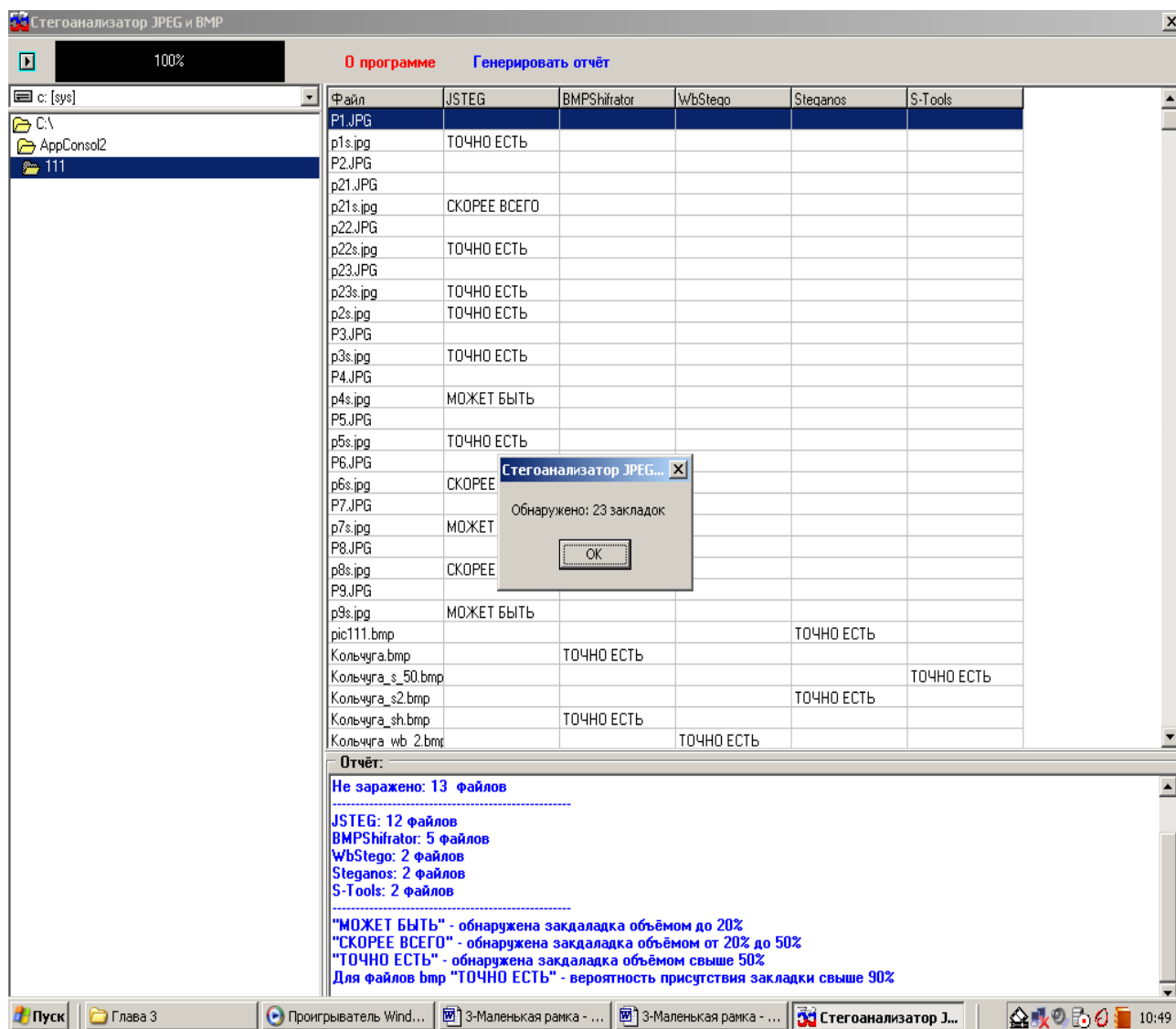


Рис. 3.11. Процесс работы программы Стегоанализатор JPEG

Для удобства пользователя вверху окна отображается окно процесса, которое заполняется по ходу сканирования файлов.

4.2 Описание работы программного комплекса

4.2.1 Состав и основные характеристики специального программного обеспечения стегоанализа графических файлов JPEG

Основным назначением разрабатываемого программного обеспечения стегоанализа графических файлов JPEG является:

«Стегоанализатор - 2006»:

- загрузка файлов с коэффициентами дискретного косинусного преобразования пустого контейнера и аналогичного заполненного для их сравнительного статистического анализа;
- реализация различных методов тестирования (анализа) входных данных;
- реализация комплекса в виде расширяемой оболочки с возможностью дополнения других методов стегоанализа.

«Стегоанализатор JPEG Anti JSteg»

- выбор рабочего каталога для анализа;
- анализ всех JPEG файлов, содержащихся в рабочем каталоге;
- отображение результат анализа в виде таблицы **файл-вероятность**, где **файл** – это имя очередного файла рабочего каталога, **вероятность** – вероятность наличия в нём закладки;
- возможность дополнения комплекса другими методами стегоанализа.

4.2.2. Макет программного средства стегоанализа графических файлов формата JPEG

4.2.2.1. Состав и назначение программного комплекса

Программное средство стегоанализа графических файлов формата JPEG позволяет решать следующие задачи:

- реализация различных методов стегоанализа файлов JPEG с целью определения их эффективности против конкретного рассматриваемого стеганографа;

- стегоанализ файлов на предмет наличия закладки, встроенной стеганографом JSteg;

- добавление новых методов стегоанализа с целью повышения суммарной эффективности комплекса.

4.2.2.2 Порядок работы с программным комплексом стегоанализа графических файлов формата JPEG

«Стегоанализатор - 2006»

Программа представлена в виде одного исполняемого файла: **Стегоанализатор.exe**. Главное окно программы представлено на рисунке 4.1:

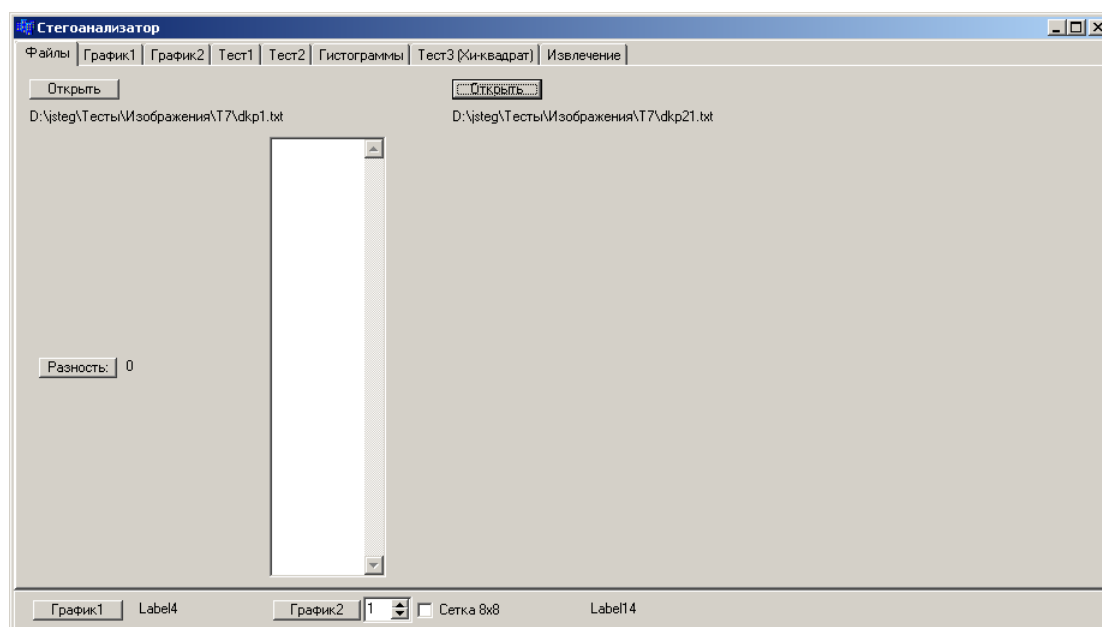


Рис. 4.1. Интерфейс программного комплекса «Стегоанализатор - 2006»

Для начала работы необходимо загрузить исходные данные - файлы коэффициентов дискретного косинусного преобразования. Слева – пустого контейнера, справа – контейнера с закладкой. После этого в программном комплексе становятся доступными все имеющиеся методы анализа.

Вверху окна на различных страничках расположены различные статистические тесты. Так же есть функции отображения гистограмм

распределения ДКП коэффициентов, одномерных и двумерных диаграмм распределения закладки по объёму файла.

При построение двумерной диаграммы распределения стего

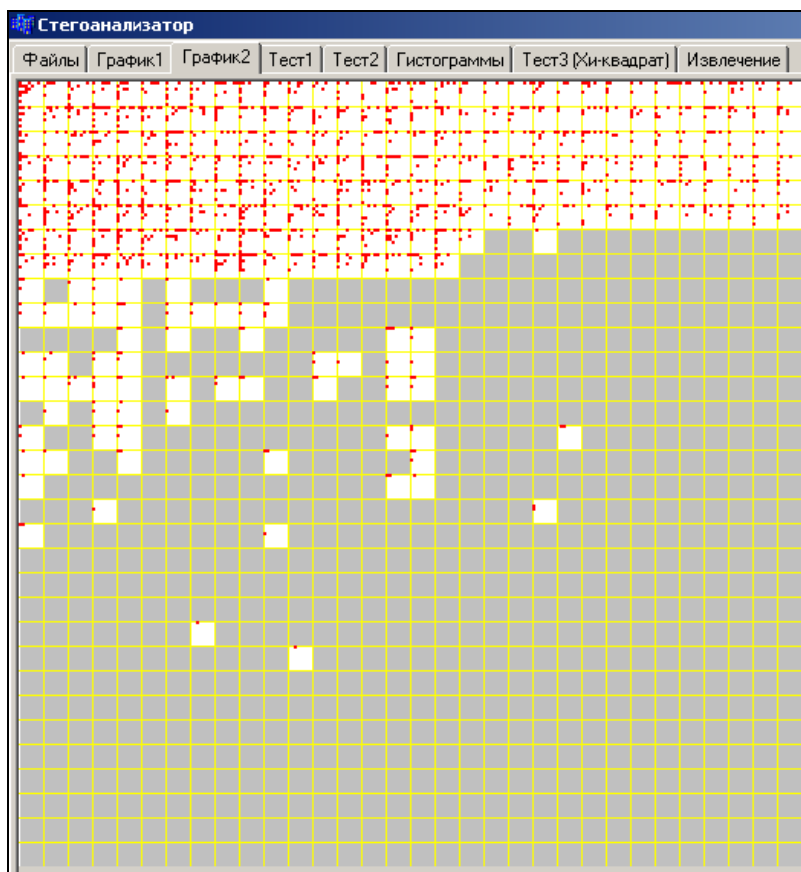


Рис. 4.2. Двумерная диаграмма распределения стего

жёлтыми линиями вся область разделена на квадраты 8x8 коэффициентов. Если квадрат покрашен серым цветом, значит, в нём не было ни одного искажения. В местах, где отмечено красным, соответствующие коэффициенты ДКП пустого и заполненного контейнеров не равны. Таким образом, по данной диаграмме можно визуальное сделать вывод о характере распределения закладки по всему объёму файла.

На графике гистограмм столбцы соответствуют всем различным коэффициентам ДКП. А высота каждого столбца соответствует количеству соответствующих коэффициентов во всём множестве.

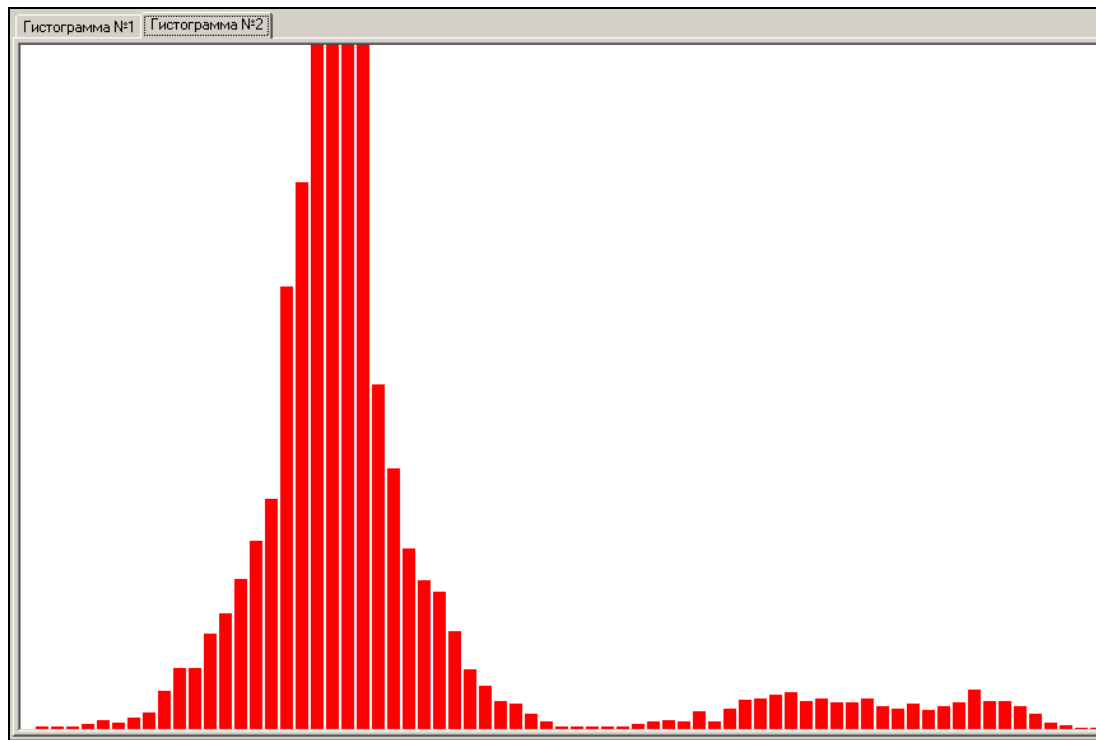


Рис. 4.2. Графики гистограмм

На следующем рисунке:

Гист1	График2	Тест1	Тест2	Гистограммы	Тест3 (Хи-квадрат)	Изм
117.941177368164					88.2058792114258	
6.09999990463257					2.14705872535706	
7.39285707473755					2.47058820724487	
11.6999998092651					3.14705872535706	
15.8387098312378					3.35294127464294	
18.9090900421143					4.11764717102051	
24.0606060028076					4.23529434204102	
28.2424240112305					4.11764717102051	
31.8181819915771					4.29411745071411	
35.1176452636719					7.17647075653076	
37.794116973877					9.67647075653076	
42.205883026123					13.970588684082	
47.205883026123					19.264705657959	
51.3823547363281					23.0588226318359	
54.3235282897949					25.8823528289795	
58.5588226318359					29.794116973877	
62.735294342041					34.264705657959	
66.5					38.4117660522461	
69.7647094726563					41.029411315918	
73.1470565795898					44.1764717102051	
76.3235321044922					47	
79.8529434204102					50.529411315918	
82.9117660522461					53.764705657959	
87.3823547363281					58.470588684082	
93.5882339477539					64.5588226318359	
97.7647094726563					68.7941207885742	
100.764709472656					71.3823547363281	

Рис. 4.3. Вывод значений средний разности между соседними значениями гистограммы

изображены результаты одного из статистических тестов. В i -ой строчке находится число, равное средней разнице между соседними значениями

гистограммы ДКП коэффициентов. Причём рассматриваются коэффициенты, принадлежащие сегменту изображения размером $(i*8) \times \text{Ширину}$.

Равномерное возрастание коэффициентов в левом столбце и переменное (сначала постоянное, потом возрастающее) изменение коэффициентов в правом столбце есть явный демаскирующий признак стегозакладки.

«Стегоанализатор JPEG Anti JSteg»

Программа представлена в виде одного исполняемого файла: **AntiJSteg.exe**.

Главное окно программы представлено на рисунке:

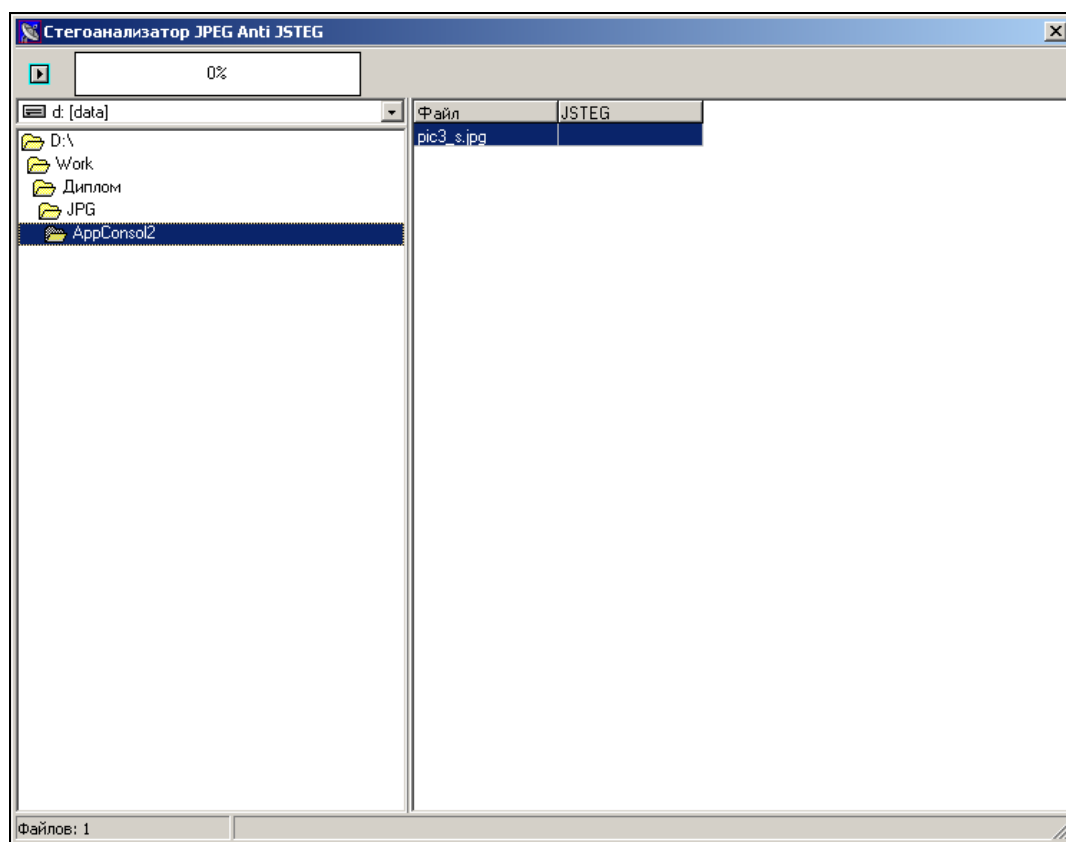


Рис. 4.4. Интерфейс программы «Стегоанализатор JPEG Anti JSteg»

Программа отличается удобным и интуитивно понятным даже для начинающих пользователей интерфейсом. Сначала требуется выбрать каталог, в котором пользователь желает проанализировать файлы JPEG на предмет наличия в них закладки. После этого можно начинать сканирование. Если в очередном файле обнаружена закладка, то справа от него в столбце “JSTEG” указывается вероятность этой закладки. В противном случае полагается, что закладки нет.

4. Рекомендуемая литература

1. Грибунин, В. Г. Цифровая стеганография / В. Г. Грибунин, И. Н. Оков, И. В. Туринцев. - М.: Солон-Пресс, 2002. – 272 с.
2. Ross J. Anderson, Fabien A.P. Petitcolas On The Limits of Steganography IEEE Journal of Selected Areas in Communications, 16(4):474-481, May 1998, Special Issue on Copyright & Privacy Protection. ISSN 0733-8716.
3. Рублев Д.П. Метод анализа стеганосообщений, основанный на корреляции точек изображения. -Сб.: "Известия ТРТУ" Таганрог, 2003, С. 307-310.
4. Кустов В.Н., Федчук А.А. Методы встраивания скрытых сообщений. //000 "Конфидент" журнал "Защита информации. Конфидент".-2000.-Я23.-С.34-37.

Лабораторная работа 3. Исследование отечественных алгоритмов шифрования информации ГОСТ 28147-89

1. Цель работы

Объектом исследования является введенный в 1990 году алгоритм криптографической защиты ГОСТ 28147-89.

Цель работы – создание виртуального блока, который бы шифровала и расшифровывала данные на основе алгоритма криптографической защиты ГОСТ 28147-89.

В процессе работы был разработан алгоритм шифрования и расшифровки файлов заданным методом. По разработанному алгоритму построены логические узлы, объединенные в блок шифрования.

Проектная работа выполнена при помощи пакета Simulink/Matlab.

Отчет по проектной работе выполнен в программе Microsoft Office Word 2007 и предоставлен в распечатанном виде.

2. Краткие теоретические сведения

Введение

Для защиты информации, представленной в электронном виде и хранимой на носителях информации, используются различные методы. От тривиального и устаревшего способа закрытия физического доступа (путем запираания компьютера на ночь в сейф) до сложных систем администрирования и разграничения доступа к различным данным, хранимым на носителях или где-то в сети (но тоже на носителях). Информацию частного порядка (конфиденциальную) или информацию, которая не должна стать доступной кругу лиц, большему, чем дозволено, при условии, что наличие этой информации скрыть нельзя, люди издавна старались преобразовать к виду, не возможному для чтения не посвященному. Среди всего спектра методов защиты

данных от нежелательного доступа особое место занимают криптографические методы. В отличие от других методов, они опираются лишь на свойства самой информации и не используют свойства ее материальных носителей, особенности узлов ее обработки, передачи и хранения. Образно говоря, криптографические методы строят барьер между защищаемой информацией и реальным или потенциальным злоумышленником из самой информации. Конечно, под криптографической защитой в первую очередь – так уж сложилось исторически – подразумевается шифрование данных. Раньше, когда эта операция выполнялось человеком вручную или с использованием различных приспособлений, и при посольствах содержались многолюдные отделы шифровальщиков, развитие криптографии сдерживалось проблемой реализации шифров, ведь придумать можно было все что угодно, но как это реализовать... Появление цифровых электронно-вычислительных машин, приведшее в конечном итоге к созданию мощной информационной индустрии, изменило все коренным образом и в этой сфере. С одной стороны, взломщики шифров получили в свои руки чрезвычайно мощное орудие, с другой стороны, барьер сложности реализации исчез, и для создателей шифров открылись практически безграничные перспективы.

1 Стандарт шифрования ГОСТ 28147-89

1.1 Описание алгоритма

ГОСТ 28147-89 — советский и российский стандарт симметричного шифрования, введённый в 1990 году, также является стандартом СНГ. Полное название — «ГОСТ 28147-89 Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования». С момента опубликования ГОСТа на нём стоял ограничительный гриф «Для служебного пользования», и формально шифр был объявлен «полностью открытым» только в мае 1994 года.

Алгоритм криптографического преобразования предназначен для аппаратной или программной реализации, удовлетворяет криптографическим

требованиям и по своим возможностям не накладывает ограничений на степень секретности защищаемой информации.

Стандарт обязателен для организаций, предприятий и учреждений, применяющих криптографическую защиту данных, хранимых и передаваемых в сетях, в отдельных вычислительных комплексах или на персональных компьютерах.

То, что в его названии вместо термина «шифрование» фигурирует более общее понятие «криптографическое преобразование», вовсе не случайно. Помимо нескольких тесно связанных между собой процедур шифрования, в документе описан один построенный на общих принципах с ними алгоритм выработки имитовставки.

Имитовставка – отрезок информации фиксированной длины, полученный по определенному правилу из открытых данных и ключа, и добавленный к зашифрованным данным для обеспечения имитозащиты (защиты системы шифрованной связи от навязывания ложных данных).

К симметричному шифрованию предъявляются следующие требования:

- Отсутствие линейности (то есть условия $f(a) \text{ xor }^1 f(b) = f(a \text{ xor } b)$), в противном случае облегчается применение дифференциального криптоанализа² к шифру.
- Полная утрата всех статистических закономерностей исходного сообщения. Для этого шифр должен иметь «эффект лавины». Лавинный эффект проявляется в зависимости всех выходных битов от каждого входного бита:
 1. Криптографический алгоритм удовлетворяет *лавинному критерию*, если при изменении одного бита входной последовательности, изменяется в среднем половина выходных битов.
 2. Криптографический алгоритм удовлетворяет *строгому лавинному критерию*, если при изменении одного бита входной последовательности,

каждый бит выходной последовательности изменяется с вероятностью одна вторая.

3. Криптографический алгоритм удовлетворяет *критерию независимости битов*, если при изменении любого входного бита, любые два выходных бита изменяются независимо.

Алгоритм ГОСТ 28147-89 является *блочным шифром* – разновидность симметричного шифра. Особенностью блочного шифра является обработка блока нескольких байт за одну итерацию (как правило, 8 или 16). Как и большинство современных блочных шифров, ГОСТ основан на *сети Фейстеля* (рисунок 1.1). Сеть представляет собой определённую многократно повторяющуюся (итерированную) структуру, называемую ячейкой Фейстеля. При переходе от одной ячейки к другой меняется ключ, причём выбор ключа зависит от конкретного алгоритма.

1. Сложение по модулю 2 — булева функция и логическая операция. Результат выполнения операции является истинным только при условии, если является истинным в точности один из аргументов.

2. Дифференциальный криптоанализ — метод криптоанализа симметричных блочных шифров основанный на изучении преобразования разностей между шифруемыми значениями на различных раундах шифрования. Является статистической атакой, в результате работы предлагает список наиболее вероятных ключей шифрования блочного симметричного шифра.

Операции шифрования и расшифрования на каждом этапе очень просты, и при определённой доработке совпадают, требуя только обратного порядка используемых ключей. Шифрование при помощи данной конструкции легко реализуется как на программном уровне, так и на аппаратном, что обеспечивает широкие возможности применения.

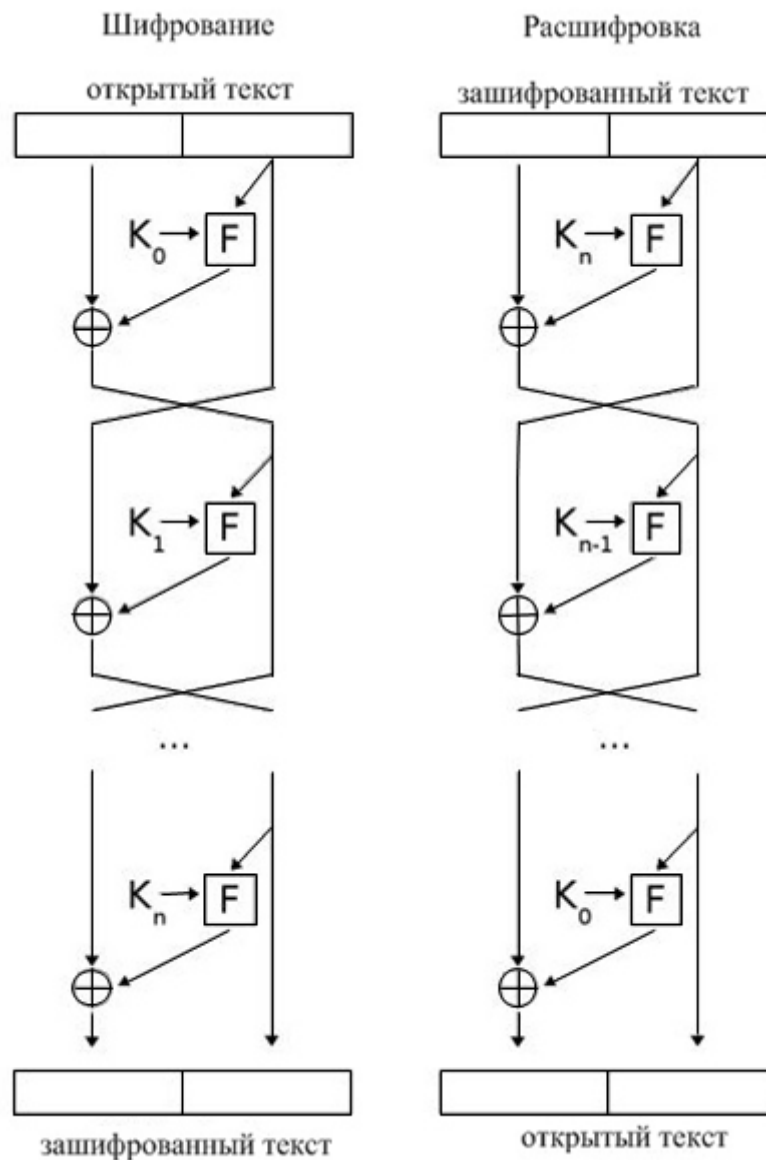


Рис. 1.1. Принцип работы сети Фейстеля

1. Каждый блок разбивается на два «подблока» (левый и правый, соответственно).
2. Исходное заполнение правого блока записывается в левый блок на выходе.
3. Над правым блоком производится криптографическое преобразование с применением ключевых данных.
4. Левый (исходный) и правый (преобразованный) блоки складываются по модулю 2.
5. Полученная комбинация записывается в правый блок на выходе

6. Так повторяется несколько раз.

В ГОСТе ключевая информация состоит из двух структур данных. Помимо собственно ключа, необходимого для всех шифров, она содержит еще и таблицу замен. Ниже приведены основные характеристики ключевых структур ГОСТа:

1. *Таблица замен K* состоит из восьми узлов замены $K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_8$ с памятью на 64 бита каждый (таблица 1.1). Поступающий на блок подстановки 32-разрядный вектор разбивается на восемь последовательно идущих 4-разрядных векторов, каждый из которых преобразуется в 4-разрядный вектор соответствующим узлом замены, представляющим собой таблицу из шестнадцати строк, содержащих по четыре бита заполнения в строке. Входной вектор определяет адрес строки в таблице, заполнение данной строки является выходным вектором. Затем 4-разрядные выходные векторы последовательно объединяются в 32-разрядный вектор. Таким образом, общий объем таблицы замен равен 512 бит (64 байта).

Таблица 1.1. Таблица замен, приведенная в ГОСТ Р 34.11-94 для целей тестирования

Номер S-блока	Значение															
	4	10	9	2	13	8	0	14	6	11	1	12	7	15	5	3
1	4	10	9	2	13	8	0	14	6	11	1	12	7	15	5	3
2	14	11	4	12	6	13	15	10	2	3	8	1	0	7	5	9
3	5	8	1	13	10	3	4	2	14	15	12	7	6	0	9	11
4	7	13	10	1	0	8	9	15	14	4	6	12	11	2	5	3
5	6	12	7	1	5	15	13	8	4	10	9	14	0	3	11	2
6	4	11	10	0	7	2	1	13	3	6	8	5	9	12	15	14
7	13	11	4	1	3	15	5	9	0	10	14	7	6	8	2	12

8	1	15	13	0	5	7	10	4	9	2	3	14	6	11	8	12
---	---	----	----	---	---	---	----	---	---	---	---	----	---	----	---	----

2. При сложении и циклическом сдвиге двоичных векторов старшими разрядами считаются разряды накопителей с большими номерами.

3. При записи ключа $(W_1, W_2, \dots, W_{256})$, $W_q \in \{0,1\}$, $q=1 \div 256$, в ключевое запоминающее устройство (КЗУ) значение W_1 вводится в 1-й разряд накопителя X_0 , значение W_2 вводится во 2-й разряд накопителя X_0 , ... , значение W_{32} вводится в 32-й разряд накопителя X_0 ; значение W_{33} вводится в 1-й разряд накопителя X_1 , значение W_{34} вводится во 2-й разряд накопителя X_1 , ... , значение W_{64} вводится в 32-й разряд накопителя X_1 ; значение W_{65} вводится в 1-й разряд накопителя X_2 и т.д., значение W_{256} вводится в 32-й разряд накопителя X_7 .

3. При перезаписи информации содержимое p -го разряда одного накопителя (сумматора) переписывается в p -й разряд другого накопителя (сумматора).

4. Ключи, определяющие заполнения КЗУ и таблиц блока подстановки K , являются секретными элементами и поставляются в установленном порядке.

Ключ является массивом из восьми 32-битовых элементов кода в ГОСТе элементы ключа используются как 32-разрядные целые числа без знака. Таким образом, размер ключа составляет 256 бит (32 байта). Ключ должен являться массивом статистически независимых битов, принимающих с равной вероятностью значения 0 и 1. При этом некоторые конкретные значения ключа могут оказаться «слабыми», то есть шифр может не обеспечивать заданный уровень стойкости в случае их использования. Однако, предположительно, доля таких значений в общей массе всех возможных ключей ничтожно мала. Поэтому ключи, выработанные с помощью некоторого датчика истинно случайных чисел, будут качественными с вероятностью, отличающейся от единицы на ничтожно малую величину.

1.2. Режимы шифрования

ГОСТ 28147-89 предусматривает следующие режимы шифрования данных:

- простая замена,
- гаммирование,
- гаммирование с обратной связью,
- дополнительный режим выработки имитовставки.

В любом из этих режимов данные обрабатываются блоками по 64 бита, на которые разбивается массив, подвергаемый криптопреобразованию. Однако в двух режимах гаммирования есть возможность обработки неполного блока данных размером меньше 8 байт, что существенно при шифровании массивов данных с произвольным размером, который может быть не кратным 8 байтам.

Схемы программной реализации алгоритма криптографического преобразования приведены в приложении А.

1.3 Основной шаг криптопреобразования

Если внимательно изучить ГОСТ 28147–89, можно заметить, что в нем содержится описание алгоритмов нескольких уровней. На самом верхнем находятся практические алгоритмы, предназначенные для шифрования массивов данных и выработки для них имитовставки. Все они опираются на три алгоритма низшего уровня, называемые в тексте ГОСТа циклами. Эти фундаментальные алгоритмы будут называться «базовые циклы», чтобы отличать их от всех прочих циклов. Они имеют следующие названия и обозначения: (последние приведены в скобках)

- цикл зашифрования (32-3);
- цикл расшифрования (32-Р);
- цикл выработки имитовставки (16-3).

В свою очередь, каждый из базовых циклов представляет собой многократное повторение одной единственной процедуры, называемой «основным шагом криптопреобразования».

Основной шаг криптопреобразования по своей сути является оператором, определяющим преобразование 64-битового блока данных. Дополнительным параметром этого оператора является 32-битовый блок, в качестве которого используется какой-либо элемент ключа. Схема алгоритма основного шага приведена в приложении А (рисунок 1). Рассмотрим подробнее этапы основного шага криптопреобразования:

Шаг 0 Определяет исходные данные для основного шага криптопреобразования.

N – преобразуемый 64-битовый блок данных, в ходе выполнения шага его младшая (N_1) и старшая (N_2) части обрабатываются как отдельные 32-битовые целые числа без знака.

X – 32-битовый элемент ключа;

Шаг 1 Сложение с ключом.

Младшая половина преобразуемого блока складывается по модулю 2^{32} с используемым на шаге элементом ключа, результат передается на следующий шаг;

Шаг 2 Поблочная замена.

32-битовое значение, полученное на предыдущем шаге, интерпретируется как массив из восьми 4-битовых блоков кода (причем CM_1 содержит 4 самых младших, а CM_8 – 4 самых старших бита). Далее значение каждого из восьми блоков заменяется новым, которое выбирается по таблице замен следующим образом: значение блока CM_i меняется на CM_i -тый по порядку элемент i -того узла замены. Другими словами, в качестве замены для значения блока выбирается элемент из таблицы замен с номером строки, равным номеру заменяемого блока, и номером столбца, равным значению заменяемого блока как 4-битового целого неотрицательного числа. Отсюда становится понятным размер таблицы замен: число строк в ней равно числу 4-битовых элементов в 32-битовом блоке данных, то есть восьми, а число столбцов равно числу различных значений 4-битового блока данных, равному, как известно 2^4 (т. е. шестнадцати).

Шаг 3 Циклический сдвиг на 11 бит влево.

Результат предыдущего шага сдвигается циклически на 11 бит в сторону старших разрядов и передается на следующий шаг.

Шаг 4 Побитовое сложение

Значение, полученное на шаге 3, побитно складывается по модулю 2 со старшей половиной преобразуемого блока.

Шаг 5 Сдвиг по цепочке

Младшая часть преобразуемого блока сдвигается на место старшей, а на ее место помещается результат выполнения предыдущего шага.

Шаг 6 Завершение криптопреобразования.

Полученное значение преобразуемого блока возвращается как результат выполнения алгоритма основного шага криптопреобразования.

1.4 Базовые циклы криптографических преобразований

ГОСТ 28147-89 относится к классу блочных шифров, то есть единицей обработки информации в нем является блок данных. Следовательно, вполне логично ожидать, что в нем будут определены алгоритмы для криптографических преобразований, то есть для зашифрования, расшифрования и «учета» в контрольной комбинации одного блока данных. Именно эти алгоритмы и называются *базовыми циклами* ГОСТа, что подчеркивает их фундаментальное значение для построения этого шифра.

Базовые циклы построены из основных шагов криптопреобразования, рассмотренного в предыдущем пункте. В процессе выполнения основного шага используется только один 32-битовый элемент ключа, в то время как ключ ГОСТа содержит восемь таких элементов. Следовательно, чтобы ключ был использован полностью, каждый из базовых циклов должен многократно выполнять основной шаг с различными его элементами. Вместе с тем кажется вполне естественным, что в каждом базовом цикле все элементы ключа должны быть использованы одинаковое число раз, по соображениям стойкости шифра это число должно быть больше одного.

Все сделанные выше предположения, опирающиеся просто на здравый смысл, оказались верными. Базовые циклы заключаются в многократном выполнении основного шага с использованием разных элементов ключа и отличаются друг от друга только числом повторения шага и порядком использования ключевых элементов.

Каждый из циклов имеет собственное буквенно-цифровое обозначение, соответствующее шаблону «n-X», где первый элемент обозначения (n), задает число повторений основного шага в цикле, а второй элемент обозначения (X), буква, задает порядок зашифрования («З») или расшифрования («Р») в использовании ключевых элементов. Цикл расшифрования должен быть обратным циклу зашифрования, то есть последовательное применение этих двух циклов к произвольному блоку должно дать в итоге исходный блок, что отражается следующим соотношением:

$$Ц_{32-Р}(Ц_{32-З}(T))=T, \quad (1.1)$$

где T – произвольный 64-битовый блок данных,

$Ц_X(T)$ – результат выполнения цикла X над блоком данных T

Для выполнения этого условия для алгоритмов, подобных ГОСТу, необходимо и достаточно, чтобы порядок использования ключевых элементов соответствующими циклами был взаимно обратным. Из этого следует интересное следствие: свойство цикла быть обратным другому циклу является взаимным, то есть цикл 32-З является обратным по отношению к циклу 32-Р. Другими словами, зашифрование блока данных теоретически может быть выполнено с помощью цикла расшифрования, в этом случае расшифрование блока данных должно быть выполнено циклом зашифрования. Из двух взаимно обратных циклов любой может быть использован для зашифрования, тогда второй должен быть использован для расшифрования данных, однако стандарт ГОСТ 28147-89 закрепляет роли за циклами и не предоставляет пользователю права выбора в этом вопросе.

Цикл выработки имитовставки (рисунок 3.5) вдвое короче циклов шифрования, порядок использования ключевых элементов в нем такой же, как в первых 16-ти шагах цикла зашифрования, поэтому этот порядок в обозначении цикла кодируется той же самой буквой «З».

Схемы базовых циклов приведены в приложении А (рисунки 2 и 3). Каждый из них принимает в качестве аргумента и возвращает в качестве результата 64-битовый блок данных, обозначенный на схемах «N». Между циклами шифрования и вычисления имитовставки есть еще одно отличие, не упомянутое выше: в конце базовых циклов шифрования старшая и младшая часть блока результата меняются местами, это необходимо для их взаимной обратимости.

1.5. Режим простой замены

Зашифрование в данном режиме заключается в применении цикла З2-З к блокам открытых данных, расшифрование – цикла З2-Р к блокам зашифрованных данных. Это наиболее простой из режимов, 64-битовые блоки данных обрабатываются в нем независимо друг от друга. Схемы алгоритмов зашифрования и расшифрования в режиме простой замены приведены в приложении А (рисунки 4 и 5 соответственно). Размер массива открытых или зашифрованных данных, подвергающийся соответственно зашифрованию или расшифрованию, должен быть кратен 64 битам:

$$|T_o|=|T_{ш}|=64 \cdot n \quad (1.2)$$

где n – целое положительное число

после выполнения операции размер полученного массива данных не изменяется.

Режим шифрования простой заменой имеет следующие особенности:

- Так как блоки данных шифруются независимо друг от друга и от их позиции в массиве данных, при зашифровании двух одинаковых блоков открытого текста получаются одинаковые блоки шифртекста и наоборот. Отмеченное свойство позволит криптоаналитику сделать заключение о

тождественности блоков исходных данных, если в массиве зашифрованных данных ему встретились идентичные блоки, что является недопустимым для серьезного шифра.

- Если длина шифруемого массива данных не кратна 8 байтам (64 битам), возникает проблема, чем и как дополнять последний неполный блок данных массива до полных 64 бит. Эта задача не так проста, как кажется на первый взгляд. Очевидные решения вроде «дополнить неполный блок нулевыми битами» или, более общее, «дополнить неполный блок фиксированной комбинацией нулевых и единичных битов» могут при определенных условиях дать в руки криптоаналитика возможность методами перебора определить содержимое этого самого неполного блока, и этот факт означает снижение стойкости шифра. Кроме того, длина шифртекста при этом изменится, увеличившись до ближайшего целого, кратного 64 битам, что часто бывает нежелательным.

На первый взгляд, перечисленные выше особенности делают практически невозможным использование режима простой замены, ведь он может применяться только для шифрования массивов данных с размером кратным 64 битам, не содержащим повторяющихся 64-битовых блоков. Кажется, что для любых реальных данных гарантировать выполнение указанных условий невозможно. Это почти так, но есть одно очень важное исключение – размер ключа составляет 32 байта, а размер таблицы замен – 64 байта. Кроме того, наличие повторяющихся 8-байтовых блоков в ключе или таблице замен будет говорить об их весьма плохом качестве, поэтому в реальных ключевых элементах такого повторения быть не может. Таким образом, мы выяснили, что режим простой замены вполне подходит для шифрования ключевой информации, тем более, что прочие режимы для этой цели менее удобны, поскольку требуют наличия дополнительного синхронизирующего элемента данных – синхропосылки (см. следующий подпункт). И поэтому ГОСТ

предписывает использовать режим простой замены исключительно для шифрования ключевых данных.

Открытые данные, подлежащие зашифрованию, разбивают на блоки по 64 бита в каждом. Ввод любого блока

$$T_0 = (a_1(0), a_2(0), \dots, a_{31}(0), a_{32}(0), b_1(0), b_2(0), \dots, b_{32}(0)) \quad (1.3)$$

двоичной информации в накопители N_1 и N_2 производится так, что значение $a_1(0)$ вводится в 1-й разряд N_1 , значение $a_2(0)$ вводится во 2-й разряд N_1 и т.д., значение $a_{32}(0)$ вводится в 32-й разряд N_1 ; значение $b_1(0)$ вводится в 1-й разряд N_2 , значение $b_2(0)$ вводится во 2-й разряд N_2 и т.д., значение $b_{32}(0)$ вводится в 32-й разряд N_2 . В результате получают состояние $(a_{32}(0), a_{31}(0), \dots, a_2(0), a_1(0))$ накопителя N_1 и состояние $(b_{32}(0), b_{31}(0), \dots, b_1(0))$ накопителя N_2 .

В КЗУ вводятся 256 бит ключа. Содержимое восьми 32-разрядных накопителей X_0, X_1, \dots, X_7 имеет вид:

$$\begin{aligned} X_0 &= (W_{32}, W_{31}, \dots, W_2, W_1) \\ X_1 &= (W_{64}, W_{63}, \dots, W_{34}, W_{33}) \\ &\dots \\ X_7 &= (W_{256}, W_{255}, \dots, W_{226}, W_{225}) \end{aligned} \quad (1.4)$$

Алгоритм зашифрования 64-разрядного блока открытых данных в режиме простой замены состоит из 32 циклов:

1. В первом цикле начальное заполнение накопителя N_1 суммируется по модулю 2^{32} в сумматоре SM_1 с заполнением накопителя X_0 , при этом заполнение накопителя N_1 сохраняется.

2. Результат суммирования преобразуется в блоке подстановки K и полученный вектор поступает на вход регистра R , где циклически сдвигается на одиннадцать шагов в сторону старших разрядов. Результат сдвига суммируется поразрядно по модулю 2 в сумматоре SM_2 с 32-разрядным заполнением

накопителя N_2 . Полученный в CM_2 результат записывается в M_1 , при этом старое заполнение M_1 переписывается в N_2 . Первый цикл заканчивается.

3. Последующие циклы осуществляются аналогично. При этом во 2-м цикле из КЗУ считывается заполнение X_1 в 3-м цикле из КЗУ считывается заполнение X_2 и т.д., в 8-м цикле из КЗУ считывается заполнение X_7 . В циклах с 9-го по 16-й, а также в циклах с 17-го по 24-й заполнения из КЗУ считываются в том же порядке. В последних восьми циклах с 25-го по 32-й порядок считывания заполнений КЗУ обратный.

Таким образом, при зашифровании в 32 циклах осуществляется следующий порядок выбора заполнений накопителей:

$$\begin{aligned} X_0, X_1, X_2, X_3, X_4, X_5, X_6, X_7, \quad X_0, X_1, X_2, X_3, X_4, X_5, X_6, X_7, \\ X_0, X_1, X_2, X_3, X_4, X_5, X_6, X_7, \quad X_7, X_6, X_5, X_4, X_3, X_2, X_1, X_0 \end{aligned} \quad (1.5)$$

4. В 32 цикле результат из сумматора CM_2 вводится в накопитель N_2 , а в накопителе M_1 сохраняется старое заполнение.

5. Полученные после 32-го цикла зашифрования заполнения накопителей M_1 и N_2 являются блоком зашифрованных данных, соответствующим блоку открытых данных.

Уравнения зашифрования в¹ режиме простой замены имеют вид:

$$\begin{cases} a(j) = (a(j-1) \boxplus X_{(j-1) \pmod{8}})KR \oplus b(j-1) \\ b(j) = a(j-1) \end{cases} \quad (1.6)$$

при $j = 1 \div 24$;

$$\begin{cases} a(j) = (a(j-1) \boxplus X_{(32-j)})KR \oplus b(j-1) \\ b(j) = a(j-1) \end{cases} \quad (1.7)$$

при $j = 25 \div 31$;

$$a(32) = a(31)$$

$$b(32) = (a(31) \boxplus X_0)KR \oplus b(31) \quad (1.8)$$

при $j = 32$,

где $a(0) = (a_{32}(0), a_{31}(0), +, a_1(0))$ - начальное заполнение M_1 перед первым циклом зашифрования;

$b(0) = (b_{32}(0), b_{31}(0), +, b_1(0))$ - начальное заполнение M_2 перед первым циклом зашифрования;

$a(j) = (a_{32}(j), a_{31}(j), +, a_1(j))$ - заполнение M_1 после j -го цикла зашифрования;

$b(j) = (b_{32}(j), b_{31}(j), +, b_1(j))$ - заполнение M_2 после j -го цикла зашифрования, $j = 1 \div 32$.

Знак \oplus означает поразрядное суммирование 32-разрядных векторов по модулю 2.

Знак \boxplus означает суммирование 32-разрядных векторов по модулю 2^{32} (правила суммирования по модулю 2^{32} приведены в приложении Б);

R - операция циклического сдвига на одиннадцать шагов в сторону старших разрядов, т.е.

1. Уравнение зашифрования – это соотношение, выражающее процесс образования зашифрованных данных из открытых данных в результате преобразований, заданных алгоритмом криптографического преобразования

$$\begin{aligned} R(r_{32}, r_{31}, r_{30}, r_{29}, r_{28}, r_{27}, r_{26}, r_{25}, r_{24}, r_{23}, r_{22}, r_{21}, r_{20}, \dots, r_2, r_1) = \\ = (r_{21}, r_{20}, \dots, r_2, r_1, r_{32}, r_{31}, r_{30}, r_{29}, r_{28}, r_{27}, r_{26}, r_{25}, r_{24}, r_{23}, r_{22}) \end{aligned} \quad (1.9)$$

б. 64-разрядный блок зашифрованных данных $T_{\text{ш}}$ выводится из накопителей M_1, M_2 в следующем порядке: из 1-го, 2-го, ..., 32-го разрядов накопителя M_1 затем из 1-го, 2-го, ..., 32-го разрядов накопителя M_2 , т.е.

$$T_{\text{ш}} = (a_1(32), a_2(32), \dots, a_{32}(32), b_1(32), b_2(32), \dots, b_{32}(32)) \quad (1.10)$$

7. Остальные блоки открытых данных в режиме простой замены зашифровываются аналогично.

Расшифрование зашифрованных данных в режиме простой замены:

1. Криптосхема, реализующая алгоритм расшифрования в режиме простой замены, имеет тот же вид, что и при зашифровании. В КЗУ вводятся 256 бит того же ключа, на котором осуществлялось зашифрование. Зашифрованные данные, подлежащие расшифрованию, разбиты на блоки по 64 бита в каждом.

Ввод любого блока

$$T_{\text{ш}} = (a_1(32), a_2(32), \dots, a_{32}(32), b_1(32), b_2(32), \dots, b_{32}(32)) \quad (1.11)$$

в накопители N_1 и N_2 производятся так, что значение $a_1(32)$ вводится в 1-й разряд N_1 , значение $a_2(32)$ вводится во 2-й разряд N_1 и т.д., значение $a_{32}(32)$ вводится в 32-й разряд N_1 ; значение $b_1(32)$ вводится в 1-й разряд N_2 и т.д., значение $b_{32}(32)$ вводится в 32-й разряд N_2 .

2. Расшифрование осуществляется по тому же алгоритму, что и зашифрование открытых данных, с тем изменением, что заполнения накопителей X_0, X_1, \dots, X_7 считываются из КЗУ в циклах расшифрования в следующем порядке:

$$\begin{aligned} X_0, X_1, X_2, X_3, X_4, X_5, X_6, X_7, & \quad X_7, X_6, X_5, X_4, X_3, X_2, X_1, X_0, \\ X_7, X_6, X_5, X_4, X_3, X_2, X_1, X_0, & \quad X_7, X_6, X_5, X_4, X_3, X_2, X_1, X_0 \end{aligned} \quad (1.12)$$

3. Уравнения расшифрования¹ имеют вид:

$$\begin{cases} a(32-j) = (a(32-j+1) \boxplus X_{j-1})KR \oplus b(32-j+1) \\ b(32-j) = a(32-j+1) \end{cases} \quad (1.13)$$

при $j = 1+8$;

$$\begin{cases} a(32-j) = (a(32-j+1) \boxplus X_{(32-j)(\text{mod}8)})KR \oplus b(32-j+1) \\ b(32-j) = a(32-j+1) \end{cases} \quad (1.14)$$

при $j = 9+31$;

$$a(0) = a(1)$$

$$b(0) = (a(1) \boxplus X_0)KR \oplus b(1) \quad (1.15)$$

при $j = 32$.

4. Полученные после 32 циклов работы заполнения накопителей M_1 и N_2 составляют блок открытых данных:

$$T_0 = (a_1(0), a_2(0), +, a_{32}(0), b_1(0), b_2(0), +, b_{32}(0)) \quad (1.16)$$

соответствующий блоку зашифрованных данных, при этом значение $a_1(0)$ блока

T_0 соответствует содержимому 1-го разряда M_1 , значение $a_2(0)$ соответствует содержимому 2-го разряда M_1 и т.д., значение $a_{32}(0)$ соответствует содержимому 32-го

1. Уравнение расшифрования – это соотношение, выражающее процесс образования открытых данных из зашифрованных данных в результате преобразований, заданных алгоритмом криптографического преобразования

разряда M_1 ; значение $b_1(0)$ соответствует содержимому 1-го разряда N_2 , значение $b_2(0)$ соответствует содержимому 2-го разряда N_2 и т.д., значение $b_{32}(0)$ соответствует содержимому 32-го разряда N_2 .

5. Аналогично расшифровываются остальные блоки зашифрованных данных.

Алгоритм зашифрования в режиме простой замены 64-битового блока T_0 обозначается через A , т.е.

$$A(T_0) = A(a(0), b(0)) = (a(32), b(32)) = T_{\text{ш}} \quad (1.17)$$

1.6. Режим гаммирования

Как же можно избавиться от недостатков режима простой замены? Для этого необходимо сделать возможным шифрование блоков с размером менее 64 бит и обеспечить зависимость блока шифртекста от его номера, иными словами,

рандомизировать процесс шифрования. В ГОСТе это достигается двумя различными способами в двух режимах шифрования, предусматривающих гаммирование.

Гаммирование – это процесс наложения по определенному закону гаммы шифра на открытые данные.

Гамма шифра – это псевдослучайная двоичная последовательность, вырабатываемая по заданному алгоритму для зашифрования открытых данных и расшифрования зашифрованных данных.

Для наложения гаммы при зашифровании и ее снятия при расшифровании должны использоваться взаимно обратные бинарные операции.

В ГОСТе для этой цели используется операция побитового сложения по модулю 2, поскольку она является обратной самой себе и, к тому же, наиболее просто реализуется аппаратно. Гаммирование решает обе упомянутые проблемы: во-первых, все элементы гаммы различны для реальных шифруемых массивов и, следовательно, результат зашифрования даже двух одинаковых блоков в одном массиве данных будет различным. Во-вторых, хотя элементы гаммы и вырабатываются одинаковыми порциями в 64 бита, использоваться может и часть такого блока с размером, равным размеру шифруемого блока.

Особенности гаммирования как режима шифрования:

1. Одинаковые блоки в открытом массиве данных дадут при зашифровании различные блоки шифртекста, что позволит скрыть факт их идентичности.
2. Поскольку наложение гаммы выполняется побитно, шифрование неполного блока данных легко выполнимо как шифрование битов этого неполного блока, для чего используется соответствующие биты блока гаммы. Так, для зашифрования неполного блока в 1 бит согласно стандарту следует использовать самый младший бит из блока гаммы.
3. Синхропосылка, использованная при зашифровании, каким-то образом должна быть передана для использования при расшифровании. Это может быть достигнуто следующими путями:

- хранить или передавать синхропосылку вместе с зашифрованным массивом данных, что приведет к увеличению размера массива данных при зашифровании на размер синхропосылки, то есть на 8 байт;
- использовать predetermined значение синхропосылки или вырабатывать ее синхронно источником и приемником по определенному закону, в этом случае изменение размера передаваемого или хранимого массива данных отсутствует

4. Биты массива данных шифруются независимо друг от друга. Таким образом, каждый бит шифртекста зависит от соответствующего бита открытого текста и, естественно, порядкового номера бита в массиве. Из этого вытекает, что изменение бита шифртекста на противоположное значение приведет к аналогичному изменению бита открытого текста на противоположный. Данное свойство дает злоумышленнику возможность, воздействуя на биты шифртекста, вносить предсказуемые и даже целенаправленные изменения в соответствующий открытый текст, получаемый после его расшифрования, не обладая при этом секретным ключом. Это иллюстрирует хорошо известный в криптологии факт, что *секретность* и *аутентичность* различные свойства криптографических систем.

Зашифрование данных в режиме гаммирования (схема программной реализации алгоритма приведена в приложении А рисунок 6):

1. Открытые данные, разбитые на 64-разрядные блоки $T_0^{(1)}, T_0^{(2)} \dots, T_0^{(M-1)}, T_0^{(M)}$ зашифровываются в режиме гаммирования путем поразрядного суммирования по модулю 2 в сумматоре CM_5 с гаммой шифра $\Gamma_{\text{ш}}$, которая вырабатывается блоками по 64 бита, т.е.

$$\Gamma_{\text{ш}} = (\Gamma_{\text{ш}}^{(1)}, \Gamma_{\text{ш}}^{(2)}, \dots, \Gamma_{\text{ш}}^{(M-1)}, \dots, \Gamma_{\text{ш}}^{(M)}) \quad (1.18)$$

где M - определяется объемом шифруемых данных.

$\Gamma_{\text{ш}}^{(i)}$ - i -й 64-разрядный блок,

$$i = 1 \div M$$

Число двоичных разрядов в блоке $T_0^{(M)}$ может быть меньше 64, при этом неиспользованная для зашифрования часть гаммы шифра из блока $\Gamma_{\text{ш}}^{(M)}$ отбрасывается.

2. В КЗУ вводятся 256 бит ключа. В накопители N_1 , N_2 вводится 64-разрядная двоичная последовательность (синхропосылка¹) $S = (S_1, S_2, \dots, S_{64})$, являющаяся исходным заполнением этих накопителей для последующей выработки M блоков гаммы шифра. Синхропосылка вводится в N_1 и N_2 так, что значение S_1 вводится в 1-й разряд N_1 , значение S_2 вводится во 2-й разряд N_1 и т.д., значение S_{32} вводится в 32-й разряд N_1 ; значение S_{33} вводится в 1-й разряд N_2 , значение S_{34} вводится во 2-й разряд N_2 и т.д., значение S_{64} вводится в 32-й разряд N_2 .

3. Исходное заполнение накопителей N_1 и N_2 (синхропосылка S) зашифровывается в режиме простой замены. Результат зашифрования $A(S) = Y_0$, Z_0 переписывается в 32-разрядные накопители N_3 и N_4 , так, что заполнение N_1 переписывается в N_3 , а заполнение N_2 переписывается в N_4 .

4. Заполнение накопителя N_4 суммируется по модулю $(2^{32} - 1)$ в сумматоре CM_4 с 32-разрядной константой C_1 из накопителя N_6 , результат записывается в N_4 . Правила суммирования по модулю $(2^{32} - 1)$ приведены в приложении Б.

1. Синхропосылка (вектор инициализации) – значения исходных открытых параметров алгоритма криптографического преобразования т. е. случайное число, которое регулярно обновляется, передается по каналу управления и используется для инициализации алгоритма шифрования.

Заполнение накопителя N_3 суммируется по модулю 2^{32} в сумматоре CM_3 в 32-разрядной константой C_2 из накопителя N_5 , результат записывается в N_3 .

5. Заполнение N_3 переписывается в N_1 , а заполнение N_4 переписывается в N_2 , при этом заполнение N_3 , N_4 сохраняется.

6. Заполнение N_1 и N_2 зашифровывается в режиме простой замены. Полученное в результате зашифрования заполнение N_1 , N_2 образует первый 64-разрядный блок гаммы шифра $T_{\text{ш}}^{(1)}$, который суммируется поразрядно по модулю 2 в сумматоре CM_5 с первым 64-разрядным блоком открытых данных

$$T_0^{(1)} = (t_1^{(1)}, t_2^{(1)}, \dots, t_{63}^{(1)}, t_{64}^{(1)}). \quad (1.19)$$

В результате суммирования получается 64-разрядный блок зашифрованных данных

$$T_{\text{ш}}^{(1)} = (\tau_1^{(1)}, \tau_2^{(1)}, \dots, \tau_{63}^{(1)}, \tau_{64}^{(1)}). \quad (1.20)$$

Значение $\tau_1^{(1)}$ блока $T_{\text{ш}}^{(1)}$ является результатом суммирования по модулю 2 в CM_5 значения $t_1^{(1)}$ из блока $T_0^{(1)}$ со значением 1-го разряда N_1 , значение $\tau_2^{(1)}$ блока $T_{\text{ш}}^{(1)}$ является результатом суммирования по модулю 2 в CM_5 значения $t_2^{(1)}$ из блока $T_0^{(1)}$ со значением 2-го разряда N_1 и т.д., значение $\tau_{64}^{(1)}$ блока $T_{\text{ш}}^{(1)}$ является результатом суммирования по модулю 2 в CM_5 значения $t_{64}^{(1)}$ из блока $T_0^{(1)}$ со значением 32-го разряда N_2 .

7. Для получения следующего 64-разрядного блока гаммы шифра $T_{\text{ш}}^{(2)}$ заполнение N_4 суммируется по модулю $(2^{32}-1)$ в сумматоре CM_4 с константой C_1 из N_6 , заполнение N_3 суммируется по модулю 2^{32} в сумматоре CM_3 с константой C_2 из N_5 .

Новое заполнение N_3 переписывается в N_1 , а новое заполнение N_4 переписывается в N_2 , при этом заполнение N_3 и N_4 сохраняется.

8. Заполнение N_1 и N_2 зашифровывается в режиме простой замены. Полученное в результате зашифрования заполнение N_1, N_2 образует второй 64-разрядный блок гаммы шифра $\Gamma_{\text{ш}}^{(2)}$, который суммируется поразрядно по модулю 2 в сумматоре CM_5 со вторым блоком открытых данных $T_0^{(2)}$. Аналогично вырабатываются блоки гаммы шифра $\Gamma_{\text{ш}}^{(3)}, \Gamma_{\text{ш}}^{(4)} \dots, \Gamma_{\text{ш}}^{(M)}$ и зашифровываются блоки открытых данных $T_0^{(3)}, T_0^{(4)} \dots, T_0^{(M)}$. Если длина последнего M -го блока открытых данных $T_0^{(M)}$ меньше 64 бит, то из последнего M -го блока гаммы шифра $\Gamma_{\text{ш}}^{(M)}$ для зашифрования используется только соответствующее число разрядов гаммы шифра, остальные разряды отбрасываются.

9. В канал связи или память ЭВМ передаются синхросылка S и блоки зашифрованных данных $T_{\text{ш}}^{(1)}, T_{\text{ш}}^{(2)}, \dots, T_{\text{ш}}^{(M)}$.

10. Уравнение зашифрования имеет вид:

$$T_{\text{ш}}^{(i)} = A(Y_{i-1} \boxplus C_2, Z_{i-1} \boxplus C_1) \oplus T_0^{(i)} = \Gamma_{\text{ш}}^i \oplus T_0^{(i)}, \quad (1.21)$$

где $i = 1 \dots M$

\boxplus - означает суммирование 32-разрядных заполнений по модулю $(2^{32} - 1)$;

\boxplus - поразрядное суммирование по модулю 2 двух заполнений;

Y_i - содержимое накопителя N_3 после зашифрования i -го блока открытых данных $T_0^{(i)}$;

Z_i - содержимое накопителя N_4 после зашифрования i -го блока открытых данных $T_0^{(i)}$;

$$(Y_0, Z_0) = A(S)$$

Расшифрование зашифрованных данных в режиме гаммирования (схема программной реализации алгоритма приведена в приложении А рисунок 7):

При расшифровании криптосхема имеет тот же вид, что и при зашифровании. В КЗУ вводятся 256 бит ключа, с помощью которого осуществлялось зашифрование данных $T_0^{(1)}, T_0^{(2)}, \dots, T_0^{(M)}$. Синхропосылка S вводится в накопители N_1 и N_2 и осуществляется процесс выработки M блоков гаммы шифра $\Gamma_{\text{ш}}^{(1)}, \Gamma_{\text{ш}}^{(2)}, \dots, \Gamma_{\text{ш}}^{(M)}$. Блоки зашифрованных данных $T_{\text{ш}}^{(1)}, T_{\text{ш}}^{(2)}, \dots, T_{\text{ш}}^{(M)}$ суммируются поразрядно по модулю 2 в сумматоре CM_5 с блоками гаммы шифра, в результате получают блоки открытых данных $T_0^{(1)}, T_0^{(2)}, \dots, T_0^{(M)}$, при этом $T_0^{(M)}$ может содержать меньше 64 разрядов.

Уравнение расшифрования имеет вид:

$$T_0^{(i)} = A(Y_{i-1} \boxplus C_2, Z_{i-1} \boxplus C_1) \oplus T_{\text{ш}}^{(i)} = \Gamma_{\text{ш}}^{(i)} \oplus T_{\text{ш}}^{(i)},$$

$$i = 1 + M \quad (1.22)$$

1.7. Режим гаммирования с обратной связью

Данный режим очень похож на режим гаммирования и отличается от него только способом выработки элементов гаммы – очередной элемент гаммы вырабатывается как результат преобразования по циклу 32-3 предыдущего блока зашифрованных данных, а для зашифрования первого блока массива данных элемент гаммы вырабатывается как результат преобразования по тому же циклу синхропосылки. Этим достигается зацепление блоков – каждый блок шифртекста в этом режиме зависит от соответствующего и всех предыдущих блоков открытого текста, это видно и в уравнении режима шифрования (расшифрования) гаммирования с обратной связью (формула 1.23). Поэтому данный режим иногда называется *гаммированием с зацеплением блоков*. На стойкость шифра факт зацепления блоков не оказывает никакого влияния.

Шифрование в режиме гаммирования с обратной связью обладает теми же особенностями, что и шифрование в режиме обычного гаммирования, за исключением влияния искажений шифртекста на соответствующий открытый текст (свойство 4).

$$T_i^o = T_i^u \oplus U_{32-3}(T_{i-1}^u) \quad (1.23)$$

Если в режиме обычного гаммирования изменения в определенных битах шифртекста влияют только на соответствующие биты открытого текста, то в режиме гаммирования с обратной связью картина несколько сложнее. Как видно из соответствующего уравнения, при расшифровании блока данных в режиме гаммирования с обратной связью, блок открытых данных зависит от соответствующего и предыдущего блоков зашифрованных данных. Поэтому, если внести искажения в зашифрованный блок, то после расшифрования искаженными окажутся два блока открытых данных – соответствующий и следующий за ним, причем искажения в первом случае будут носить тот же характер, что и в режиме гаммирования, а во втором случае – как в режиме простой замены. Другими словами, в соответствующем блоке открытых данных искаженными окажутся те же самые биты, что и в блоке шифрованных данных, а в следующем блоке открытых данных все биты независимо друг от друга с вероятностью 1/2 изменят свои значения.

Зашифрование (схема программной реализации алгоритма приведена в приложении А рисунок 8):

1. Открытые данные, разбитые на 64-разрядные блоки $T_0^{(1)}, \dots, T_0^{(M)}$, зашифровываются в режиме гаммирования с обратной связью путем поразрядного суммирования по модулю 2 в сумматоре CM_5 с гаммой шифра $\Gamma_{ш}$, которая вырабатывается блоками по 64 бита, т.е.

$$\Gamma_{ш} = (\Gamma_{ш}^{(1)}, \Gamma_{ш}^{(2)}, \dots, \Gamma_{ш}^{(M)}), \quad (2.24)$$

где M определяется объемом открытых данных,

$\Gamma_{ш}^{(i)}$ - i -й 64-разрядный блок,

$i = 1 \div M$.

Число двоичных разрядов в блоке $T_0^{(M)}$ может быть меньше 64.

2. В КЗУ вводятся 256 бит ключа. Синхропосылка

$$S = (S_1, S_2, \dots, S_{64}) \quad (1.25)$$

из 64 бит вводится в N_1 и N_2 .

3. Исходное заполнение N_1 и N_2 зашифровывается в режиме простой замены. Полученное в результате зашифрования заполнение N_1 и N_2 образует первый 64-разрядный блок гаммы шифра $\Gamma_{\text{ш}}^{(1)} = A(S)$, который суммируется поразрядно по модулю 2 в сумматоре CM_5 с первым 64-разрядным блоком открытых данных

$$T_0^{(1)} = (t_1^{(1)}, t_2^{(1)}, \dots, t_{64}^{(1)}) \quad (1.26)$$

В результате получается 64-разрядный блок зашифрованных данных

$$T_{\text{ш}}^{(1)} = (\tau_1^{(1)}, \tau_2^{(1)}, \dots, \tau_{64}^{(1)}) \quad (1.27)$$

4. Блок зашифрованных данных $T_{\text{ш}}^{(1)}$ одновременно является также исходным состоянием N_1, N_2 для выработки второго блока гаммы шифра $\Gamma_{\text{ш}}^{(2)}$ и по обратной связи записывается в указанные накопители. При этом значение $\tau_1^{(1)}$ вводится в 1-й разряд N_1 , значение $\tau_2^{(1)}$ вводится во 2-й разряд N_1 и т.д., значение $\tau_{32}^{(1)}$ вводится в 32-й разряд N_1 ; значение $\tau_{33}^{(1)}$ вводится в 1-й разряд N_2 , значение $\tau_{34}^{(1)}$ вводится во 2-й разряд N_2 и т.д., значение $\tau_{64}^{(1)}$ вводится в 32-й разряд N_2 .

5. Заполнение N_1, N_2 зашифровывается в режиме простой замены. Полученное в результате зашифрования заполнение N_1, N_2 образует второй 64-разрядный блок гаммы шифра $\Gamma_{\text{ш}}^{(2)}$, который суммируется поразрядно по модулю 2 в сумматоре CM_5 со вторым блоком открытых данных $T_0^{(2)}$.

6. Выработка последующих блоков гаммы шифра $\Gamma_{\text{ш}}^{(i)}$ и зашифрование соответствующих блоков открытых данных $T_0^{(i)}$ ($i = 3 + M$) производится аналогично. Если длина последнего M -го блока открытых данных $T_0^{(M)}$ меньше

64 разрядов, то из $\Gamma_{\text{ш}}^{(M)}$ используется только соответствующее число разрядов гаммы шифра, остальные разряды отбрасываются.

7. Уравнения зашифрования в режиме гаммирования с обратной связью имеют вид:

$$\begin{cases} T_{\text{ш}}^{(1)} = A(S) \oplus T_0^{(1)} = \Gamma_{\text{ш}}^{(1)} \oplus T_0^{(1)} \\ T_{\text{ш}}^{(i)} = A(T_{\text{ш}}^{(i-1)}) \oplus T_0^{(i)} = \Gamma_{\text{ш}}^{(i)} \oplus T_0^{(i)}, \quad i = 2 + M. \end{cases} \quad (1.28)$$

8. В канал связи или память ЭВМ передаются синхропосылка S и блоки зашифрованных данных $T_{\text{ш}}^{(1)}, T_{\text{ш}}^{(2)}, \dots, T_{\text{ш}}^{(M)}$.

Расшифрование зашифрованных данных в режиме гаммирования с обратной связью (схема программной реализации алгоритма приведена в приложении А рисунок 9):

При расшифровании криптосхема имеет тот же вид, что и при зашифровании.

1. В КЗУ вводятся 256 бит того же ключа, на котором осуществлялось зашифрование $T_0^{(1)}, T_0^{(2)}, \dots, T_0^{(M)}$. Синхропосылка S вводится в N_1 и N_2 .

2. Исходное заполнение N_1, N_2 (синхропосылка S) зашифровывается в режиме простой замены. Полученное в результате зашифрования заполнение N_1, N_2 образует первый блок гаммы шифра $\Gamma_{\text{ш}}^{(1)} = A(S)$, который суммируется поразрядно по модулю 2 в сумматоре CM_5 с блоком зашифрованных данных $T_{\text{ш}}^{(1)}$. В результате получается первый блок открытых данных $T_0^{(1)}$.

3. Блок зашифрованных данных $T_{\text{ш}}^{(1)}$ является исходным заполнением N_1, N_2 для выработки второго блока гаммы шифра $\Gamma_{\text{ш}}^{(2)}$. Блок $T_{\text{ш}}^{(1)}$ записывается в N_1, N_2 . Полученное заполнение N_1, N_2 зашифровывается в режиме простой замены, полученный в результате блок $\Gamma_{\text{ш}}^{(2)}$ суммируется поразрядно по модулю 2 в сумматоре CM_5 со вторым блоком зашифрованных данных $T_{\text{ш}}^{(2)}$. В результате получается блок открытых данных $T_{\text{ш}}^{(2)}$.

Аналогично в N_1, N_2 последовательно записываются блоки зашифрованных данных $T_{\text{ш}}^{(2)}, T_{\text{ш}}^{(3)}, \dots, T_{\text{ш}}^{(M-1)}$, из которых в режиме простой замены вырабатываются блоки гаммы шифра $\Gamma_{\text{ш}}^{(3)}, \Gamma_{\text{ш}}^{(4)}, \dots, \Gamma_{\text{ш}}^{(M)}$. Блоки гаммы шифра суммируются поразрядно по модулю 2 в сумматоре SM_5 с блоками зашифрованных данных $T_{\text{ш}}^{(3)}, T_{\text{ш}}^{(4)}, \dots, T_{\text{ш}}^{(M)}$, в результате получают блоки открытых данных $T_0^{(3)}, T_0^{(4)} \dots, T_0^{(M)}$, при этом длина последнего блока открытых данных $T_0^{(M)}$ может содержать меньше 64 разрядов.

Уравнения расшифрования в режиме гаммирования с обратной связью имеют вид:

$$\begin{cases} T_0^{(1)} = A(S) \oplus T_{\text{ш}}^{(1)} = \Gamma_{\text{ш}}^{(1)} \oplus T_{\text{ш}}^{(1)} \\ T_0^{(i)} = A(T_{\text{ш}}^{(i-1)}) \oplus T_{\text{ш}}^{(i)} = \Gamma_{\text{ш}}^{(i)} \oplus T_{\text{ш}}^{(i)}, \quad i = 2 + M \end{cases} \quad (1.29)$$

1.8. Режим выработки имитовставки

Для решения задачи обнаружения искажений в зашифрованном массиве данных с заданной вероятностью в ГОСТе предусмотрен дополнительный режим криптографического преобразования – выработка имитовставки (приложение А рисунок 10).

Целью использования имитовставки является обнаружение всех случайных или преднамеренных изменений в массиве информации. Для потенциального злоумышленника две следующие задачи практически неразрешимы, если он не владеет ключом:

- вычисление имитовставки для заданного открытого массива информации;
- подбор открытых данных под заданную имитовставку

В качестве имитовставки берется часть блока, полученного на выходе, обычно – 32 его младших бита. При выборе размера имитовставки надо принимать во внимание, что вероятность успешного навязывания ложных данных равна величине $2^{-|I|}$ на одну попытку подбора, если в распоряжении

злоумышленника нет более эффективного метода подбора, чем простое угадывание. При использовании имитовставки размером 32 бита эта вероятность равна $2^{-32} \approx 0.23 \cdot 10^{-9}$.

1. Для обеспечения имитозащиты открытых данных, состоящих из M 64-разрядных блоков $T_0^{(1)}, T_0^{(2)}, \dots, T_0^{(M)}$, $M \geq 2$, вырабатывается дополнительный блок из l бит (имитовставка I_l). Процесс выработки имитовставки единообразен для всех режимов шифрования.

2. Первый блок открытых данных

$$T_0^{(1)} = (t_1^{(1)}, t_2^{(1)}, \dots, t_{64}^{(1)}) = (a_1^{(1)}(0), a_2^{(1)}(0), \dots, a_{32}^{(1)}(0), b_1^{(1)}(0), b_2^{(1)}(0), \dots, b_{32}^{(1)}(0)) \quad (1.30)$$

записывается в накопители N_1 и N_2 , при этом значение $t_1^{(1)} = a_1^{(1)}(0)$ вводится в 1-й разряд N_1 , значение $t_2^{(1)} = a_2^{(1)}(0)$ вводится во 2-й разряд N_1 и т.д., значение $t_{32}^{(1)} = a_{32}^{(1)}(0)$ вводится в 32-й разряд N_1 ; значение $t_{33}^{(1)} = b_1^{(1)}(0)$ вводится в 1-й разряд N_2 и т.д., значение $t_{64}^{(1)} = b_{32}^{(1)}(0)$ вводится в 32-й разряд N_2 .

3. Заполнение N_1 и N_2 подвергается преобразованию, соответствующему первым 16 циклам алгоритма зашифрования в режиме простой замены. В КЗУ при этом находится тот же ключ, которым зашифровываются блоки открытых данных $T_0^{(1)}, T_0^{(2)}, \dots, T_{\text{ш}}^{(M)}$ в соответствующие блоки зашифрованных данных $T_{\text{ш}}^{(1)}, T_{\text{ш}}^{(2)}, \dots, T_{\text{ш}}^{(M)}$.

Полученное после 16 циклов работы заполнение N_1 и N_2 , имеющее вид

$$(a_1^{(1)}(16), a_2^{(1)}(16), \dots, a_{32}^{(1)}(16), b_1^{(1)}(16), b_2^{(1)}(16), \dots, b_{32}^{(1)}(16)), \quad (1.31)$$

суммируется в CM_5 по модулю 2 со вторым блоком $T_0^{(2)} = (t_1^{(2)}, t_2^{(2)}, \dots, t_{64}^{(2)})$.

Результат суммирования

$$\begin{aligned} & (a_1^{(1)}(16) \oplus t_1^{(2)}, a_2^{(1)}(16) \oplus t_2^{(2)}, \dots, a_{32}^{(1)}(16) \oplus t_{32}^{(2)}, \\ & b_1^{(1)}(16) \oplus t_{33}^{(2)}, b_2^{(1)}(16) \oplus t_{34}^{(2)}, \dots, b_{32}^{(1)}(16) \oplus t_{64}^{(2)} = \\ & = (a_1^{(2)}(0), a_2^{(2)}(0), \dots, a_{32}^{(2)}(0), b_1^{(2)}(0), b_2^{(2)}(0), \dots, b_{32}^{(2)}(0)) \quad (1.32) \end{aligned}$$

заносится в N_1 и N_2 и подвергается преобразованию, соответствующему первым 16 циклам алгоритма зашифрования в режиме простой замены.

4. Полученное заполнение N_1 и N_2 суммируется по CM_5 по модулю 2 с третьим блоком $T_0^{(3)}$ и т.д., последний блок $T_0^{(m)} = (t_1^{(m)}, t_2^{(m)}, +, t_{64}^{(m)})$, при необходимости дополненный до полного 64-разрядного блока нулями, суммируется в CM_5 по модулю 2 с заполнением N_1, N_2

$$(a_1^{(m-1)}(16), a_2^{(m-1)}(16), \dots, a_{32}^{(m-1)}(16), b_1^{(m-1)}(16), b_2^{(m-1)}(16), \dots, b_{32}^{(m-1)}(16))$$

Результат суммирования

$$\begin{aligned} & (a_1^{(m-1)}(16) \oplus t_1^{(m)}, a_2^{(m-1)}(16) \oplus t_2^{(m)}, \dots, a_{32}^{(m-1)}(16) \oplus t_{32}^{(m)}, \\ & b_1^{(m-1)}(16) \oplus t_{33}^{(m)}, b_2^{(m-1)}(16) \oplus t_{34}^{(m)}, \dots, b_{32}^{(m-1)}(16) \oplus t_{64}^{(m)} = \\ & = (a_1^{(m)}(0), a_2^{(m)}(0), \dots, a_{32}^{(m)}(0), b_1^{(m)}(0), b_2^{(m)}(0), \dots, b_{32}^{(m)}(0)) \end{aligned} \quad (1.33)$$

заносится в N_1, N_2 и зашифровывается в режиме простой замены по первым 16 циклам работы алгоритма. Из полученного заполнения накопителей N_1 и N_2

$$(a_1^{(m)}(16), a_2^{(m)}(16), \dots, a_{32}^{(m)}(16), b_1^{(m)}(16), b_2^{(m)}(16), \dots, b_{32}^{(m)}(16))$$

выбирается отрезок I_l (имитовставка) длиной l бит:

$$I_l = [a_{32-l+1}^{(m)}(16), a_{32-l+2}^{(m)}(16), \dots, a_{32}^{(m)}(16)] \quad (1.34)$$

Имитовставка I_l передается по каналу связи или в память вычислительного комплекса в конце зашифрованных данных, т.е.

$$T_{\text{ш}}^{(1)}, T_{\text{ш}}^{(2)}, +, T_{\text{ш}}^{(m)}, I_l. \quad (1.35)$$

Поступившие зашифрованные данные $T_{\text{ш}}^{(1)}, T_{\text{ш}}^{(2)}, +, T_{\text{ш}}^{(m)}$ расшифровываются, из полученных блоков открытых данных $T_0^{(1)}, T_0^{(2)}, +, T_0^{(m)}$ вырабатывается имитовставка I_l , которая затем сравнивается с имитовставкой I_l , полученной вместе с зашифрованными данными из канала связи или из памяти вычислительного комплекса. В случае несовпадения имитовставок полученные блоки открытых данных $T_0^{(1)}, T_0^{(2)}, +, T_0^{(m)}$ считают ложными.

4. Порядок выполнения работы

Криптоанализ алгоритма

В 1994 г. описание алгоритма ГОСТ 28147-89 было переведено на английский язык и опубликовано в статье Волонгонгского университета (Австралия) «Советский алгоритм шифрования». Именно после этого стали появляться результаты его анализа, выполненного зарубежными специалистами; однако, в течение значительного времени не было найдено каких-либо атак, приближающихся к практически осуществимым.

4.1 Анализ таблиц замен

Поскольку таблицы замен в стандарте не приведены, в ряде работ высказывается предположение, что «компетентная организация» может выдать как «хорошие», так и «плохие» таблицы замен. Ясно, что криптостойкость алгоритма во многом зависит от свойств используемых таблиц замен, соответственно, существуют слабые таблицы замен, применение которых может упростить вскрытие алгоритма. Тем не менее, возможность использования различных таблиц замен кажется весьма достойной идеей, в пользу которой можно привести два следующих факта из истории стандарта шифрования США DES:

- Атаки с помощью как линейного, так и дифференциального криптоанализа алгоритма DES используют конкретные особенности таблиц замен. При использовании других таблиц криптоанализ придется начинать сначала
- Были попытки усилить DES против линейного и дифференциального криптоанализа путем использования более стойких таблиц замен. Такие таблицы, действительно более стойкие, были предложены, например, в алгоритме 3DES. Но, увы, заменить DES на 3DES было невозможно, поскольку таблицы замен жестко определены в стандарте, соответственно,

реализации алгоритма наверняка не поддерживают возможность смены таблиц на другие

Однако, в работе доказано, что секретные таблицы замен могут быть вычислены с помощью следующей атаки, которая может быть применена практически:

1. Устанавливается нулевой ключ и выполняется поиск «нулевого вектора». Этот этап занимает порядка 2^{32} операций шифрования.
2. С помощью нулевого вектора вычисляются значения таблиц замен, что занимает не более 2^{11} операций.

4.2. Модификации алгоритма и их анализ

Алгоритмы:

- GOST-H, в котором, относительно оригинального алгоритма, изменен порядок использования подключей, а именно, в раундах с 25-го по 32-й подключи используются в прямом порядке, т.е. точно так же, как и в предыдущих раундах алгоритма;
- 20-раундовый алгоритм GOST \dot{A} , в раунде которого для наложения ключа используется операция XOR вместо сложения по модулю 2^{32} .

По результатам анализа сделан вывод о том, что GOST-H и GOST \dot{A} слабее исходного алгоритма ГОСТ 28147-89, поскольку оба имеют классы слабых ключей.

4.3. Анализ полнораундового алгоритма

Существуют атаки и на полнораундовый ГОСТ 28147-89 без каких-либо модификаций. Одна из первых открытых работ, в которых был проведен анализ алгоритма посвящена атакам, использующим слабости процедур расширения ключа ряда известных алгоритмов шифрования. В частности, полнораундовый алгоритм ГОСТ 28147-89 может быть вскрыт с помощью дифференциального криптоанализа на связанных ключах, но только в случае использования слабых таблиц замен. 24-раундовый вариант алгоритма (в котором отсутствуют первые

8 раундов) вскрывается аналогичным образом при любых таблицах замен, однако, сильные таблицы замен делают такую атаку абсолютно непрактичной.

Отечественные ученые А.Г. Ростовцев и Е.Б. Маховенко в 2001 г. предложили принципиально новый метод криптоанализа (по мнению авторов, существенно более эффективный, чем линейный и дифференциальный криптоанализ) путем формирования целевой функции от известного открытого текста, соответствующего ему шифртекста и искомого значения ключа и нахождения ее экстремума, соответствующего истинному значению ключа. Они же нашли большой класс слабых ключей алгоритма ГОСТ 28147-89, которые позволяют вскрыть алгоритм с помощью всего 4-х выбранных открытых текстов и соответствующих им шифртекстов с достаточно низкой сложностью.

В 2004 г. группа специалистов из Кореи предложила атаку, с помощью которой, используя дифференциальный криптоанализ на связанных ключах, можно получить с вероятностью 91,7% 12 бит секретного ключа. Для атаки требуется 2^{35} выбранных открытых текстов и 2^{36} операций шифрования. Как видно, данная атака, практически, бесполезна для реального вскрытия алгоритма.

4.4. Достоинства ГОСТа

- бесперспективность силовой атаки, т.е. полным перебором (XSL-атаки в учёт не берутся, т.к. их эффективность на данный момент полностью не доказана);
- эффективность реализации и соответственно высокое быстродействие на современных компьютерах.
- наличие защиты от навязывания ложных данных (выработка имитовставки) и одинаковый цикл шифрования во всех четырех алгоритмах ГОСТа

4.5. Недостатки ГОСТа

Основные проблемы ГОСТа связаны с неполнотой стандарта в части генерации ключей и таблиц замен. Тривиально доказывается, что у ГОСТа существуют «слабые» ключи и таблицы замен, но в стандарте не описываются

критерии выбора и отсева «слабых». Также стандарт не специфицирует алгоритм генерации таблицы замен (К-блоков). С одной стороны, это может являться дополнительной секретной информацией (помимо ключа), а с другой, поднимает ряд проблем:

- нельзя определить криптостойкость алгоритма, не зная заранее таблицы замен;
- реализации алгоритма от различных производителей могут использовать разные таблицы замен и могут быть несовместимы между собой;
- возможность преднамеренного предоставления слабых таблиц замен лицензирующими органами;
- потенциальная возможность (отсутствие запрета в стандарте) использования таблиц замены, в которых узлы не являются перестановками, что может привести к чрезвычайному снижению стойкости шифра.

4.6. Аппаратная реализация алгоритма

Ранее был создан программный шифровальный комплекс на базе языка программирования C++.

Сейчас стоит задача построения программно аппаратного шифровального комплекса.

Эта работа проходит в два этапа: виртуальная реализация, аппаратная реализация.

Для виртуальной реализации шифровального блока было выбрано приложение Simulink программы Matlab, как наиболее функциональное и простое в обращении.

Модель была построена с низким уровнем глобализации для того, чтобы иметь возможность, получив реальное устройство, иметь возможность, исходя из параметров ПЛИС, получить максимальную скорость шифрования, не прибегая к дополнительным вычислительным мощностям.

В данной работе представлен основной шаг криптопреобразования (приложение В, рисунок 1). При создании на его основе циклической структуры можно получить все режимы шифрования алгоритмом ГОСТ.

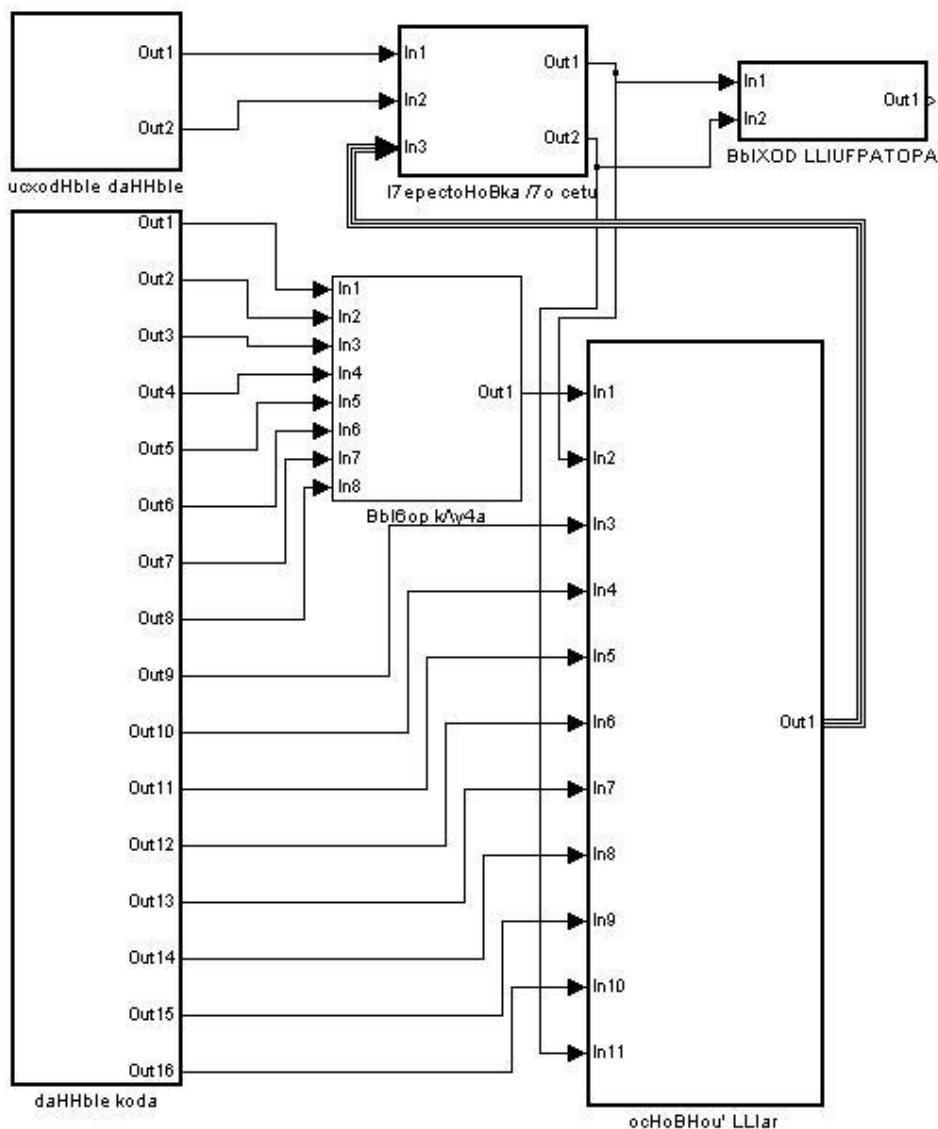


Рис. 3.1. Структурная схема режима простой замены

Краткое пояснение схемы:

Поступающий в виде вектора блок исходного текста и ключ разбиваются на 32-х битные части, инициализируется таблица замен (рисунок 3.2 и 3.3). задание таблицы замен приведено в приложении В (рисунок 2, 3). Младшая часть блока складывается по модулю 2^{32} с младшей частью ключа (приложение В, рисунки 4-7). Результат разбивается на восемь четырехбитовых набора,

определяющих входные индексы таблицы замен. После создания нового 32-х битного числа на основе таблицы замен (приложение В, рисунки 8-11) и поступивших индексов функция производит сдвиг на 11 бит в сторону старших разрядов (приложение В, рисунок 1). Полученный результат и старшая часть блока поступают на сумматор по модулю 2 (приложение В, рисунок 12). Младшая часть обрабатываемого блока признается старшей частью нового блока, а результат на выходе сумматора – его младшей частью. Новый блок уже может использоваться в следующей итерации шифрования.

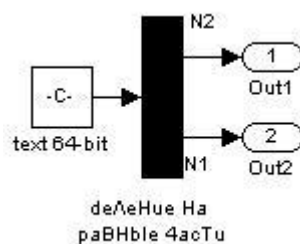


Рис. 3.2. Разбиение блока текста

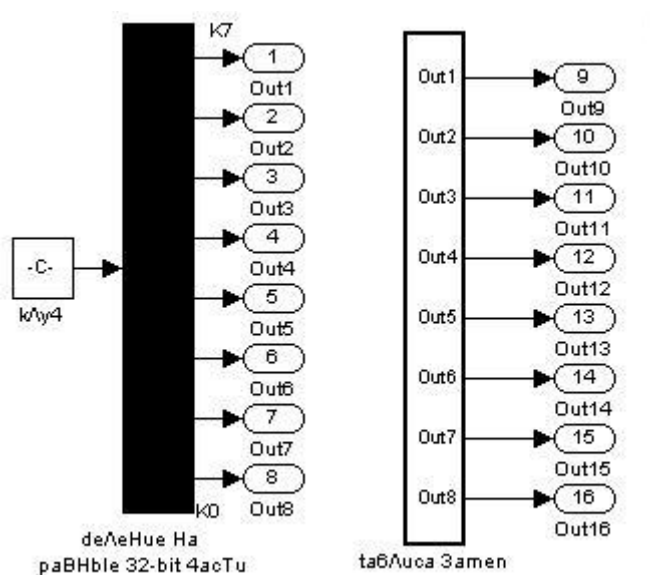


Рис. 3.3. Инициализация таблицы замен и разбиение ключа

Аппаратная реализация будет происходить с использованием ПЛИС. Для этого мы воспользуемся языком программирования QUARTUS. Он позволит выполнить поставленную задачу и реализовать некоторые преимущества, полученные в результате использования детальной проработки модели.

4.4. Методика работы виртуального устройства

1. Определить задачу и требуемые результаты.
2. Выделить дополнительные требования и условия, налагаемые на разработку.
3. Определиться с методом реализации продукта.
4. Продумать возможные пути решения задачи.
5. Определить необходимость использования того или иного пакета разработчика.
6. Обозначить входные и выходные параметры.
7. Составить словесное описание основных функций, которые будет выполнять устройство.
8. Построить структурную схему основных блоков.
9. Определить возможные системы взаимодействия блоков друг с другом.
10. Разработать пошаговую реализацию команд для каждого блока.
11. По необходимости разработать подсистемы внутри основных блоков, для упрощения реализации программы и взаимодействия ее внутренних структур.
12. Провести анализ разработанных блоков и систем, с целью проверки их обоснованности и возможности упрощения структуры будущего продукта.
13. Определиться с элементной базой.
14. Разработать принципиальную схему, используя набор возможностей программного пакета.
15. Произвести рационализацию полученной схемы для облегчения ее понимания другими пользователями и увеличения производительности аппаратуры, на которой будет реализовываться продукт.
16. Произвести построение виртуальной модели продукта с использованием программного пакета и разработанных схем.

17. Проверить правильность построения, соединения и расположения элементов и блоков.
18. При помощи тестирующих сигналов и/или констант проверить работоспособность и правильность выполнения задач продукта.
19. Составить отчет по проделанной работе, указав назначение элементов, осуществляемые блоками функции, и сопроводив его техническими схемами и характеристиками разработанного продукта.

Заключение

В ходе работы подробно изучается российский алгоритм криптографической защиты ГОСТ 28147-89, его слабые и сильные места, способы его программной и аппаратной реализации.

Кроме того, на его основе был создан программный и аппаратный шифровальные продукты, которые в будущем послужат базой для создания программно-аппаратного комплекса для шифрования и дешифровки данных, как на локальном компьютере, так и в сетях передачи информации.

В дальнейшем планируется создание программно-аппаратных шифраторов на базе ПЛИС, не уступающих по своим характеристикам шифровальным комплексам «Криптон».

Так же планируется рассмотрение возможности использования ГОСТа для создания поточного сетевого шифратора.

Приложение А

(Справочное)

Схемы программной реализации алгоритма криптографического преобразования

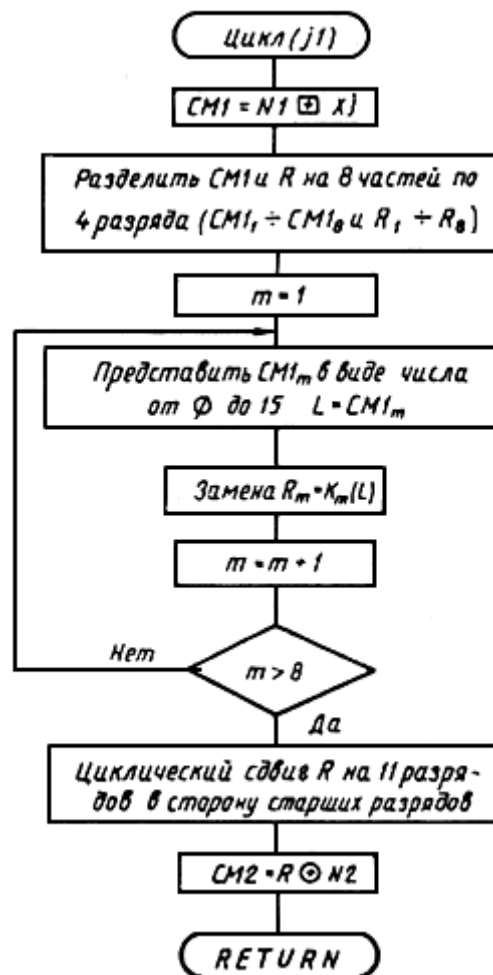


Рис. 1. Схема одного цикла шифрования

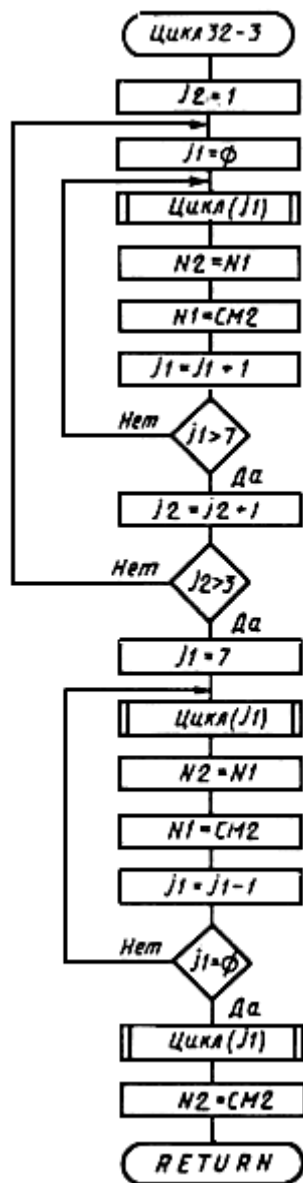


Рис. 2. Схема 32-х циклов зашифрования

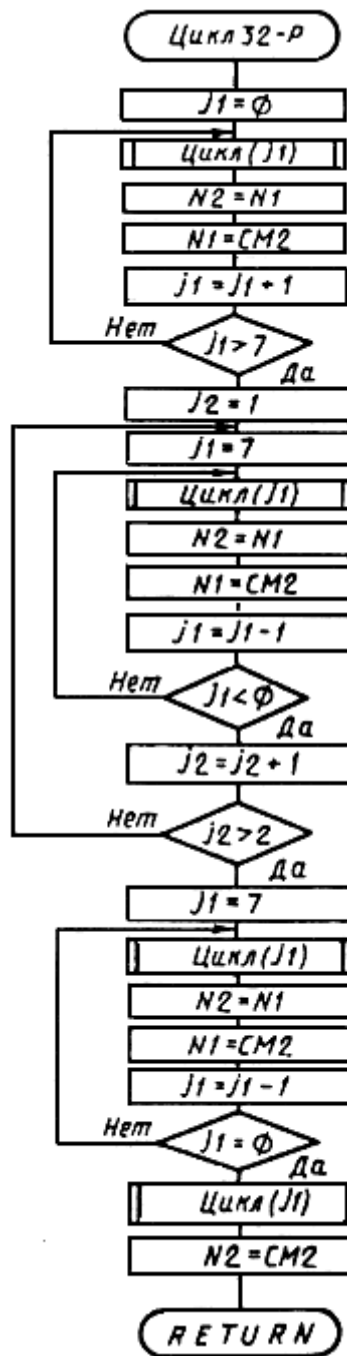


Рис. 3. Схема 32-х циклов расшифрования

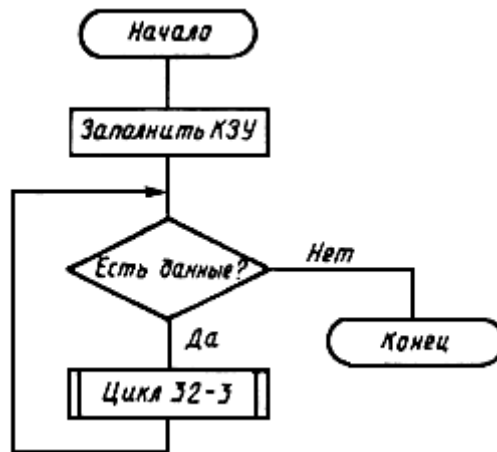


Рис. 4. Схема алгоритма зашифрования в режиме простой замены

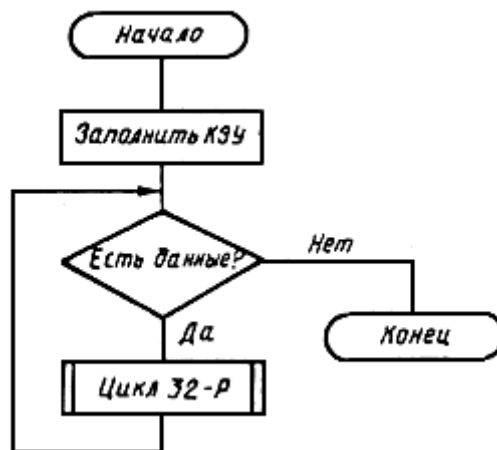


Рис. 5. Схема алгоритма расшифрования в режиме простой замены

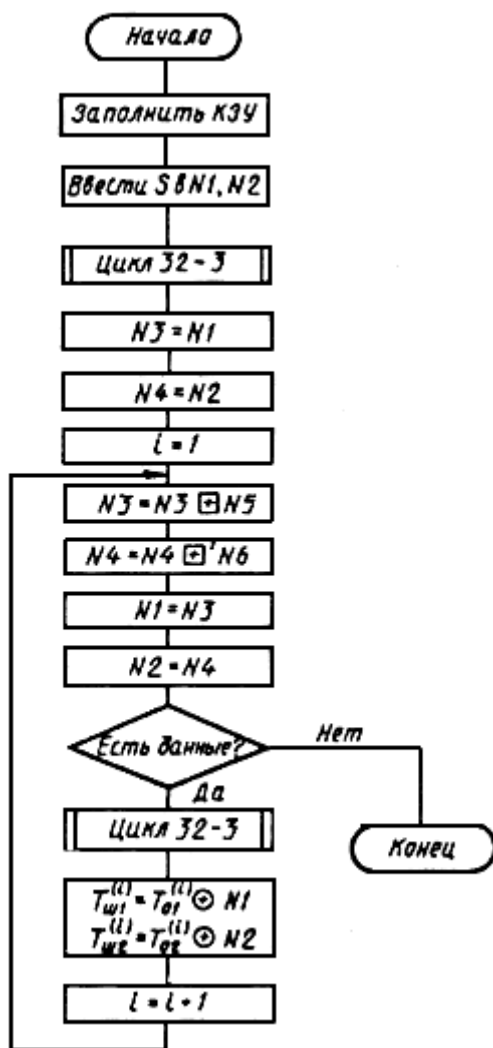


Рис. 6. Схема алгоритма зашифрования в режиме гаммирования

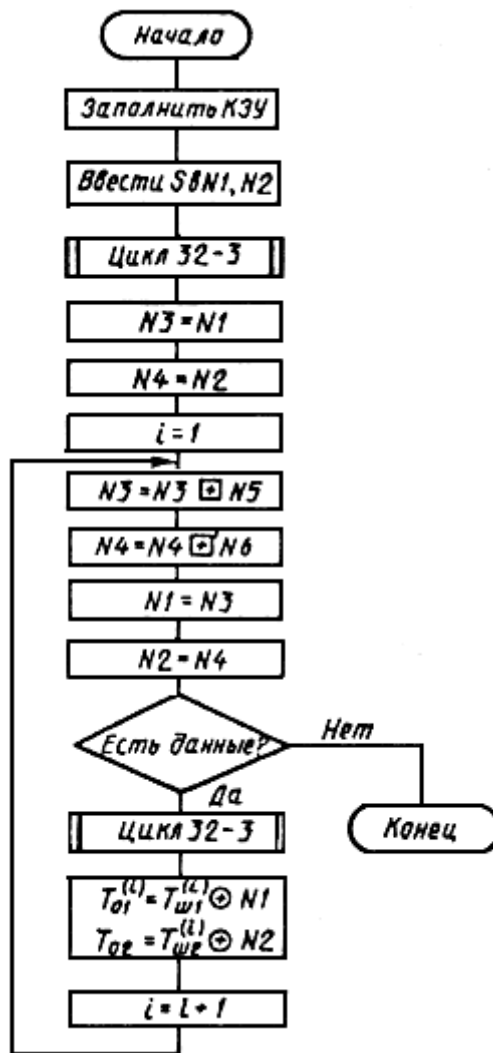


Рис. 7. Схема алгоритма расшифрования в режиме гаммирования

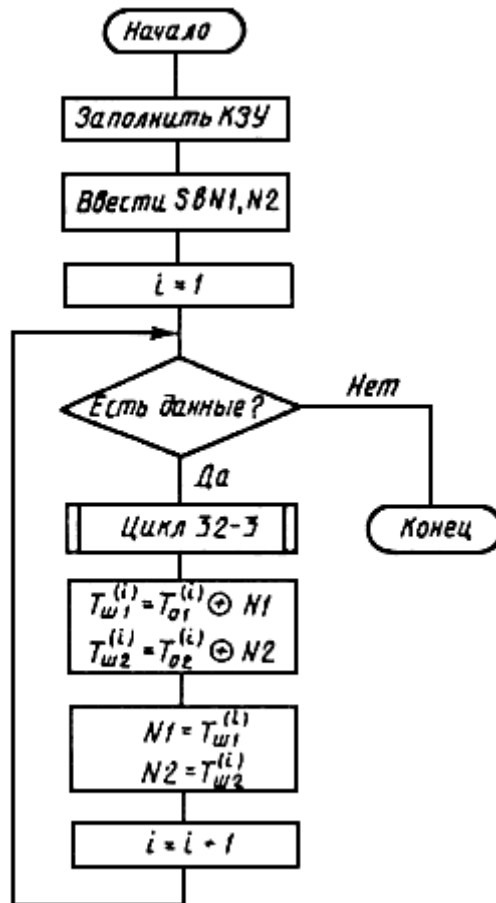


Рис. 8. Схема алгоритма зашифрования в режиме гаммирования с обратной СВЯЗЬЮ

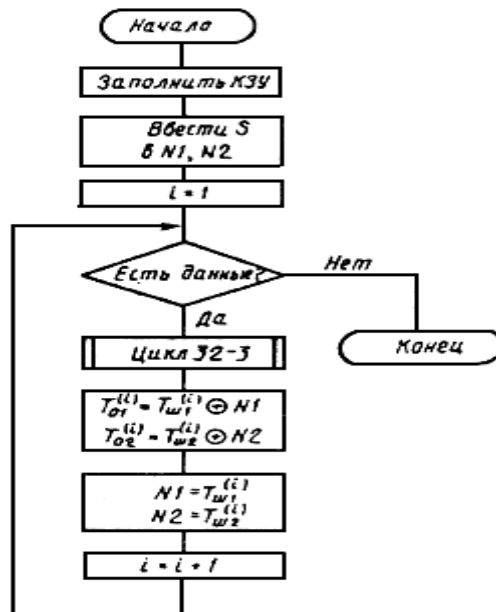


Рис. 9. Схема алгоритма расшифрования в режиме гаммирования с обратной СВЯЗЬЮ

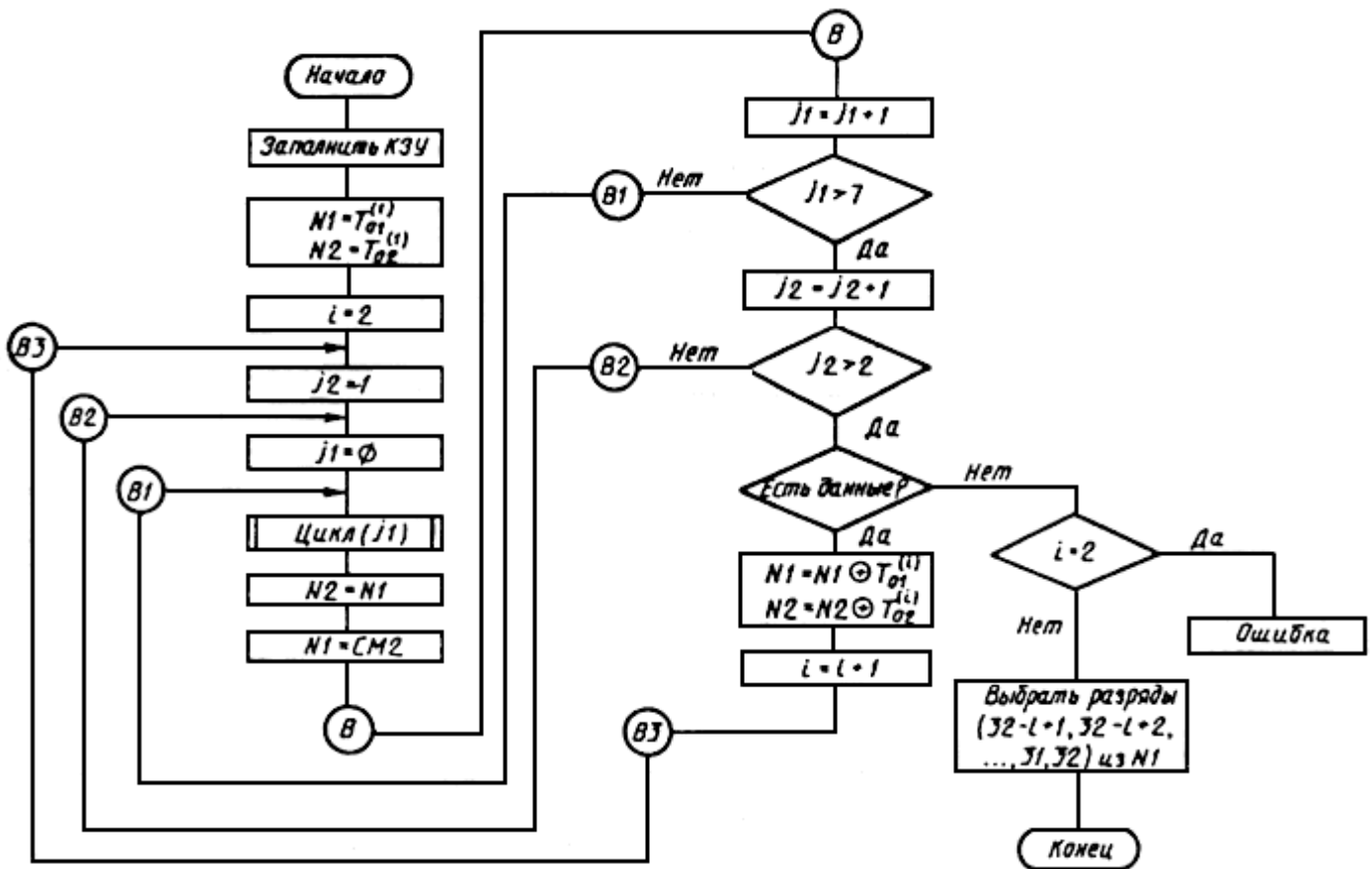


Рисунок 10 – схема алгоритма криптографического преобразования в режиме выработки имитовставок

Приложение Б

(Справочное)

Правила суммирования по модулю 2^{32} и по модулю $(2^{32}-1)$

1. Два целых числа a, b , где $0 \leq a, b < 2^{32}-1$, представленные в двоичном виде

$$a = (a_{32}, a_{31}, \dots, a_2, a_1), \quad b = (b_{32}, b_{31}, \dots, b_2, b_1),$$

$$\text{т.е. } a = a_{32} \cdot 2^{31} + a_{31} \cdot 2^{30} + \dots + a_2 \cdot 2 + a_1,$$

$$b = b_{32} \cdot 2^{31} + b_{31} \cdot 2^{30} + \dots + b_2 \cdot 2 + b_1,$$

суммируются по модулю 2^{32} (операция \boxplus) по следующему правилу:

$$a \boxplus b = a + b, \text{ если } a + b < 2^{32},$$

$$a \boxplus b = a + b - 2^{32}, \text{ если } a + b \geq 2^{32},$$

где операция $+(-)$ есть арифметическая сумма (разность) двух целых чисел.

2. Два целых числа a, b , где $0 \leq a, b \leq 2^{32}-1$, представленные в двоичном виде

$$a = (a_{32}, a_{31}, \dots, a_2, a_1),$$

$$b = (b_{32}, b_{31}, \dots, b_2, b_1),$$

суммируются по модулю $(2^{32}-1)$ (операция \boxplus') по следующему правилу:

$$a \boxplus' b = a + b, \text{ если } a + b < 2^{32},$$

$$a \boxplus' b = a + b - 2^{32} + 1, \text{ если } a + b \geq 2^{32}.$$

Приложение В (Справочное)

Структурные схемы построения устройства

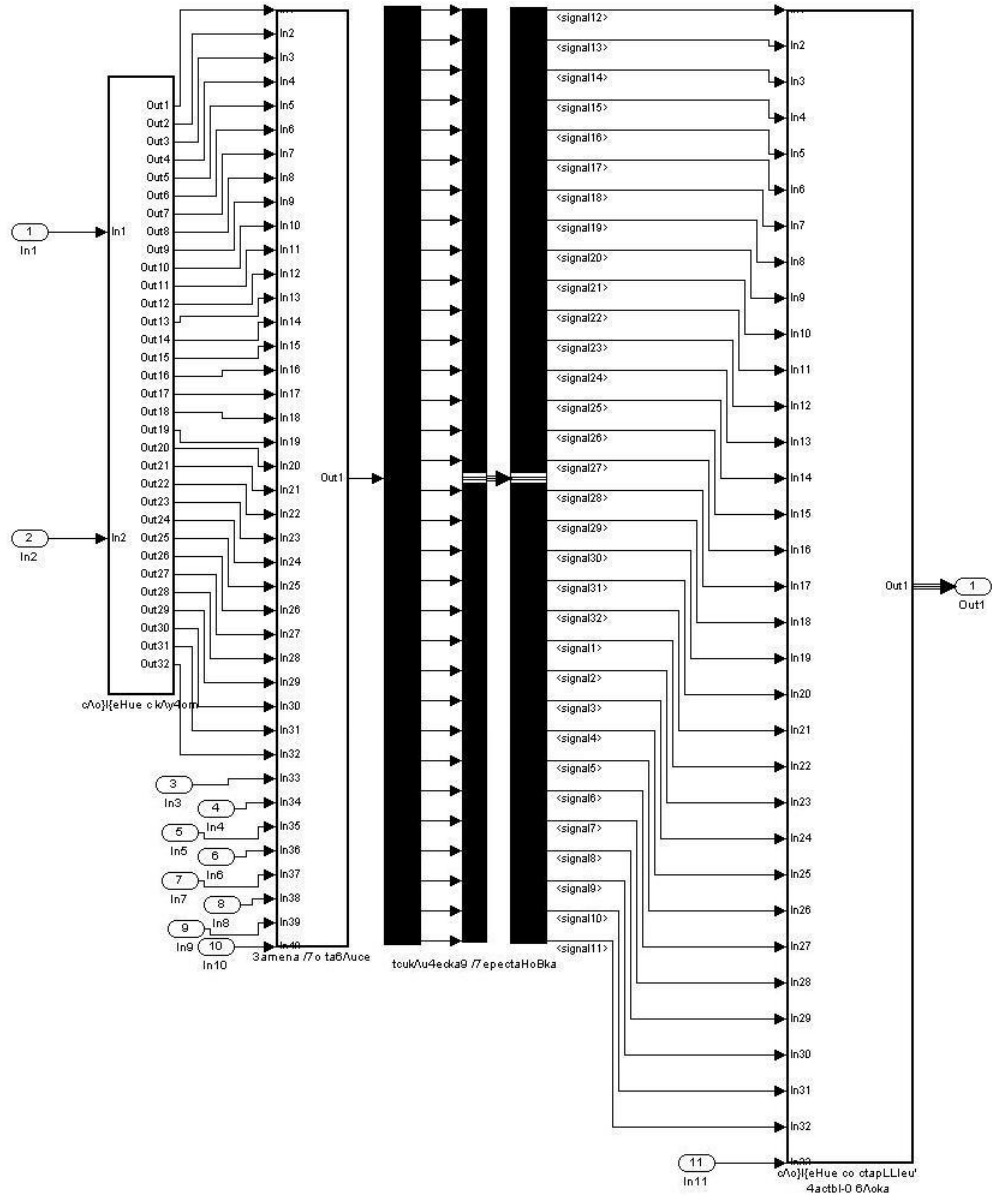


Рис. 1. Структурная схема основного шага криптопреобразования

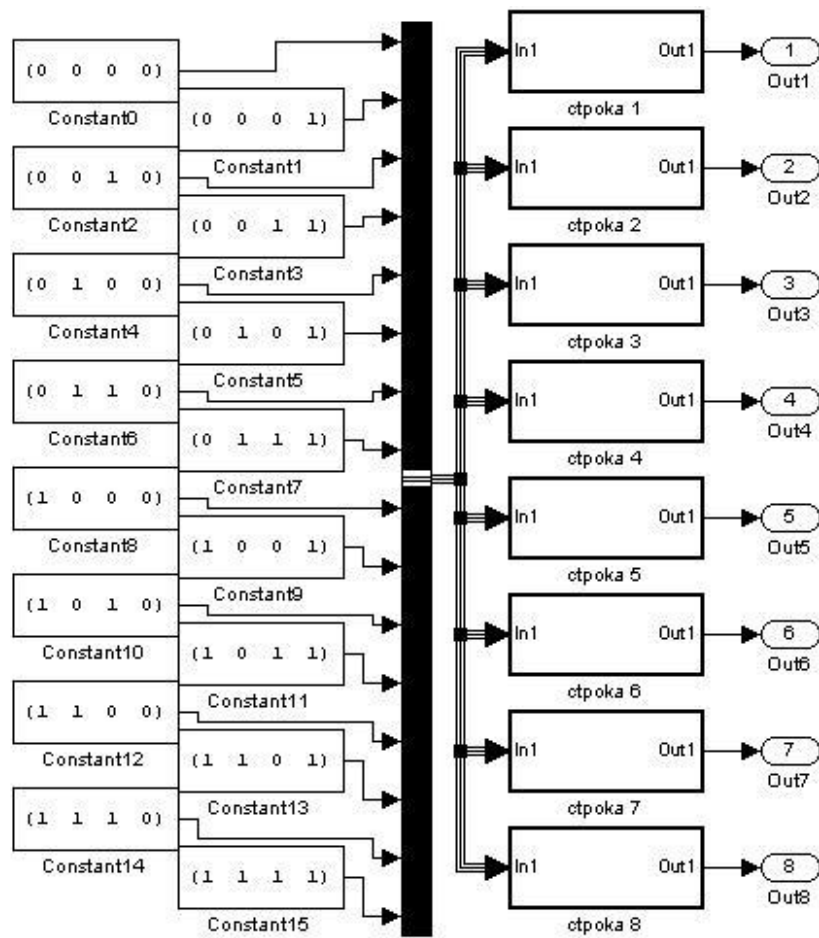


Рис. 2. Задание таблицы замен

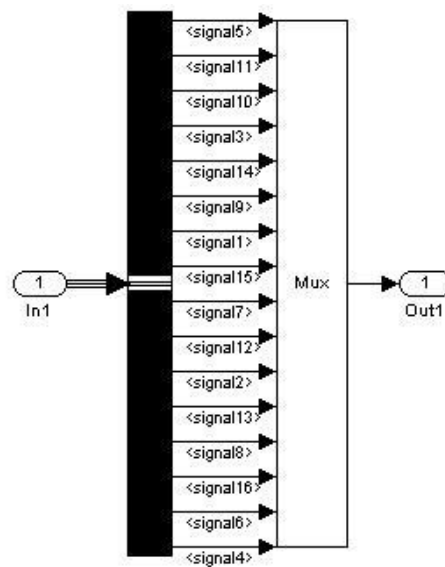


Рис. 3. Задание строк таблицы замен

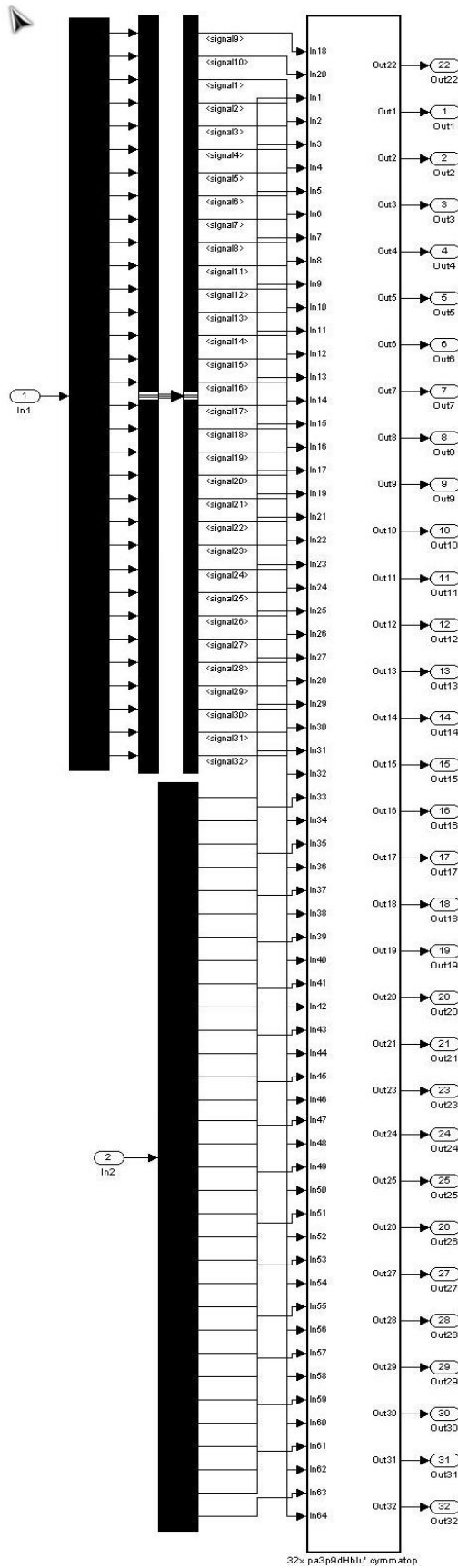


Рис. 4. Сложение части блока текста с ключом

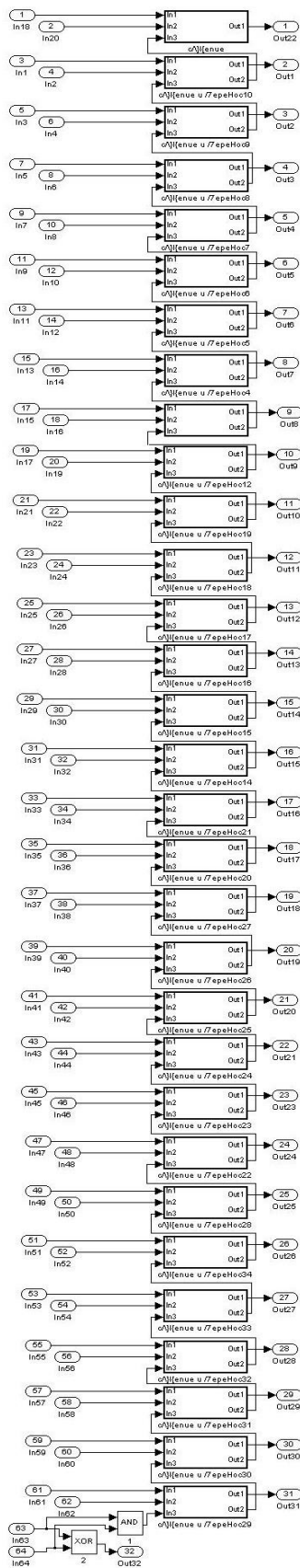


Рис. 5. Структурная схема 32-х разрядного сумматора

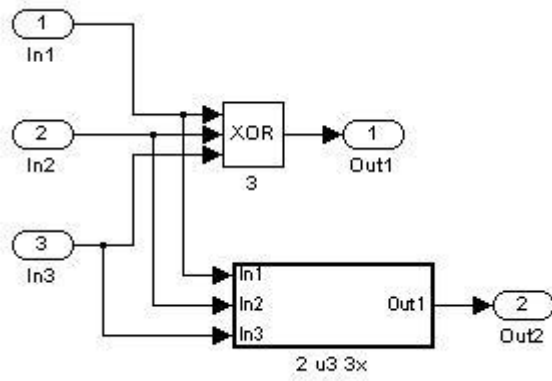


Рис. 6. Схема сложения и переноса в старший разряд

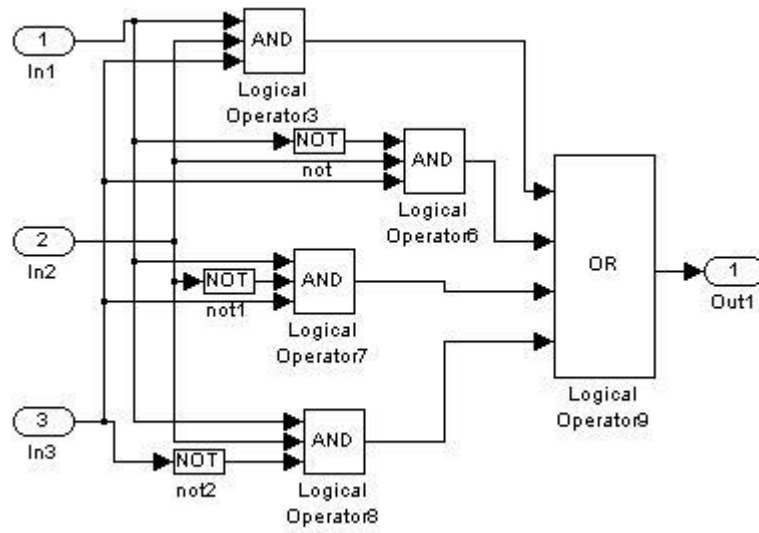


Рис. 7. Выбор необходимости переноса по принципу «2 из 3»

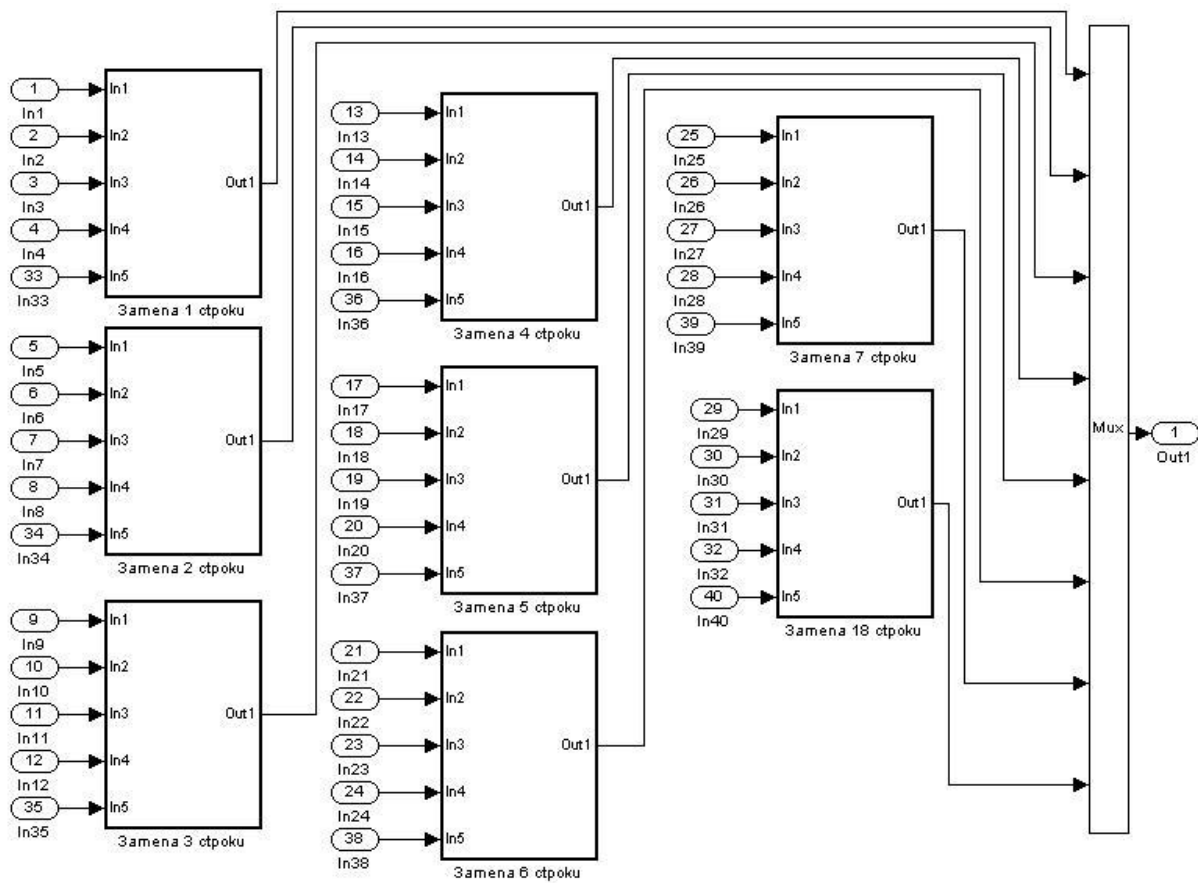


Рис. 8. Структурная схема замены по таблице

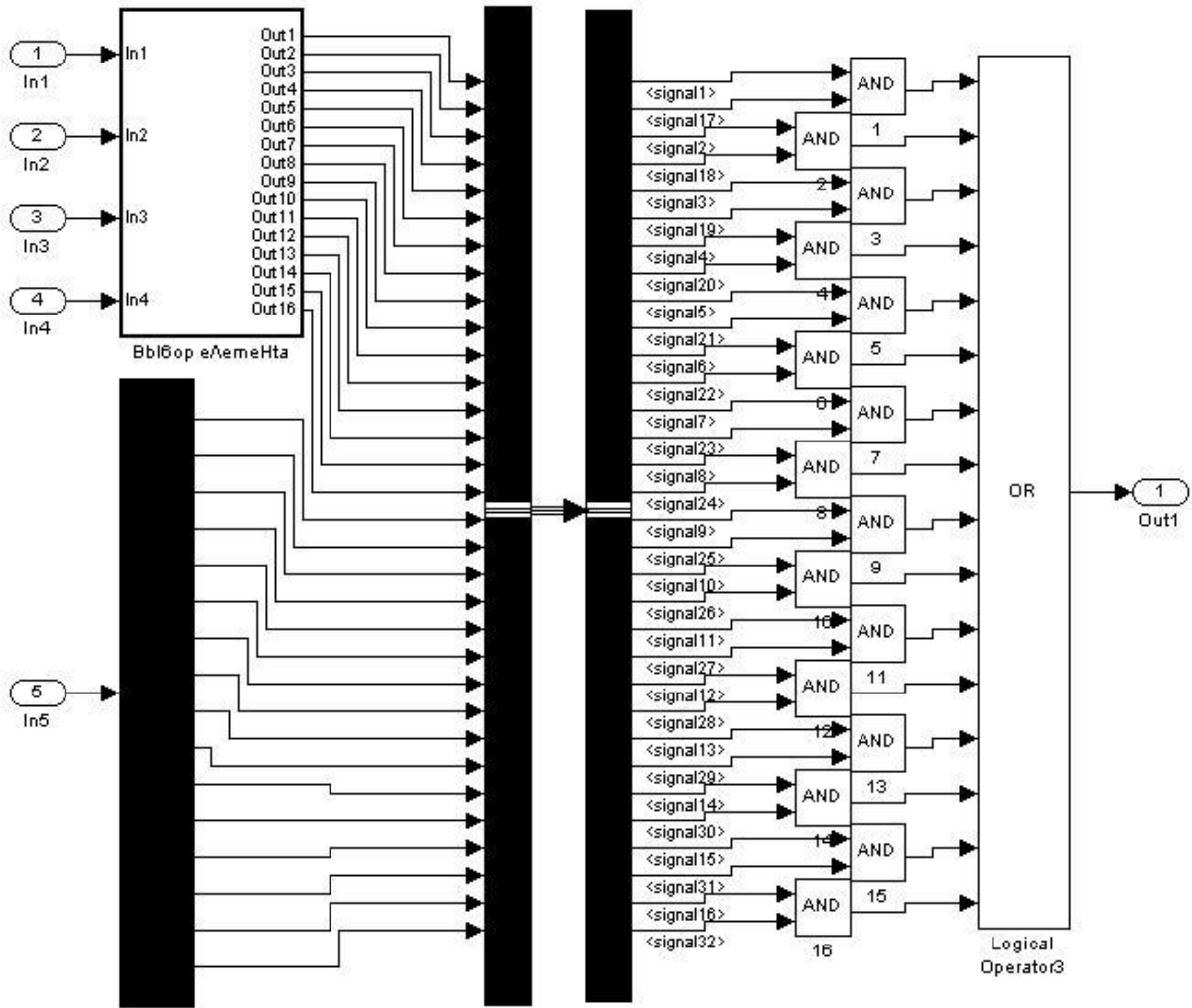


Рис. 9. Схема замены по одной строке таблицы

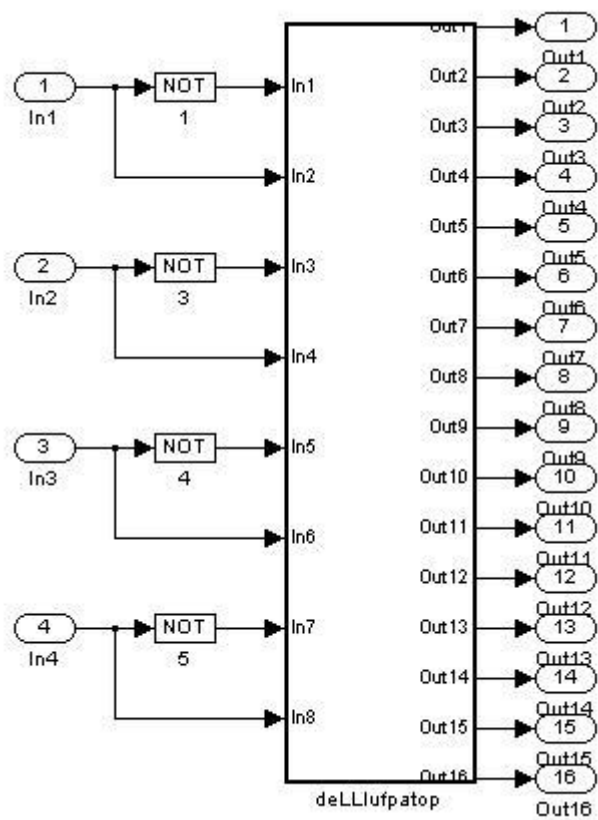


Рис. 10. Определение элемента, который будет заменяться

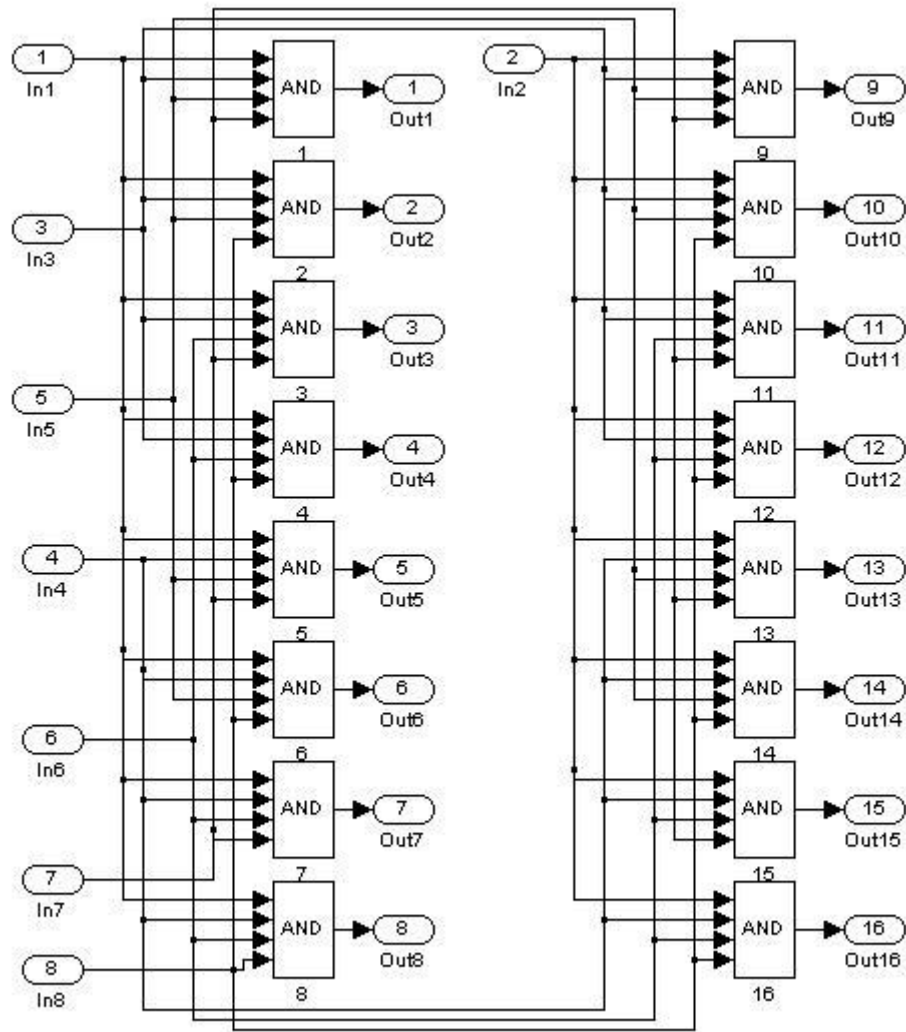


Рис. 11. Схема дешифратора для 4-х битового кода

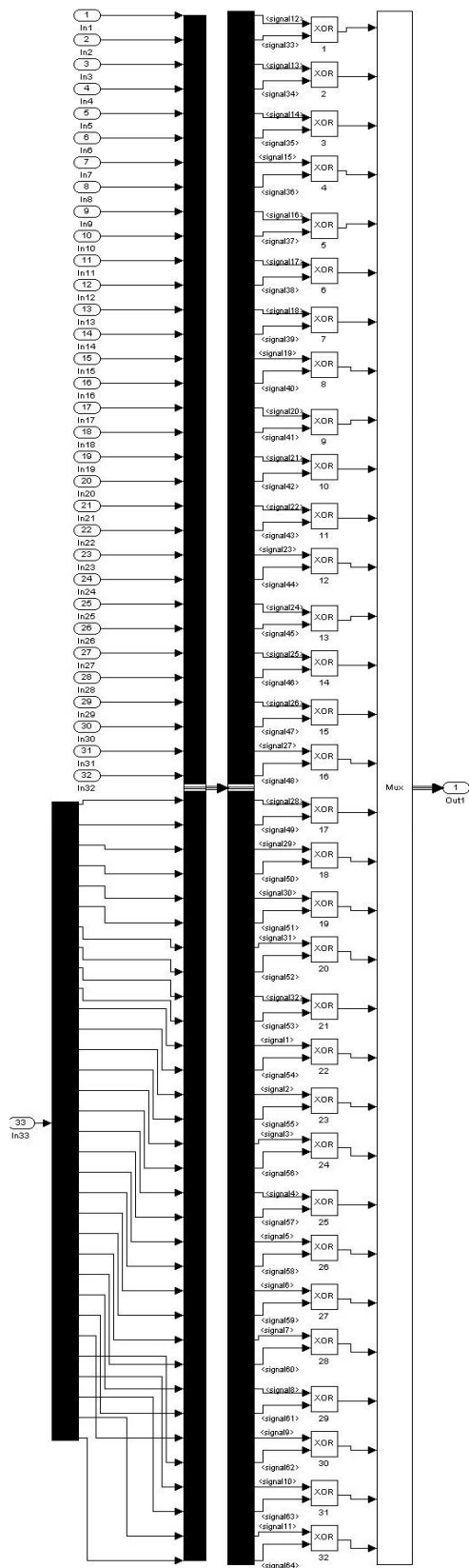


Рис. 12. Схема сложения предыдущих результатов со старшей частью текста

4. Рекомендуемая литература

1. ГОСТ 28147-89 / Группа П85 / ГОСУДАРСТВЕННЫЙ СТАНДАРТ СОЮЗА ССР / СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ. ЗАЩИТА КРИПТОГРАФИЧЕСКАЯ / Алгоритм криптографического преобразования / ОКП 40 4000 / Дата введения 1990-07-01
2. Винокуров А. Алгоритм шифрования ГОСТ 28147-89, его использование и программная реализация для компьютеров платформы Intel x86
3. Панасенко С. П. Стандарт шифрования ГОСТ 28147-89 - 15 августа 2007 года

Лабораторная работа 4. Исследование международных алгоритмов шифрования информации AES (Advanced Encryption Standard)

1. Цель работы

Изучить криптографический стандарт шифрования AES и его особенности, познакомиться с различными режимами блочного шифрования и единственной атакой «Квадрат».

2. Краткие теоретические сведения

В 1997 г. Национальный институт стандартов и технологий США (NIST) объявил о начале программы по принятию нового стандарта криптографической защиты - стандарта XXI в. для закрытия важной информации правительственного уровня на замену существующему с 1974 г. алгоритму DES, самому распространенному криптоалгоритму в мире.

Требования к кандидатам были следующие:

- криптоалгоритм должен быть открыто опубликован;
- криптоалгоритм должен быть симметричным блочным шифром, допускающим размеры ключей в 128, 192 и 256 бит;
- криптоалгоритм должен быть предназначен как для аппаратной, так и для программной реализации;
- криптоалгоритм должен быть доступен для открытого использования в любых продуктах, а значит, не может быть запатентован, в противном случае патентные права должны быть аннулированы;
- криптоалгоритм подвергается изучению по следующим параметрам: стойкости, стоимости, гибкости, реализуемости в smart-картах.

В финал конкурса вышли следующие алгоритмы: MARS, TWOFISH и RC6 (США), RUNDAEL (Бельгия), SERPENT (Великобритания, Израиль, Норвегия). По своей структуре TWOFISH является классическим шифром Фейстеля; MARS и RC6 можно отнести к модифицированным шифрам Фейстеля, в них используется новая малоизученная операция циклического "прокручивания" битов слова на число позиций, изменяющихся в зависимости от шифруемых данных и секретного ключа; RUNDAEL и SERPENT являются классическими SP-сетями. MARS и TWOFISH имеют самую сложную конструкцию, RUNDAEL и RC6 - самую простую.

В октябре 2000 г. конкурс завершился. Победителем был признан бельгийский шифр RIJNDAEL, как имеющий наилучшее сочетание стойкости, производительности, эффективности реализации и гибкости. Его низкие требования к объему памяти делают его идеально подходящим для встроенных систем. Авторами шифра являются Йон Дэмен (Joan Daemen) и Винсент Рюмен (Vincent Rijmen), начальные буквы фамилий которых и образуют название алгоритма - RIJNDAEL.

После этого NIST начал подготовку предварительной версии Федерального Стандарта Обработки Информации (Federal Information Processing Standart - FIPS) и в феврале 2001 г. опубликовал его на сайте <http://csrc.nist.gov/encryption/aes/>. В течение 90-дневного периода открытого обсуждения предварительная версия FIPS пересматривалась с учетом комментариев, после чего начался процесс исправлений и утверждения. Наконец 26 ноября 2001 г. была опубликована окончательная версия стандарта FIPS-197, описывающего новый американский стандарт шифрования AES. Согласно этому документу стандарт вступил в силу с 26 мая 2002 г.

2.2. Блочный криптоалгоритм RIJNDAEL и стандарт AES

Высочайшую надежность AES NIST подтверждает астрономическими числами. 128битный ключ обеспечивает 340 андециллионов ($340 \cdot 10^{36}$) возможных комбинаций, а 256битный ключ увеличивает это число до $11 \cdot 10^{76}$. Для сравнения, старый алгоритм DES, дает общее число комбинаций в $72 \cdot 10^{15}$. На их перебор у специально построенной машины "DES Cracker" уходит несколько часов. Но даже если бы она делала это всего за одну секунду, то на перебор 128битного ключа машина потратила бы 149 триллионов лет. Между тем, возраст всей Вселенной ученые оценивают в менее, чем 20 миллиардов лет.

2.2.1. Математические предпосылки

Алгоритм оперирует байтами, которые рассматриваются как элементы конечного поля $GF(2^8)$.

Элементами поля $GF(2^8)$ являются многочлены степени не более 7, которые могут быть заданы строкой своих коэффициентов. Если представить байт в виде

$$b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0,$$

то элемент поля описывается многочленом с коэффициентами из $\{0, 1\}$:

$$b_7 x^7 + b_6 x^6 + b_5 x^5 + b_4 x^4 + b_3 x^3 + b_2 x^2 + b_1 x^1 + b_0$$

Например, байту $\{11001011\}$ (или $\{cb\}$ в шестнадцатеричной форме) соответствует многочлен $x^7 + x^6 + x^3 + x + 1$.

Для элементов конечного поля определены аддитивные и мультипликативные операции.

Сложение

Сложение суть операция поразрядного XOR и поэтому обозначается как \oplus . Пример выполнения операции сложения:

$$(x^6 + x^4 + x^2 + x + 1) \oplus (x^7 + x + 1) = x^1 + x^6 + x^4 + x^2 \text{ (в виде многочленов)}$$

$$\{01010111\} \oplus \{10000011\} = \{11010100\} \text{ (двоичное представление)}$$

$$\{57\} \oplus \{83\} = \{d4\} \text{ (шестнадцатеричное представление)}$$

В конечном поле для любого ненулевого элемента **a** существует обратный элемент **-a**, при этом $a + (-a) = 0$, где нулевой элемент -это $\{00\}$. В $GF(2^8)$ справедливо $a + a = 0$, т. е. каждый ненулевой элемент является своей собственной аддитивной инверсией.

Умножение

Умножение, обозначаемое далее как \bullet , более сложная операция. Умножение в $GF(2^8)$ - это операция умножения многочленов со взятием результата по модулю неприводимого многочлена $m(x)$ восьмой степени и с использованием операции XOR при приведении подобных членов. В RIJNDAEL выбран $m(x) = x^8 + x^4 + x^3 + x + 1$, или в шестнадцатеричной форме $1\{1b\}$ (такая запись обозначает, что присутствует «лишний» девятый бит). Пример операции умножения:

$$\{57\} \cdot \{83\} = \{c1\},$$

или

$$\begin{aligned} &(x^6 + x^4 + x^2 + x + 1) (x^7 + x + 1) = \\ &= x^{13} + x^{11} + x^9 + x^8 + x^7 + x^7 + x^5 + x^3 + x^2 + x + x^6 + x^4 + x^2 + x + 1 = \\ &= x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1 \end{aligned}$$

Следовательно,

$$x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1 \bmod (x^8 + x^4 + x^3 + x + 1) = x^7 + x^6 + 1$$

Ясно, что результат является двоичным полиномом не выше 8 степени. В отличие от сложения, простой операции умножения на уровне байтов не существует.

Умножение, определенное выше, является ассоциативным, и существует единичный элемент ('01'). Для любого двоичного полинома $b(x)$ не выше 8-й степени можно использовать расширенный алгоритм Евклида для вычисления полиномов $a(x)$ и $c(x)$ таких, что

$$b(x) a(x) + m(x) c(x) = 1$$

Следовательно,

$$a(x) * b(x) \bmod m(x) = 1$$

или

$$b^{-1}(x) = a(x) \bmod m(x)$$

Более того, можно показать, что

$$a(x) * (b(x) + c(x)) = a(x) * b(x) + a(x) * c(x)$$

Из всего этого следует, что множество из 256 возможных значений байта образует конечное поле $GF(2^8)$ с XOR в качестве сложения и умножением, определенным выше.

Умножение на x

Если умножить $b(x)$ на полином x , мы будем иметь:

$$b_7 x^8 + b_6 x^7 + b_5 x^6 + b_4 x^5 + b_3 x^4 + b_2 x^3 + b_1 x^2 + b_0 x$$

$x * b(x)$ получается понижением предыдущего результата по модулю $m(x)$. Если $b_7 = 0$, то данное понижение является тождественной операцией. Если $b_7 = 1$, $m(x)$ следует вычесть (т.е. XORed). Из этого следует, что умножение на x может быть реализовано на уровне байта как левый сдвиг и последующий побитовый XOR с '1B'. Данная операция обозначается как $b = xtime(a)$.

Полиномы с коэффициентами из GF (2⁸)

Полиномы могут быть определены с коэффициентами из GF (2⁸). В этом случае четырехбайтный вектор соответствует полиному степени 4.

Полиномы могут быть сложены простым сложением соответствующих коэффициентов. Как сложение в GF (2⁸) является побитовым XOR, так и сложение двух векторов является простым побитовым XOR.

Умножение представляет собой более сложное действие. Предположим, что мы имеем два полинома в GF (2⁸).

$$a(x) = a_3 x^3 + a_2 x^2 + a_1 x + a_0$$

$$b(x) = b_3 x^3 + b_2 x^2 + b_1 x + b_0$$

$$c(x) = a(x) b(x)$$

определяется следующим образом

$$c(x) = c_6 x^6 + c_5 x^5 + c_4 x^4 + c_3 x^3 + c_2 x^2 + c_1 x + c_0$$

$$c_0 = a_0 * b_0$$

$$c_1 = a_1 * b_0 \oplus a_0 * b_1$$

$$c_2 = a_2 * b_0 \oplus a_1 * b_1 \oplus a_0 * b_2$$

$$c_3 = a_3 * b_0 \oplus a_2 * b_1 \oplus a_1 * b_2 \oplus a_0 * b_3$$

$$c_4 = a_3 * b_1 \oplus a_2 * b_2 \oplus a_1 * b_3$$

$$c_5 = a_3 * b_2 \oplus a_2 * b_3$$

$$c_6 = a_3 * b_3$$

Ясно, что в таком виде $c(x)$ не может быть представлен четырехбайтным вектором. Понижая $c(x)$ по модулю полинома 4-й степени, результат может быть полиномом степени ниже 4. В Rijndael это сделано с помощью полинома

$$M(x) = x^4 + 1$$

так как

$$x^j \bmod (x^4 + 1) = x^j \bmod 4$$

Модуль, получаемый из $a(x)$ и $b(x)$, обозначаемый $d(x) = a(x) \otimes b(x)$, получается следующим образом:

$$d_0 = a_0 * b_0 \oplus a_3 * b_1 \oplus a_2 * b_2 \oplus a_1 * b_3$$

$$d_1 = a_1 * b_0 \oplus a_0 * b_1 \oplus a_3 * b_2 \oplus a_2 * b_3$$

$$d_2 = a_2 * b_0 \oplus a_1 * b_1 \oplus a_0 * b_2 \oplus a_3 * b_3$$

$$d_3 = a_3 * b_0 \oplus a_2 * b_1 \oplus a_1 * b_2 \oplus a_0 * b_3$$

Операция, состоящая из умножения фиксированного полинома $a(x)$, может быть записана как умножение матрицы, где матрица является циклической. Мы имеем

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

2.2.2. Описание криптоалгоритма

Формат блоков данных и число раундов

RIJNDAEL - это итерационный блочный шифр, имеющий архитектуру "Квадрат". Шифр имеет переменную длину блоков и различные длины ключей.

Длина ключа и длина блока могут быть равны независимо друг от друга 128, 192 или 256 битам. В стандарте AES определена длина блока данных, равная 128 битам.

Введем следующие обозначения:

- N_b - число 32-битных слов содержащихся во входном блоке, $N_b=4$;
- N_k - число 32-битных слов содержащихся в ключе шифрования, $N_k=4,6,8$;
- N_r - число раундов шифрования, как функция от N_b и N_k , $N_r=10,12,14$.

Входные (input), промежуточные (state) и выходные (output) результаты преобразований, выполняемых в рамках криптоалгоритма, называются состояниями (State). Состояние можно представить в виде прямоугольного массива байтов (рис. 1). При размере блока, равном 128 битам, этот 16-байтовый массив (рис. 2, а) имеет 4 строки и 4 столбца (каждая строка и каждый столбец в этом случае могут рассматриваться как 32-разрядные слова). Входные данные для шифра обозначаются как байты состояния в порядке $S_{00}, S_{10}, S_{20}, S_{30}, S_{01}, S_{11}, S_{21}, S_{31}$

После завершения действия шифра выходные данные получаются из байтов состояния в том же порядке. В общем случае число столбцов N_b равно длине блока, деленной на 32.

State

a_0	a_4	a_8	a_{12}
a_1	a_5	a_9	a_{13}
a_2	a_6	a_{10}	a_{14}
a_3	a_7	a_{11}	a_{15}

Рис. 1. Пример представления 128-разрядного блока данных в виде массива State, где a , - байт блока данных, а каждый столбец - одно 32-разрядное слово

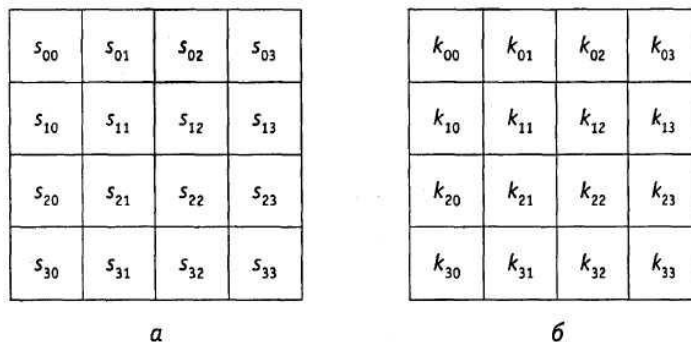


Рис. 2. Форматы данных: a - пример представления блока ($Nb, = 4$); b - ключа шифрования ($Nk = 4$), где s и k - соответственно байты массива State и ключа, находящиеся на пересечении i -й строки и j -го столбца

Рисунок 3 демонстрирует такое представление, носящее название архитектуры «Квадрат».

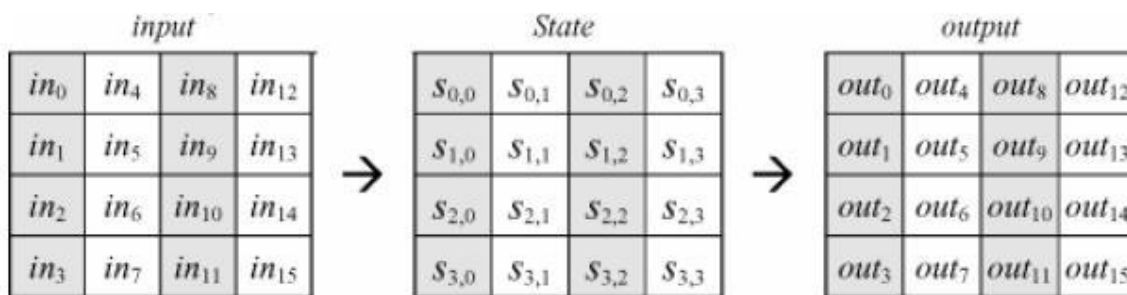


Рис. 3. Пример представления блока в виде матрицы $4Nb$

Ключ шифрования также представлен в виде прямоугольного массива с четырьмя строками (рис. 2, б). Число столбцов этого массива равно длине ключа, деленной на 32. В стандарте определены ключи всех трех размеров - 128 бит, 192 бита и 256 бит, то есть соответственно 4, 6 и 8 32-разрядных слова (или столбца – в табличной форме представления). В некоторых случаях ключ

шифрования рассматривается как линейный массив 4-байтовых слов. Слова состоят из 4 байтов, которые находятся в одном столбце (при представлении в виде прямоугольного массива).

Число раундов N_r в алгоритме RIJNDAEL зависит от значений N_b и N_k , как показано в табл. 1. В стандарте AES определено соответствие между размером ключа, размером блока данных и числом раундов шифрования, как показано в табл. 2.

Таблица 1. Число раундов N_r как функция от длины ключа N_k и длины блока N_b

N_r	$N_b = 4$	$N_b = 6$	$N_b = 8$
$N_k = 4$	10	12	14
$N_k = 6$	12	12	14
$N_k = 8$	14	14	14

Таблица 2. Соответствие между длиной ключа, размером блока данных и числом раундов в стандарте AES

Стандарт	Длина ключа (N_k слов)	Размер блока данных (N_b слов)	Число раундов (N_r)
AES-128	10	12	14
AES-192	12	12	14
AES-256	14	14	14

Раундовое преобразование

Раунд состоит из четырех различных преобразований:

- замены байтов SubBytes() - побайтовой подстановки в S-блоках с фиксированной таблицей замен размерностью 8 x 256;
- сдвига строк ShiftRows() - побайтового сдвига строк массива State на различное количество байт;

- перемешивания столбцов MixColumns() — умножения столбцов состояния, рассматриваемых как многочлены над $GF(2^8)$, на многочлен третьей степени $g(x)$ по модулю $x^4 + 1$;
- сложения с раундовым ключом AddRoundKey() - поразрядного XOR с текущим фрагментом развернутого ключа.

Замена байтов (SubBytes). Преобразование SubBytes() представляет собой нелинейную замену байтов, выполняемую независимо с каждым байтом состояния. Таблицы замены S-блока являются инвертируемыми и построены из композиции следующих двух преобразований входного байта:

1. получение обратного элемента относительно умножения в поле $GF(2^8)$, нулевой элемент $\{00\}$ переходит сам в себя;
2. применение преобразования над $GF(2^8)$, определенного следующим образом:

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

Другими словами суть преобразования может быть описана уравнениями:

$$b'_i = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i$$

где $c_0 = c_1 = c_5 = c_6 = 0$, $c_2 = c_3 = c_4 = c_7 = 1$, b'_i и b_i - соответственно преобразованное и исходное значение i -го бита, $i = 0, 1, 2, \dots, 7$.

Применение описанного S-блока ко всем байтам состояния обозначается как SubBytes(State). Рис. 4. иллюстрирует применение преобразования SubBytes() к состоянию. Логика работы S-блока при преобразовании байта {ху} отражена в табл. 3. Например, результат {ed} преобразования байта {53} находится на пересечении 5-й строки и 3-го столбца.

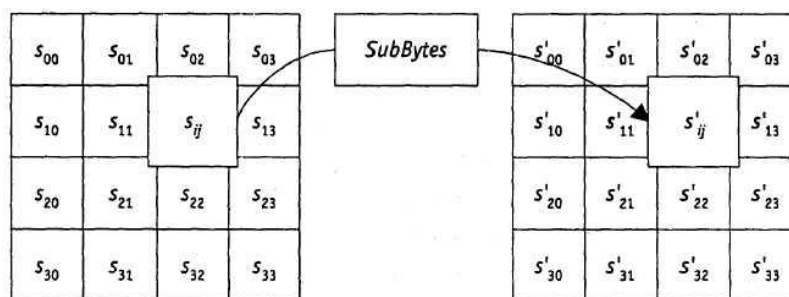


Рис. 4. SubBytes() действует на каждый байт состояния

Таблица 3. Таблица замен S-блока

x	y															
	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Преобразование сдвига строк (ShiftRows). Последние 3 строки состояния циклически сдвигаются влево на различное число байтов. Строка 1 сдвигается

на C1 байт, строка 2 - на C2 байт, и строка 3 - на C3 байт. Значения сдвигов C1, C2 и C3 в RIJNDAEL зависят от длины блока N_b . Их величины приведены в табл.4.

Таблица 4. Величина сдвига для разной длины блоков

b	N	C1	C2	C3
	4	1	2	3
	6	1	2	3
	8	1	3	3

В стандарте AES, где определен единственный размер блока, равный 128 битам, C1 = 1, C2 = 2 и C3 = 3.

Операция сдвига последних трех строк состояния обозначена как ShiftRows(State). Рис. 5 показывает влияние преобразования на состояние.

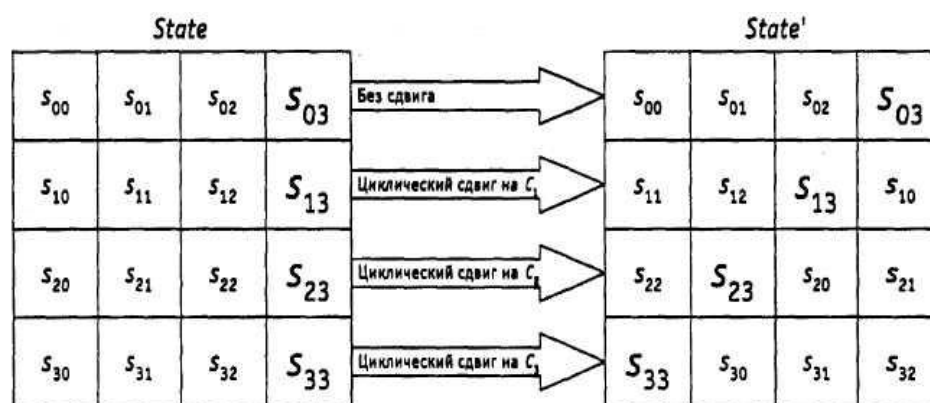


Рис. 5. ShiftRows() действует на строки состояния

Преобразование перемешивания столбцов (MixColumns). В этом преобразовании столбцы состояния рассматриваются как многочлены над $GF(2^8)$ и умножаются по модулю $x^4 + 1$ на многочлен $g(x)$, выглядящий следующим образом:

$$g(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}.$$

Это может быть представлено в матричном виде следующим образом:

$$\begin{bmatrix} s'_{0c} \\ s'_{1c} \\ s'_{2c} \\ s'_{3c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0c} \\ s_{1c} \\ s_{2c} \\ s_{3c} \end{bmatrix}, \quad 0 \leq c \leq 3,$$

где c - номер столбца массива State

В результате такого умножения байты столбца S_{0c} , S_{1c} , S_{2c} , S_{3c} заменяются соответственно на байты

$$S'_{0c} = (\{02\} \cdot S_{0c}) \oplus (\{03\} \cdot S_{1c}) \oplus S_{2c} \oplus S_{3c},$$

$$S'_{1c} = S_{0c} \oplus (\{02\} \cdot S_{1c}) \oplus (\{03\} \cdot S_{2c}) \oplus S_{3c},$$

$$S'_{2c} = S_{0c} \oplus S_{1c} \oplus (\{02\} \cdot S_{2c}) \oplus (\{03\} \cdot S_{3c}),$$

$$S'_{3c} = (\{03\} \cdot S_{0c}) \oplus S_{1c} \oplus S_{2c} \oplus (\{02\} \cdot S_{3c}).$$

Применение этой операции ко всем четырём столбцам состояния обозначено как $\text{MixColumns}(\text{State})$. Рис. 6 демонстрирует применение преобразования MixColumnsQ к столбцу состояния.

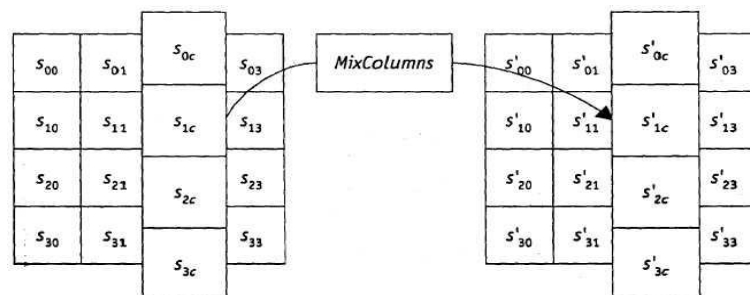


Рис. 6. MixColumnsQ действует на столбцы состояния

Добавление раундового ключа (AddRoundKey). В данной операции раундовый ключ добавляется к состоянию посредством простого поразрядного XOR. Раундовый ключ вырабатывается из ключа шифрования посредством алгоритма выработки ключей (key schedule). Длина раундового ключа (в 32-разрядных словах) равна длине блока Nb. Преобразование, содержащее добавление посредством XOR раундового ключа к состоянию (рис. 7), обозначено как $\text{AddRoundKey}(\text{State}, \text{RoundKey})$.

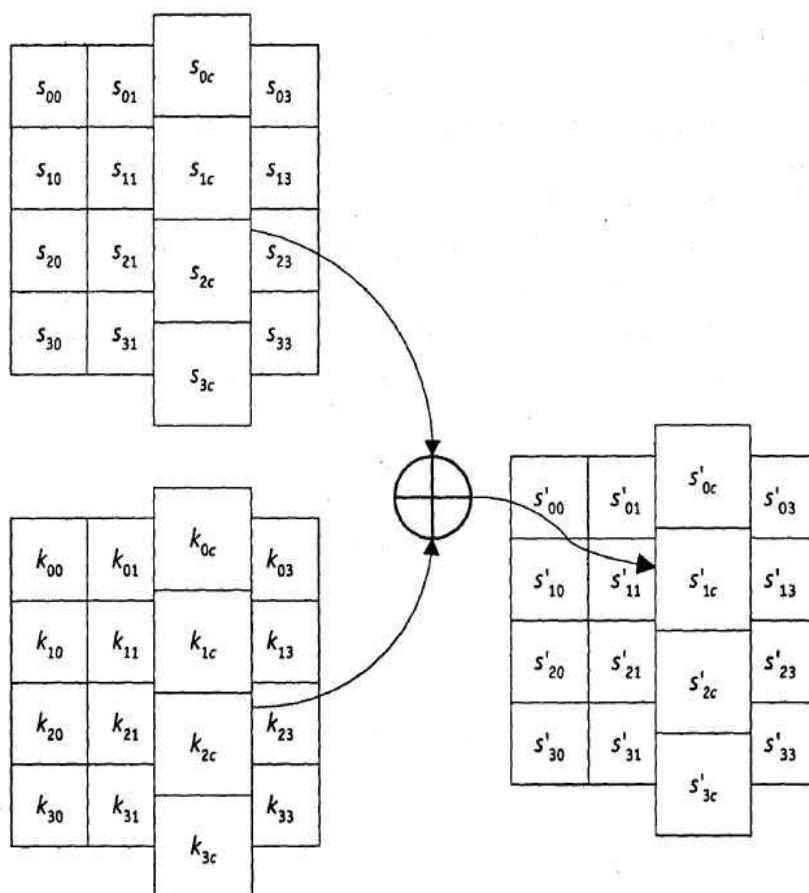


Рис. 7. При добавлении ключа раундовый ключ складывается посредством операции XOR с состоянием

Алгоритм выработки ключей (Key Schedule)

Раундовые ключи получаются из ключа шифрования посредством алгоритма выработки ключей. Он содержит два компонента: расширение ключа

(Key Expansion) и выбор раундового ключа (Round Key Selection). Основополагающие принципы алгоритма выглядят следующим образом:

- общее число битов раундовых ключей равно длине блока, умноженной на число раундов, плюс 1 (например, для длины блока 128 бит и 10 раундов требуется 1408 бит раундовых ключей);
- ключ шифрования расширяется в расширенный ключ (Expanded Key);
- раундовые ключи берутся из расширенного ключа следующим образом: первый раундовый ключ содержит первые Nb слов, второй - следующие Nb слов и т. д.

Расширение ключа (Key Expansion). Расширенный ключ в RIJNDAEL представляет собой линейный массив $w[i]$ из $Nb(N_r + 1)$ 4-байтовых слов, $i = 0, 1 \dots Nb(N_r + 1)$. В AES массив $w[i]$ состоит из $4(N_r + 1)$ 4-байтовых слов, $i = 0, 1 \dots 4(N_r + 1)$.

Примечание. Необходимо учитывать, что с целью полноты описания здесь приводится алгоритм для всех возможных длин ключей, на практике же полная его реализация нужна не всегда.

Первые N_k слов содержат ключ шифрования. Все остальные слова определяются рекурсивно из слов с меньшими индексами. Алгоритм выработки ключей зависит от величины N_k .

Как можно заметить (рис. 8, а), первые N_k слов заполняются ключом шифрования. Каждое последующее слово $w[i]$ получается посредством XOR предыдущего слова $w[i-1]$ и слова на N_k позиций ранее $w[i - N_k]$.

$$w[i] = w[i - 1] \oplus w[i - N_k].$$

Для слов, позиция которых кратна N_k , перед XOR применяется преобразование к $w[i-1]$, а затем еще прибавляется раундовая константа $Rcon$. Преобразование реализуется с помощью двух дополнительных функций: $RotWord()$, осуществляющей побайтовый сдвиг 32-разрядного слова по формуле $\{a_0 a_1 a_2 a_3\} \rightarrow \{a_1 a_2 a_3 a_0\}$, и $SubWord()$ осуществляющей побайтовую замену с использованием 5-блока функции $SubBytes()$. Значение $Rcon[j]$ равно 2^{j-1} . Значение $w[i]$ в этом случае равно

$$w[i] = SubWord(RotWord(w[i - 1])) \oplus Rcon[i/N_k] \oplus w[i - N_k].$$



Рис. 8. Процедуры

а - расширения ключа (светло-серым цветом выделены слова расширенного ключа, которые формируются без использования функций $SubWord()$ и $RotWordQ$; темно-серым цветом выделены слова расширенного ключа, при вычислении которых используются преобразования $SubWordQ$ и $RotWordQ$);

б - выбора раундового ключа для $N_k - 4$

Выбор раундового ключа (Round Key Selection). Раундовый ключ i получается из слов массива раундового ключа от $W[N_b i]$ и до $W[N_b (i + 1)]$, как показано на рис. 8.

Примечание. Алгоритм выработки ключей можно осуществлять и без использования массива $w[i]$. Для реализаций, в которых существенно требование к занимаемой памяти, раундовые ключи могут вычисляться "на лету" посредством использования буфера из N_k слов. Расширенный ключ должен всегда получаться из ключа шифрования и никогда не указывается напрямую. Нет никаких ограничений на выбор ключа шифрования.

Функция зашифрования

Шифр RIJNDAEL состоит (рис. 9):

- из начального добавления раундового ключа;
- $N_r - 1$ раундов;
- заключительного раунда, в котором отсутствует операция `MixColumns()`.

На вход алгоритма подаются блоки данных $State$, в ходе преобразований содержимое блока изменяется и на выходе образуется шифротекст, организованный опять же в виде блоков $State$, как показано на рис. 26, где $N_b = 4$, in_m и out_m - m -е байты соответственно входного и выходного блоков, $m = 0, 1, \dots, 15$, s_{ij} - байт, находящийся на пересечении i -й строки и j -го столбца массива $State$, $i = j = 0, 1, \dots, 3$.

Перед началом первого раунда происходит суммирование по модулю 2 с начальным ключом шифрования, затем - преобразование массива байтов $State$ в течение 10, 12 или 14 раундов в зависимости от длины ключа. Последний раунд несколько отличается от предыдущих тем, что не задействует функцию перемешивания байт в столбцах `MixColumns()`.

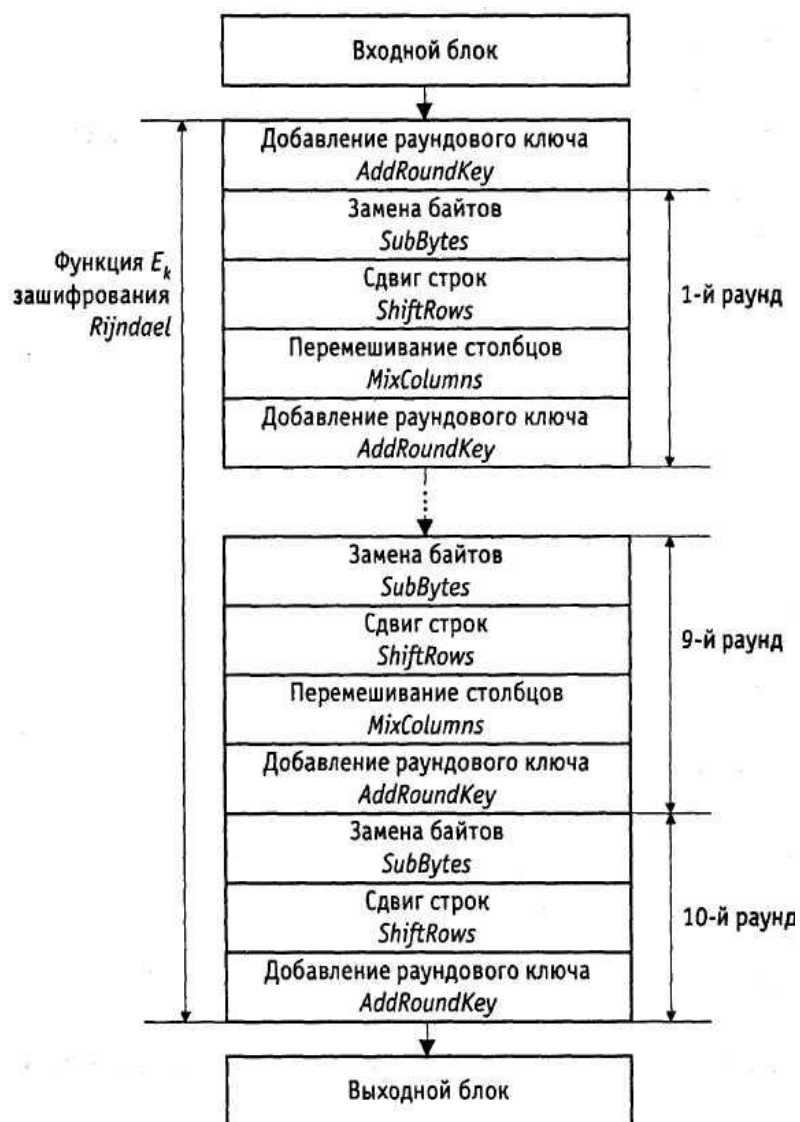


Рис. 9 Схема функции E_k зашифрования криптоалгоритма RIJNDAEL при $N_k = N_b = 4$

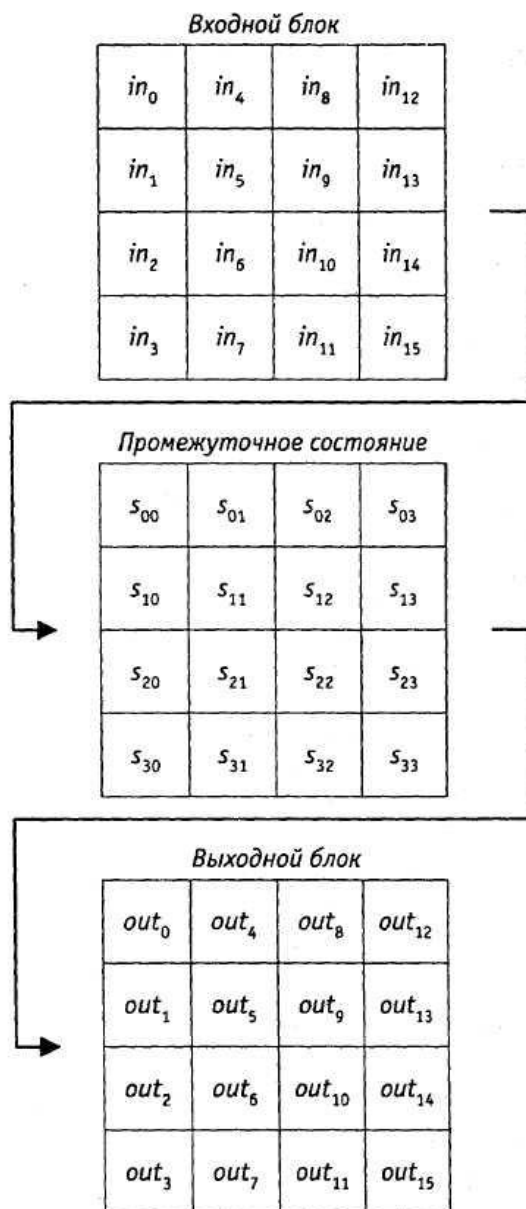


Рис. 10. Ход преобразования данных, организованных в виде блоков State

Рис. 11 демонстрирует и рассеивающие и перемешивающие свойства шифра. Видно, что два раунда обеспечивают полное рассеивание и перемешивание.

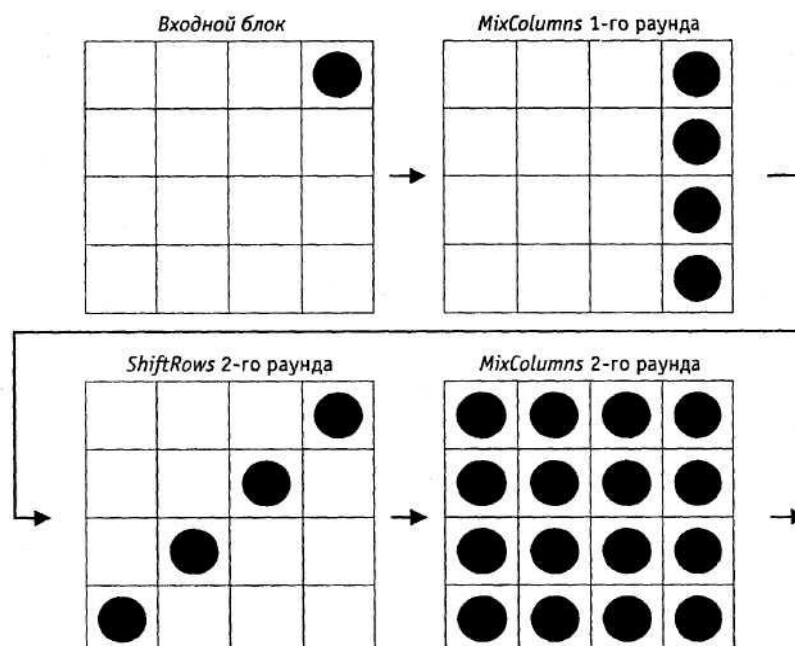


Рис. 11. Принцип действия криптоалгоритма RIJNDAEL; \bullet - измененный байт

Функция обратного расшифрования

Если вместо SubBytes(), ShiftRows(), MixColumns() и AddRoundKey() в обратной последовательности выполнить инверсные им преобразования, можно построить функцию обратного расшифрования. При этом порядок использования раундовых ключей является обратным по отношению к тому, который используется при зашифровании.

Далее приводится описание функций, обратных используемым при прямом зашифровании.

Функция AddRoundKey() обратна сама себе, учитывая свойства используемой в ней операции XOR.

Преобразование InvSubBytes. Логика работы инверсного S-блока при преобразовании байта {ху} отражена в табл. 5.

Таблица 5. Таблица замен инверсного S-блока

x	y															
	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Преобразование *InvShiftRows*. Последние 3 строки состояния циклически сдвигаются вправо на различное число байтов. Строка 1 сдвигается на C1 байт, строка 2 - на C2 байт, и строка 3 - на C3 байт. Значения сдвигов C1, C2, C3 зависят от длины блока Nb,. Их величины приведены в табл. 8.

Преобразование *InvMixColumns*. В этом преобразовании столбцы состояния рассматриваются как многочлены над $GF(2^8)$ и умножаются по модулю $x^4 + 1$ на многочлен $g^{-1}(x)$, выглядящий следующим образом:

$$g^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\}.$$

Это может быть представлено в матричном виде следующим образом:

$$\begin{bmatrix} s'_{0c} \\ s'_{1c} \\ s'_{2c} \\ s'_{3c} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} s_{0c} \\ s_{1c} \\ s_{2c} \\ s_{3c} \end{bmatrix}, 0 \leq c \leq 3.$$

В результате на выходе получаются следующие байты

$$\begin{aligned} s'_{0c} &= (\{0e\} \bullet s_{0c}) \oplus (\{0b\} \bullet s_{1c}) \oplus (\{0d\} \bullet s_{2c}) \oplus (\{09\} \bullet s_{3c}), \\ s'_{1c} &= (\{09\} \bullet s_{0c}) \oplus (\{0e\} \bullet s_{1c}) \oplus (\{0b\} \bullet s_{2c}) \oplus (\{0d\} \bullet s_{3c}), \\ s'_{2c} &= (\{0d\} \bullet s_{0c}) \oplus (\{09\} \bullet s_{1c}) \oplus (\{0e\} \bullet s_{2c}) \oplus (\{0b\} \bullet s_{3c}), \\ s'_{3c} &= (\{0b\} \bullet s_{0c}) \oplus (\{0d\} \bullet s_{1c}) \oplus (\{09\} \bullet s_{2c}) \oplus (\{0e\} \bullet s_{3c}). \end{aligned}$$

Функция прямого расшифрования

Алгоритм обратного расшифрования, описанный выше, имеет порядок приложения операций-функций, обратный порядку операций в алгоритме прямого шифрования, но использует те же параметры (развернутый ключ). Однако некоторые свойства алгоритма шифрования RIJNDAEL позволяют применить для расшифрования тот же порядок приложения функций (обратных используемым для зашифрования) за счет изменения некоторых параметров, а именно - развернутого ключа.

Два следующих свойства позволяют применить алгоритм прямого расшифрования.

Порядок приложения функций SubBytes() и ShiftRows() не играет роли. То же самое верно и для операций InvSubBytes() и InvShiftRows(). Это происходит потому, что функции SubBytes() и InvSubBytes() работают с байтами, а операции ShiftRows() и InvShiftRows() сдвигают целые байты, не затрагивая их значений.

Операция `MixColumns()` является линейной относительно входных данных, что означает

$$\begin{aligned} \text{InvMixColumns}(\text{State XOR RoundKey}) &= \\ &= \text{InvMixColumns}(\text{State}) \text{ XOR } \text{InvMixColumns}(\text{RoundKey}) \end{aligned}$$

Эти свойства функций алгоритма шифрования позволяют изменить порядок применения функций `InvSubBytes()` и `InvShiftRows()`. Функции `AddRoundKey()` и `InvMixColumns()` также могут быть применены в обратном порядке, но при условии, что столбцы (32-битные слова) развернутого ключа расшифрования предварительно пропущены через функцию `invMixColumns()`.

Таким образом, можно реализовать более эффективный способ расшифрования с тем же порядком приложения функций как и в алгоритме зашифрования.

При формировании развернутого ключа шифрования в процедуру развертывания ключа необходимо добавить следующий код

Примечание. В последнем операторе (в функции `InvMixColumn()`) происходит преобразование типа, так как развернутый ключ хранится в виде линейного массива 32-разрядных слов, в то время как входной параметр функции - двумерный массив байтов.

В табл. 6 приведена процедура зашифрования, а также два эквивалентных варианта процедуры расшифрования при использовании двухраундового варианта Rijndael. Первый вариант функции расшифрования суть обычная инверсия функции зашифрования. Второй вариант функции зашифрования получен из первого после изменения порядка следования операций в трех парах преобразований:

invShiftRows — InvSubBytes (дважды)

и AddRoundKey — InvMixColumns.

Таблица 6. Последовательность преобразований в двухраундовом варианте RIJNDAEI

Функция зашифрования двухраундового варианта RIJNDAEI	Функция обратного расшифрования двухраундового варианта RIJNDAEI	Эквивалентная функция прямого расшифрования двухраундового варианта RIJNDAEI
AddRoundKey	AddRoundKey	AddRoundKey
SubBytes	InvShiftRows	InvSubBytes
ShiftRows	InvSubBytes	InvShiftRows
MixColumns	AddRoundKey	InvMixColumns
AddRoundKey	InvMixColumns	AddRoundKey
SubBytes	InvShiftRows	InvSubBytes
ShiftRows	InvSubBytes	InvShiftRows
AddRoundKey	AddRoundKey	AddRoundKey

Очевидно, что результат преобразования при переходе от исходной к обратной последовательности выполнения операций в указанных парах не изменится.

Видно, что процедура зашифрования и второй вариант процедуры расшифрования совпадают с точностью до порядка использования раундовых ключей (при выполнении операций AddRoundKey), таблиц замен (при выполнении операций SubBytes и InvSubBytes) и матриц преобразования (при выполнении операций MixColumns и invMixColumns). Данный результат легко обобщить и на любое другое число раундов.

2.3. Атака “Квадрат”

Атака "Квадрат" была специально разработана для одноименного шифра SQUARE (авторы J. Daemen, L. Knudsen, V. Rijmen). Атака использует при своем проведении байт-ориентированную структуру шифра. Учитывая, что RIJNDAEL унаследовал многие свойства шифра SQUARE, эта атака применима и к нему. Далее приведено описание атаки "Квадрат" применительно к RIJNDAEL.

Атака "Квадрат" основана на возможности свободного подбора атакующим некоторого набора открытых текстов для последующего их зашифрования. Она независима от таблиц замен блоков, многочлена функции MixColumns() и способа разворачивания ключа. Эта атака для 6-раундового шифра RIJNDAEL, состоящего из 6 раундов, эффективнее, чем полный перебор по всему ключевому пространству. После описания базовой атаки на 4-раундовый ROUNDAEL, будет показано, как эту атаку можно продлить на 5 и даже 6 раундов. Но уже для 7 раундов “Квадрат” становится менее эффективным, чем полный перебор.

Предпосылки

Пусть L-набор - такой набор из 256 входных блоков (массивов State), каждый из которых имеет байты (назовем их активными), значения которых различны для всех 256 блоков. Остальные байты (будем называть их пассивными) остаются одинаковыми для всех 256 блоков из L-набора. То есть:

$$\forall x, y \in L \begin{cases} x_{ij} \neq y_{ij} , \text{ если байт с номером } ij \text{ активный;} \\ x_{ij} = y_{ij} , \text{ в противном случае.} \end{cases}$$

Будучи подвергнутыми обработке функциями SubBytes() и AddRoundKey() блоки L-набора дадут в результате другой L-набор с активными байтами в тех же позициях, что и у исходного. Функция ShiftRows() сместит эти байты соответственно заданным в ней смещениям в строках массивов State.

После функции MixColumns() L-набор в общем случае необязательно останется L-набором (т. е. результат преобразования может перестать удовлетворять определению L-набора). Но поскольку каждый байт результата функции MixColumns() является линейной комбинацией (с обратимыми коэффициентами) четырех входных байт того же столбца:

$$b_{ij} = 2a_{ij} \oplus 3a_{(i+1)j} \oplus a_{(i+2)j} \oplus a_{(i+3)j}$$

столбец с единственным активным байтом на входе даст в результате на выходе столбец со всеми четырьмя байтами - активными.

Базовая атака “Квадрат” на 4 раунда

Рассмотрим L-набор, во всех блоках которого активен только один байт. Иначе говоря, значение этого байта различно во всех 256 блоках, а остальные байты одинаковы (скажем, равны нулю). Проследим эволюцию этого байта на протяжении трех раундов. В первом раунде функция MixColumns() преобразует один активный байт в столбец из 4 активных байт. Во втором раунде эти 4 байта разойдутся по 4 различным столбцам в результате преобразования функцией ShiftRows(). Функция же MixColumns() следующего, третьего раунда преобразует эти байты в 4 столбца, содержащие активные байты. Этот набор все еще остается L-набором до того момента, когда он поступает на вход функции MixColumns() третьего раунда.

Основное свойство L-набора, используемое здесь, то, что поразрядная сумма по модулю 2 всех блоков такого набора всегда равна нулю. Действительно, поразрядная сумма неактивных (с одинаковыми значениями) байт равна нулю по определению операции поразрядного XOR, а активные байты, пробегаая все 256 значений, также при поразрядном суммировании дадут нуль. Рассмотрим теперь результат преобразования функцией MixColumns() в третьем раунде байтов входного массива данных a в байты выходного массива данных b . Покажем, что и в этом случае поразрядная сумма всех блоков выходного набора будет равна нулю, то есть:

$$\begin{aligned} \bigoplus_{b=\text{mixcolumns}(a), a \in L} b_{ij} &= \bigoplus_{a \in L} (2a_{ij} \oplus 3a_{(i+1)j} \oplus a_{(i+2)j} \oplus a_{(i+3)j}) = \\ &= 2 \bigoplus_{a \in L} a_{ij} \oplus 3 \bigoplus_{a \in L} a_{(i+1)j} \oplus \bigoplus_{a \in L} a_{(i+2)j} \oplus \bigoplus_{a \in L} a_{(i+3)j} = 2 \cdot 0 \oplus 3 \cdot 0 \oplus 1 \cdot 0 \oplus 1 \cdot 0 = 0 \end{aligned}$$

Таким образом, вес данные на входе четвертого раунда сбалансированы (то есть их полная сумма равна нулю). Этот баланс в общем случае нарушается последующим преобразованием данных функцией `SubBytes()`.

Мы предполагаем далее, что четвертый раунд является последним, то есть в нем нет функции `MixColumns()`. Тогда каждый байт выходных данных этого раунда зависит только от одного байта входных данных. Если обозначить через a байт выходных данных четвертого раунда, через b байт входных данных и через k - соответствующий байт раундового ключа, то можно записать:

$$a_{ij} = \text{SubBytes}(b_{ij}) \oplus k_{ij}$$

Отсюда, предполагая значение k_{ij} можно по известному a_{ij} вычислить b_{ij} , а затем проверить правильность догадки о значении k_{ij} ; если значения байта b_{ij} , полученные при данном k_{ij} не будут сбалансированы по всем блокам (то есть не дадут при поразрядном суммировании нулевой результат), значит догадка неверна. Перебрав максимум 2^8 вариантов байта раундового ключа, мы найдем его истинное значение.

По такому же принципу могут быть определены и другие байты раундового ключа. За счет того, что поиск может производиться отдельно (читай - параллельно) для каждого байта ключа, скорость подбора всего значения раундового ключа весьма велика. А по значению полного раундового ключа, при известном алгоритме его развертывания, не составляет труда восстановить сам начальный ключ шифрования.

Добавление пятого раунда в конец базовой атаки “Квадрат”

Если будет добавлен пятый раунд, то значения b_{ij} придется вычислять уже на основании выходных данных не четвертого, а пятого раунда. И

дополнительно кроме байта раундового ключа четвертого раунда перебирать еще значения четырех байт столбца раундового ключа для пятого раунда. Только так мы сможем выйти на значения сбалансированных байт b_{ij} входных данных четвертого раунда.

Таким образом, теперь нам нужно перебрать 2^{40} значений - 2^{32} вариантов для 4 байт столбца раундового ключа пятого раунда и для каждого из них 2^8 вариантов для одного байта четвертого раунда. Эту процедуру нужно будет повторить для всех четырех столбцов пятого раунда. Поскольку при подборе "верного" значения байта раундового ключа четвертого раунда количество "неверных" ключей уменьшается в 2^8 раз, то работая одновременно с пятью L-наборами, можно с большой степенью вероятности правильно подобрать все 2^{40} бита. Поиск может производиться отдельно (т. е. параллельно) для каждого столбца каждого из пяти L-наборов, что опять же гораздо быстрее полного перебора всех возможных значений ключа.

Добавление шестого раунда в начало базовой атаки “Квадрат”

Основная идея заключается в том, чтобы подобрать такой набор блоков открытого текста, который на выходе после первого раунда давал бы L-набор с одним активным байтом. Это требует предположения о значении четырех байт ключа, используемых функцией `AddRoundKey()` перед первым раундом.

Для того чтобы на входе второго раунда был только один активный байт достаточно, чтобы в первом раунде один активный байт оставался на выходе функции `MixColumns()`. Это означает, что на входе `MixColumns()` первого раунда должен быть такой столбец, байты а которого для набора из 256 блоков в результате линейного преобразования:

$$b_{ij} = 2a_i \oplus 3a_{i+1} \oplus a_{i+2} \oplus a_{i+3}, 0 \leq i \leq 3,$$

где i – номер строки, для одного определенного i давали 256 различных значений, в то время как для каждого из остальных трех значений i результат этого преобразования должен оставаться постоянным. Следуя обратно по

порядку приложения функций преобразования в первом раунде, к ShiftRows() данное условие нужно применить к соответственно разнесенным по столбцам 4 байтам. С учетом применения функции SubBytes() и сложения с предполагаемым значением 4-байтового раундового ключа можно смело составлять уравнения и подбирать нужные значения байт открытого текста, подаваемых на зашифрование для последующего анализа результата:

$$b_{ij} = 2SubBytes(a_{ij} \oplus k_{ij}) \oplus 3SubBytes(a_{(i+1)(j+1)} \oplus k_{(i+1)(j+1)}) \oplus \\ \oplus SubBytes(a_{(i+2)(j+2)} \oplus k_{(i+2)(j+2)}) \oplus SubBytes(a_{(i+3)(j+3)} \oplus k_{(i+3)(j+3)}), 0 \leq i, j \leq 3$$

Таким образом, получаем следующий алгоритм взлома. Имеем всего 2^{32} различных значений a для определенных i и j . Остальные байты для всех блоков одинаковы (пассивные байты). Предположив значения четырех байт k ключа первого раунда, подбираем (исходя из вышеописанного условия) набор из 256 блоков. Эти 256 блоков станут L-набором после первого раунда. К этому L-набору применима базовая атака для 4 раундов. Подобранный с ее помощью один байт ключа последнего раунда фиксируется. Теперь подбирается новый набор из 256 блоков для того же значения 4 байт k ключа первого раунда. Опять осуществляется базовая атака, дающая один байт ключа последнего раунда. Если после нескольких попыток значение этого байта не меняется, значит мы на верном пути. В противном случае нужно менять предположение о значении 4 байт k ключа первого раунда. Такой алгоритм действий достаточно быстро приведет к полному восстановлению всех байт ключа последнего раунда.

Таким образом, атака "Квадрат" может быть применена к 6 раундам шифра RIJNDAEL, являясь при этом более эффективной, чем полный перебор по всему ключевому пространству. Любое известное продолжение атаки "Квадрат" на 7 и более раундов становится более трудоемким, чем даже обычный полный перебор значений ключа.

3. Порядок выполнения работы

3.1. Интерфейс учебно-программного комплекса

Учебно-программный комплекс (в дальнейшем просто комплекс) был разработан на языке высокого уровня Visual Basic.

Главное окно

На рисунке 11 приведен интерфейс главного окна комплекса.

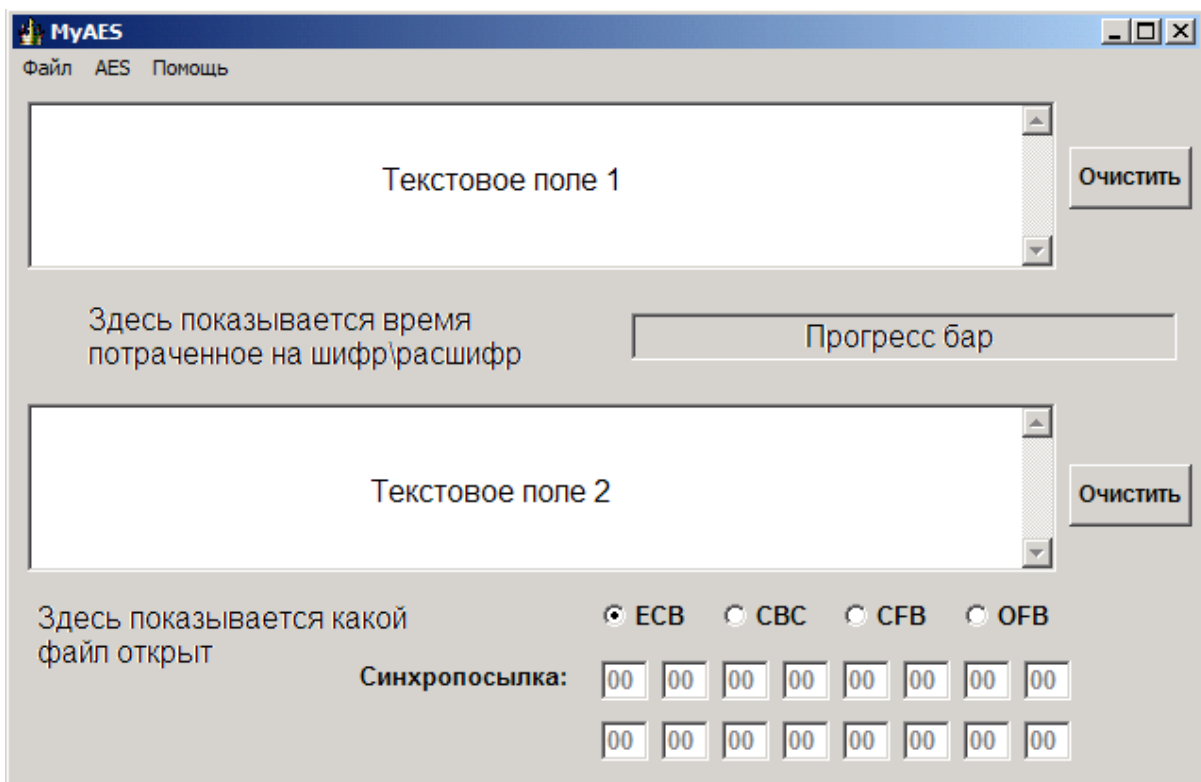


Рис. 11. Интерфейс главного окна комплекса

Главное окно состоит из:

1. Текстовое поле 1 – сюда вводится открытый текст, также если открывается файл с расширением txt, то его содержимое помещается сюда же.

2. Текстовое поле 2 – сюда помещается зашифрованный текст, также если открывается файл с расширением aes, то его содержимое помещается сюда же.

3. Прогресс бар – наглядно показывает время требуемое для шифрования \ расшифрования.

4. Кнопки “Очистить” – удаляют весь текст из текстовых полей.

5. Переключатели ECB, CBC, CFB, OFB – показывают, какой режим блочного шифрования используется.

6. Синхропосылка – задает синхропосылку для CBC, CFB и OFB.

В главном окне и происходит шифрование и расшифрование. Также в нем можно выбрать режимы симметричного шифрования (ECB, CBC, CFB, OFB). Про них сказано ниже.

Пункт меню “Файл”

На рисунке 12 приведен пункт меню “Файл”:

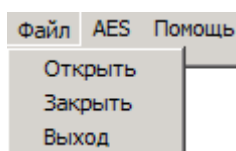


Рис. 12. Пункт меню “Файл”

Пункт меню “Файл” содержит следующие опции:

1. Открыть – открывает файл (*.txt, *.aes), но не больше 32кВ для текстовых файлов и 64кВ для файлов типа aes.

2. Закреть – закрывает выбранный файл.

3. Выход – завершает работу комплекса.

Пункт меню “AES”

На рисунке 13 приведен пункт меню “AES”:

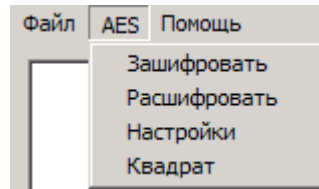


Рис. 13. Пункт меню “AES”

Пункт меню “AES” содержит следующие опции:

1. Зашифровать – шифрует текст, расположенный в текстовом поле 1, и помещает его в текстовое поле 2.
2. Расшифровать - расшифровывает текст, расположенный в текстовом поле 2, и помещает его в текстовое поле 1.
3. Настройки – позволяет выбрать свой ключ шифрования, а также включить \ выключить режим логирования. (рисунок 14)

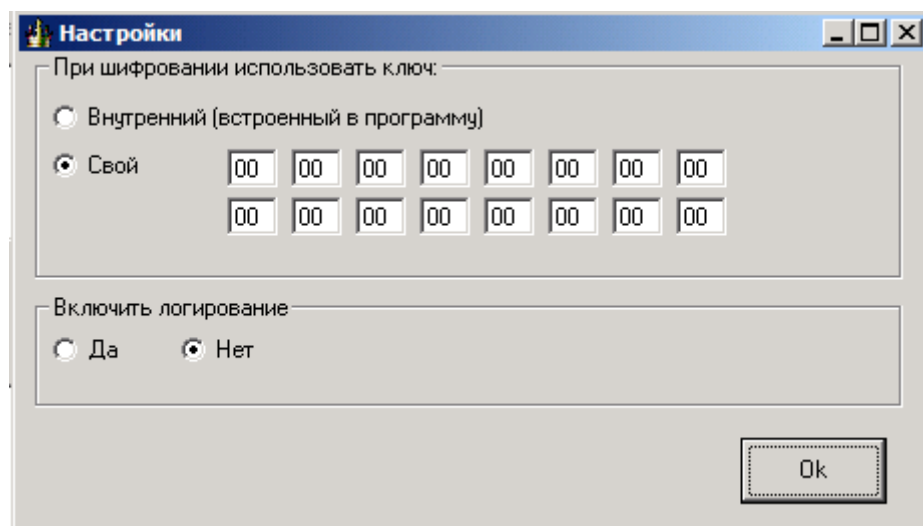


Рис. 14. Окно “Настройки”

В этом окне вы можете ввести свой ключ шифрования (например из задания по лабораторной работе) или же использовать внутренний. При использовании внутреннего ключа шифрования вы можете шифровать свои сообщения и отправлять их кому-либо. Причем получателю не обязательно знать ключ, главное чтобы у него была данная программа.

Здесь же вы можете включить логирование. При включенном логировании в папке программы создается Encrypt.txt (Decrypt.txt), в зависимости от того, что вы делаете шифруете или расшифровываете. В этом файле содержатся все преобразования, происходящие с открытым текстом. Пример такого файла приведен на рисунке 15:

```

Зашифрование
Входной блок CFF0EEE2E5F0EAE001010101010101
Раундовый ключ: 00000000000000000000000000000000
Состояние после сложения с ключом: CFF0EEE2E5F0EAE00101010101010101
Раунд №1
Состояние после SubBytes: 8A8C2898D98C87E17C7C7C7C7C7C7C7C
Состояние после ShiftRows: 8A8C7C7CD97C7C987C7C28E17C8C877C
Состояние после MixColumns: 80717A8DC93DEE5BB51D68098C916177
Раундовый ключ: 62636363626363636263636362636363
Состояние после сложения с ключом: E21219EEAB5E8D38D77E0B6AEEF20214
Раунд №2
Состояние после SubBytes: 98C9D42862585D070EF32B02288977FA
Состояние после ShiftRows: 98582BFA62F377280E89D40728C95D02
Состояние после MixColumns: 12AF832F952E07724F673D414F445DE8
Раундовый ключ: 9B9898C9F9FBFBA9B9898C9F9FBFBA
Состояние после сложения с ключом: 89371BE66CD5FCD8D4FFA588B6BFA642
Раунд №3
Состояние после SubBytes: A79AAF8E5003B061481606C44E08242C
Состояние после ShiftRows: A703062C5016248E4808AF614E9AB0C4
Состояние после MixColumns: 7A87DCAF309E87C546D3A6BD5D6EF86B
Раундовый ключ: 90973450696CCFFAF2F457330B0FAC99
Состояние после сложения с ключом: EA10E8FF59F2483FB427F18E566154F2

```

Рис. 15. Зашифрование слова “Проверка”

Логирование работает только в режиме ECB, и автоматически отключается при открытии файла или использовании другого режима.

4. Квадрат – простенькая реализация единственной атаки на AES, под названием квадрат. Реализована для сокращенной версии AES, для 4 раундов. В стандарте же их 10.

Окно Атака «Квадрат»

Смысл атаки “Квадрат” описан выше.

В этом окне (рисунок 16) можно просмотреть ключи для любого раунда. При открытии этого окна в поля “Реальный ключ” загружается ключ, заданный в настройках. При нажатии кнопок “Up” или ”Down” будет меняться номер раунда, а также поля “Реальный ключ”. Они будут равны тому раундовому ключу, который указан в строке “Номер раунда”. Нас будет интересовать ключ 4 раунда поэтому, лучше поставить значение поля “Номер раунда” в 4.

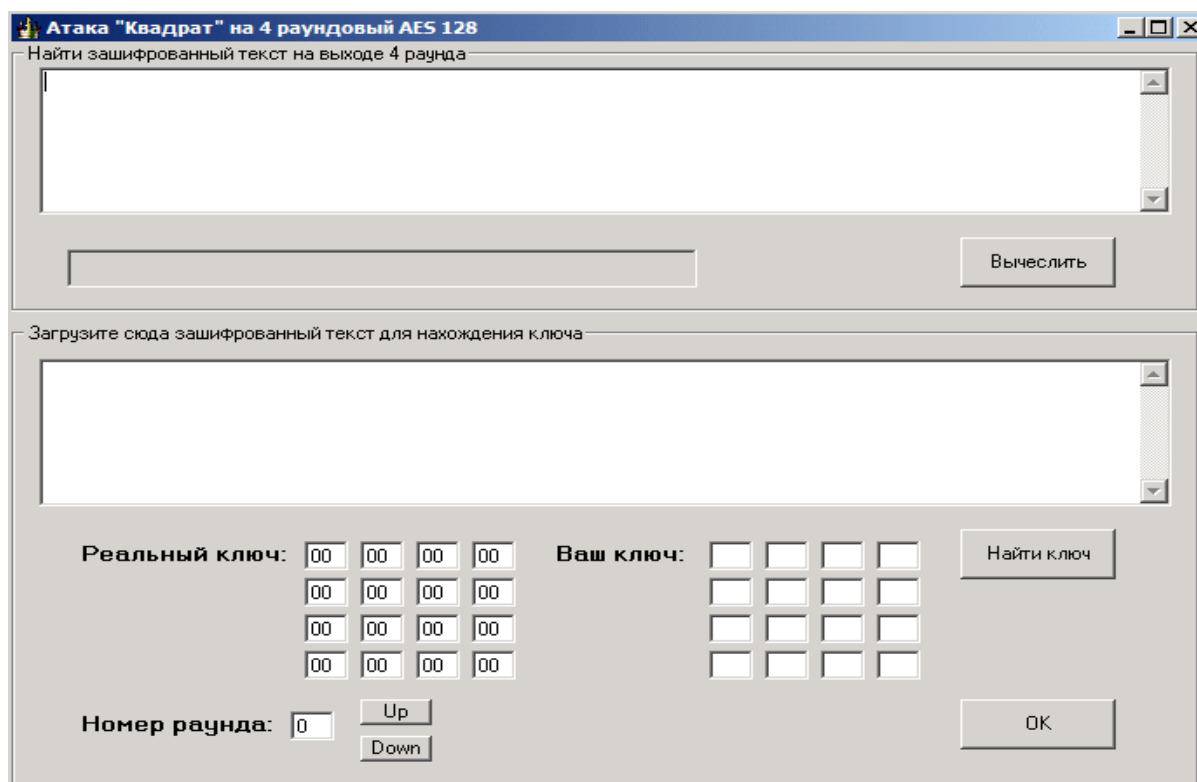


Рис. 16. Окно “Квадрат”

При нажатии кнопки “Вычислить” в текстовом поле 1 появляется зашифрованный L набор после 4 раунда. Прогресс бар наглядно показывает время требуемое для его вычисления. Пример L набора, зашифрованного ключом “00”{32}, показан на рисунке 17:

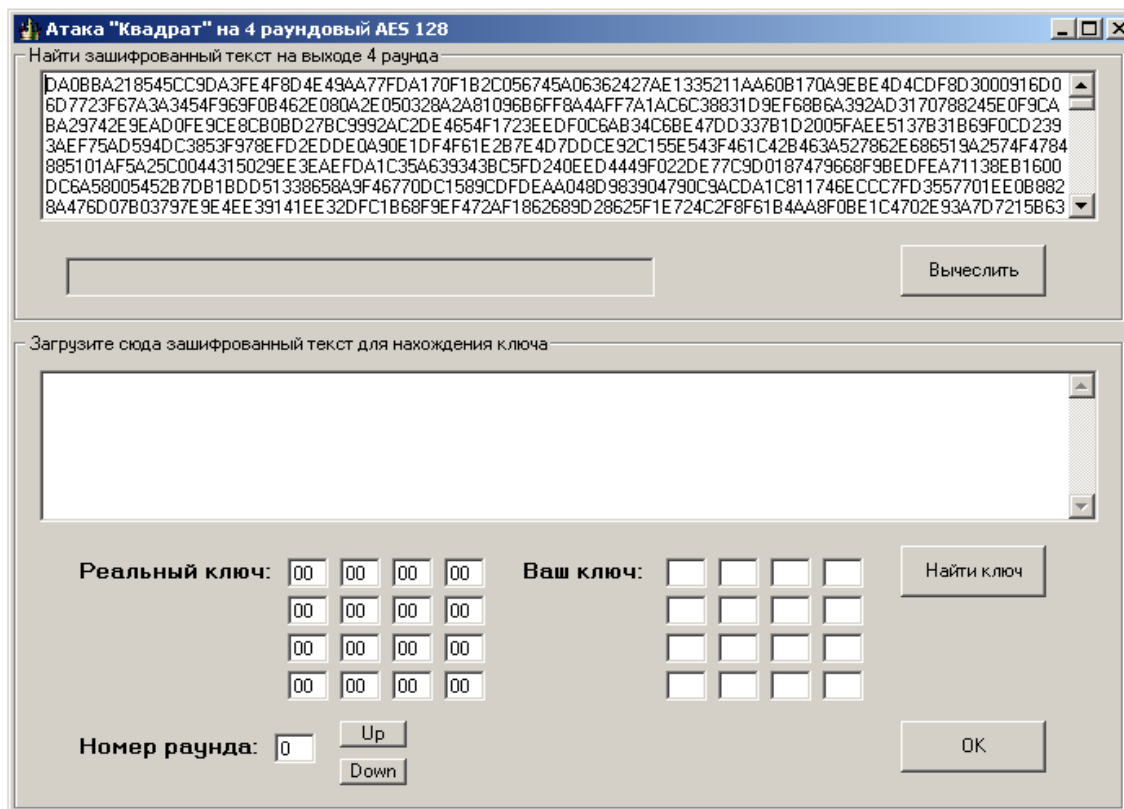


Рис. 17. Пример L набора, зашифрованного ключом “00”{32}

Этот текст нужно скопировать и вставить в текстовое поле 2.

При нажатии на кнопку “Найти ключ” начнется вычисление ключа 4 раунда на основе текста в текстовом поле 2. Прогресс бар наглядно показывает время, требуемое для его вычисления.

После вычисления ключа он появится в окошках “Ваш ключ”. Можно сравнить окна “Ваш ключ” и “Реальный ключ” и увидеть, что они практически совпадают (рисунок 18).

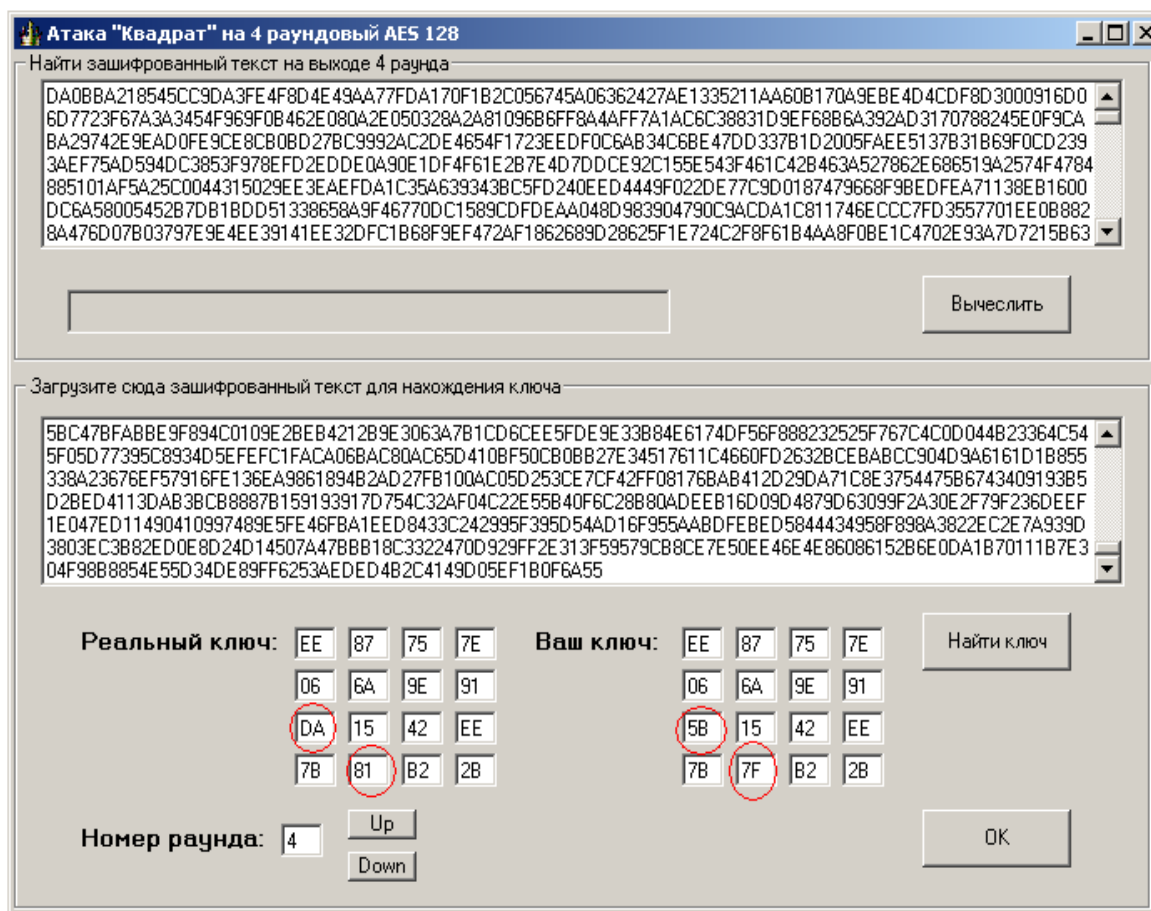


Рис. 18. Пример вычисления раундового ключа

Режимы ECB, CBC, CFB, OFB

Для различных ситуаций, встречающихся на практике, разработано значительное количество режимов шифрования.

Режим ECB (Electronic Code Book – режим электронной кодовой книги)

В режиме ECB каждый блок открытого текста заменяется блоком шифротекста. А так как один и тот же блок открытого текста заменяется одним и тем же блоком шифротекста, теоретически возможно создать кодовую книгу блоков открытого текста и соответствующих шифротекстов. Но если размер блока составляет n бит, кодовая книга будет состоять из 2^n записей.

Режим ECB - простейший режим шифрования. Все блоки открытого текста шифруются независимо друг от друга. Это важно для шифрованных

файлов с произвольным доступом, например, файлов баз данных. Если база данных зашифрована в режиме ECB, любая запись может быть добавлена, удалена, зашифрована или расшифрована независимо от любой другой записи (при условии, что каждая запись состоит из целого числа блоков шифрования). Кроме того, обработка может быть параллельной: если используются несколько шифровальных процессоров, они могут шифровать или расшифровывать различные блоки независимо друг от друга. Режим ECB показан на рисунке 19:

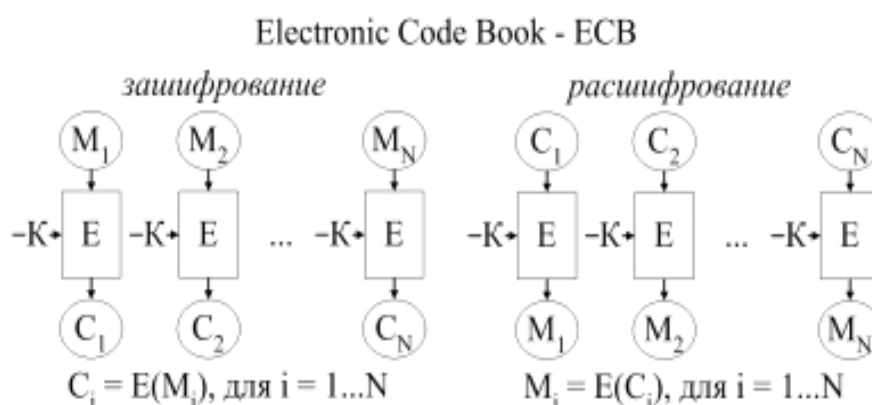


Рис. 19. Режим ECB

К недостаткам режима ECB можно отнести то обстоятельство, что если у криптоаналитика есть открытый текст и шифротекст нескольких сообщений, он может, не зная ключа, начать составлять шифровальную книгу. В большинстве реальных ситуаций фрагменты сообщений имеют тенденцию повторяться. В различных сообщениях могут быть одинаковые битовые последовательности, в сообщениях, которые, подобно электронной почте, создаются компьютером, могут быть периодически повторяющиеся структуры. Сообщения могут быть высоко избыточными или содержать длинные строки нулей или пробелов.

К достоинствам режима ECB можно отнести возможность шифрования нескольких сообщений одним ключом без снижения надежности. По существу, каждый блок можно рассматривать как отдельное сообщение, зашифрованное тем же самым ключом. При расшифровании ошибки в символах шифротекста ведут к некорректному расшифрованию соответствующего блока открытого текста,

однако не затрагивают остальной открытый текст. Но если бит шифротекста случайно потерян или добавлен, весь последующий шифротекст будет дешифрован некорректно, если только для выравнивания границ блоков не используется какое-нибудь выравнивание по границам блока. Большинство сообщений не делятся точно на n - битовые блоки шифрования – в конце обычно оказывается укороченный блок. Однако режим ECB требует использовать строго n - битовые блоки. Для решения этой проблемы используют дополнение (padding), Чтобы создать полный блок, последний блок дополняют некоторым стандартным шаблоном - нулями, единицами, чередующимися нулями и единицами.

Режим CBC (Ciphertext Block Chaining – режим сцепления блоков шифротекста)

Сцепление добавляет в блочный шифр механизм обратной связи: результаты шифрования предыдущих блоков влияют на шифрование текущего блока, т.е. каждый блок используется для модифицирования шифрования следующего блока. Каждый блок шифротекста зависит не только от шифруемого блока открытого текста, но и от всех предыдущих блоков открытого текста.

В режиме сцепления блоков шифротекста перед шифрованием над открытым текстом и предыдущим блоком шифротекста выполняется операция XOR. Процесс шифрования в режиме CBC. Когда блок открытого текста зашифрован, полученный шифротекст сохраняется в регистре обратной связи. Следующий блок открытого текста перед шифрованием подвергается операции XOR с содержимым регистра обратной связи. Результат операции XOR используется как входные данные для следующего этапа процедуры шифрования. Полученный шифротекст снова сохраняется в регистре обратной связи, чтобы подвергнуться операции XOR вместе со следующим блоком открытого текста, и так до конца сообщения. Шифрование каждого блока зависит от всех предыдущих блоков.

Расшифрование выполняется в обратном порядке. Блок шифротекста расшифровывается обычным путем, но сохраняется в регистре обратной связи. Затем следующий блок расшифровывается и подвергается операции XOR с содержимым регистра обратной связи. Теперь следующий блок шифротекста сохраняется в регистре обратной связи и т.д. до конца сообщения.

На рисунке 20 показан режим CBC (IV – синхропосылка):

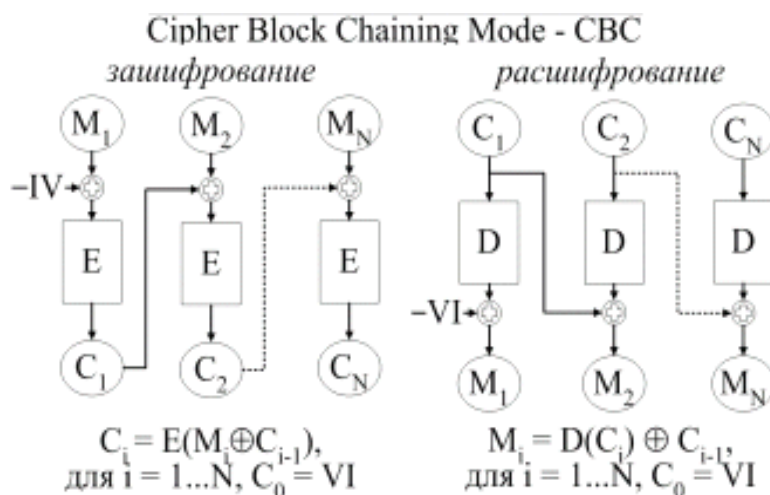


Рис. 20. Режим CBC

При шифровании в режиме CBC одинаковые блоки открытого текста превращаются в различающиеся друг от друга блоки шифротекста только в том случае, если различались какие-то предшествующие блоки открытого текста. Однако при шифровании двух 48 идентичных сообщений создается один и тот же шифротекст. Хуже того, два одинаково начинающихся сообщения будут шифроваться одинаково вплоть до первого различия.

Чтобы избежать этого, можно зашифровать в первом блоке какие-то произвольные данные. Этот блок случайных данных называют вектором инициализации (ВИ) (Initialization Vector, IV, русский термин - синхропосылка), инициализирующей переменной или начальным значением сцепления. Вектор ВИ не имеет какого-то смыслового значения, он используется только для того, чтобы сделать каждое сообщение уникальным. Когда получатель расшифровывает этот блок, он использует его только для заполнения регистра

обратной связи. В качестве вектора ВИ удобно использовать метку времени, либо какие-то случайные биты.

Если используется вектор инициализации, сообщения с идентичным открытым текстом после шифрования превращаются в сообщения с разными шифротекстами. Следовательно, злоумышленник не может попытаться повторить блок, и создание шифровальной книги затруднится. Хотя для каждого сообщения, шифруемого одним и тем же ключом, рекомендуется выбирать уникальный вектор ВИ, это требование необязательное. Вектор ВИ не обязательно хранить в секрете, его можно передавать открыто - вместе с шифротекстом.

Режим CFB (Ciphertext Feedback – обратная связь по шифротексту)

В режиме CFB начать шифрование до поступления полного блока данных невозможно. Для некоторых сетевых приложений это создает проблемы. Например, в защищенном сетевом окружении терминал должен иметь возможность передавать хосту каждый символ сразу после ввода. Если же данные нужно обрабатывать блоками в несколько байт, режим CFB просто не работает.

На рисунке 21 показан режим CFB (IV – синхропосылка):

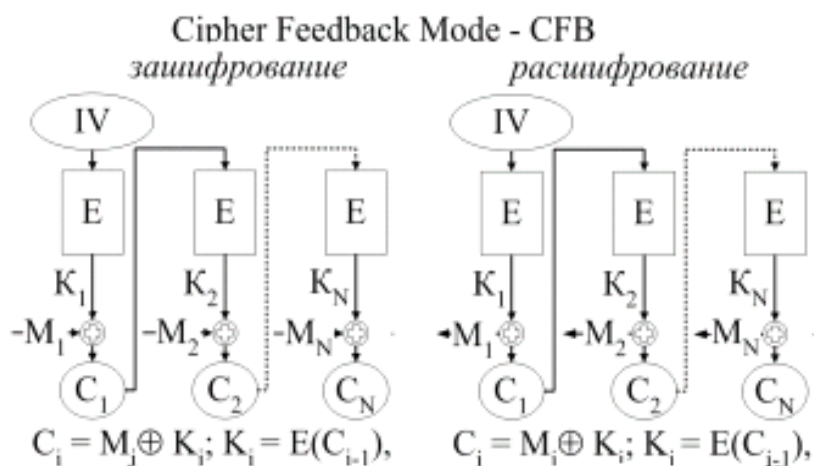


Рис. 21. Режим CFB

Как и в режиме CBC, первоначально очередь заполнена вектором инициализации ВИ. Очередь шифруется, затем выполняется операция XOR над n старшими (крайними левыми) битами результата и первым n -битовым символом открытого текста. В результате появляется первый n -битовый символ шифротекста. Теперь этот символ можно передать. Кроме того, полученные n битов попадают в очередь на место n младших битов, а все остальные биты сдвигаются на n позиций влево. Предыдущие n старших битов отбрасываются. Затем точно также шифруется следующие n битов открытого текста. Расшифрование выполняется в обратном порядке. Обе стороны - шифрующая и расшифровывающая - использует блочный алгоритм в режиме шифрования. Как и режим CBC, режим CFB сцепляет символы открытого текста с тем, чтобы шифротекст зависел от всего предыдущего открытого текста.

Для инициализации процесса шифрования в режиме CFB в качестве входного блока алгоритма можно использовать вектор инициализации ВИ. Как и в режиме CBC, хранить в тайне вектор ВИ не нужно. Однако вектор ВИ должен быть уникальным. (В отличие от режима CBC, где уникальность вектора ВИ необязательна, хотя и желательна). Если вектор ВИ в режиме CFB не уникален, криптоаналитик может восстановить соответствующий открытый текст. Вектор инициализации должен меняться в каждом сообщении. Например, вектором ВИ может служить порядковый номер, возрастающий в каждом новом сообщении и не повторяющийся все время жизни ключа. Если данные шифруются с целью последующего хранения, вектор ВИ может быть функцией индекса, используемого для поиска данных.

Режим OFB (Output Feedback – режим обратной связи по выходу)

Режим OFB представляет собой метод использования блочного шифра в качестве синхронного потокового шифра. Этот режим подобен режиму CFB.

Блочный алгоритм работает в режиме шифрования как на шифрующей, так и на расшифровывающей сторонах. Такую обратную связь иногда называют

внутренней, поскольку механизм обратной связи не зависит ни от потока открытого текста, ни от потока шифротекста.

К достоинствам режима OFB относится то, что большую часть работы можно выполнить оффлайново, даже когда открытого текста сообщения еще вовсе не существует. Когда, наконец, сообщение поступает, для создания шифротекста над сообщением и выходом алгоритма необходимо выполнить операцию XOR.

В сдвиговый регистр OFB сначала надо загрузить вектор ВИ. Вектор должен быть уникальным, но сохранять его в тайне не обязательно.

На рисунке 22 показан режим OFB (IV – синхропосылка):

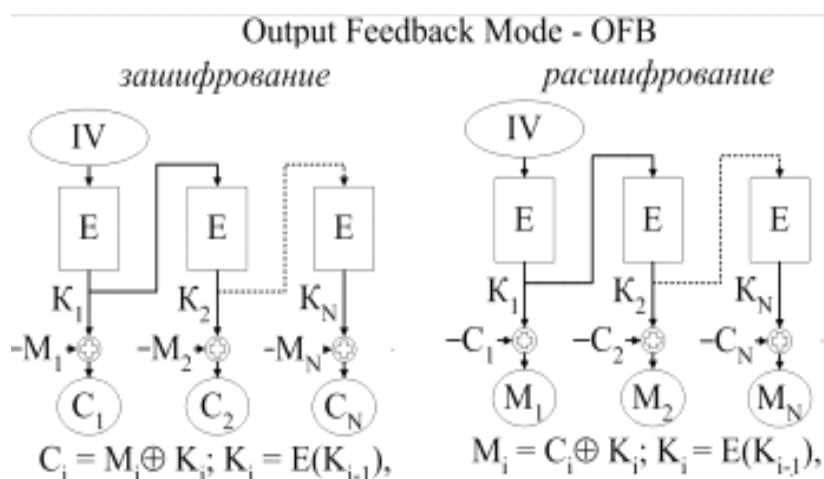


Рис. 22. Режим OFB

Анализ режима OFB показывает, что OFB целесообразно использовать только если разрядность обратной связи совпадает с размером блока.

В режиме OFB над гаммой и текстом выполняется операция XOR. Эта гамма, в конце концов, повторяется. Существенно, чтобы она не повторялась для одного и того же ключа, в противном случае секретность не обеспечивается ничем.

3.2. Порядок выполнения работы

1. Ознакомится с теорией по стандарту AES (из чего состоит раунд шифрования, как работают раундовые преобразования, с необходимым математическим аппаратом).

2. Взять у преподавателя задание, состоящие из:

- уникального ключа для шифрования/дешифрования текстовых сообщений
- набора файлов разного размера для исследования стандарта
- файла для исследования режимов шифрования
- вектора инициализации
- файла для проведения атаки «Квадрат»

3. Исследование стандарт шифрования

3.1. Открыть окно «Настройки» (AES>Настройки), установить переключатель в положение «Свой ключ» и ввести посимвольное значение ключа (32 символа, 16 ячеек). Нажать «ОК».

3.2. В главном окне поставить переключатель в положение ECB.

3.3. Открыть файл из папки программы (Файл >Открыть)

3.4. Зашифровать и расшифровать его. (AES > Зашифровать (Расшифровать)).

3.5. Прodelать пункты 3.2-3.4 для оставшихся трех файлов.

3.5. По полученным результатам заполнить таблицу.

Размер файла				
Время зашифрования				
Время расшифрования				

3.6. Построить графики зависимости размера файла от времени его шифрования \ расшифрования. На рисунке 23 приведен пример графика зависимости размера файла от времени его шифрования \ расшифрования при проведении эксперимента на Athlon 2600+.

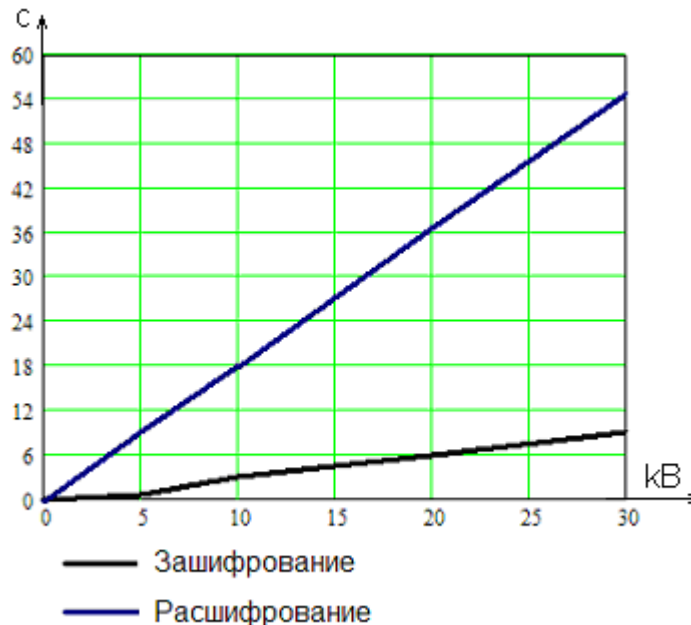


Рис. 23. Пример зависимости размера файла от времени его шифрования \ расшифрования при проведении эксперимента на Athlon 2600+

3.7. По полученным данным оценить примерное время взлома стандарта AES с помощью всего перебора ключей. Построить график.

Пример:

На расшифрование 5кВ (5120 байт) текста на Athlon 2600+ тратится 9с. Поэтому один блок (16 байт) программа расшифровывает за $9/(5120/16) = (9*16)/5120 = 28$ мс. Таким образом, на перебор 1 ключа, а это 128 бит, потребуется 28 мс.

За 1 секунду программа способна перебрать 4571 бит или примерно 35 вариантов ключей.

За год непрерывной работы программа переберет $(35*60*60*24*365) = 1\ 103\ 760\ 000$ ключей.

Если поделить 2^{128} на 1 103 760 000 , то получим $3.08 \cdot 10^{29}$ лет непрерывной работы программы.

3.8. Сделать выводы

4. Исследование быстродействия различных режимов шифрования

4.1. Поставить переключатель в режим CBC.

4.2. Открыть файл из папки программы (Файл >Открыть)

4.3. Зашифровать и расшифровать его (AES > Зашифровать (Расшифровать))

4.4. Повторить пункт 4.2-4.3 для оставшихся трех файлов

4.5. Поставить переключатель в режим CFB.

4.6. Повторить пункты 4.2 – 4.3.

4.7. Поставить переключатель в режим OFB.

4.8. Повторить пункты 4.2 – 4.3

4.9. По полученным результатам заполнить таблицу:

Размер файла				
ECB время заш/расш				
CBC время заш/расш				
CFB время заш/расш				
OFB время заш/расш				

4.10. Построить на одном графике зависимости размера файла от времени его шифрования \ расшифрования

4.11. Для каждого режима загрузить еще раз файл (любой). Зашифровать его. В текстовом поле 2 изменить какой-нибудь символ. Нажать AES > Расшифровать.

4.12. Сделать выводы.

5. Исследование свойств различных режимов шифрования

5.12. Загрузить файл «Свойства режимов».

5.13. Поочередно зашифровать текстовое сообщение во всех четырех режимах.

5.14. Сравнить полученные шифротексты и сделать выводы.

6. Исследование пораундовой работы алгоритма

6.1 Открыть любой из предоставленных для работы файлов

6.2. Включить функцию логирования (AES>Настройки>Включить логирование>Да).

6.3. В главном окне поставить переключатель в положение ECB.

6.4. Зашифровать и расшифровать.

6.5. Открыть с помощью текстового редактора файлы Encrypt.txt и Decrypt.txt из папки программы

6.6. Проанализировать полученный результат.

7. Знакомство с атакой «Квадрат»

7.1. В главном окне в текстовое поле 1 загрузить файл «Атака Квадрат» (Файл>Открыть)

7.2. Открыть окно Атака «Квадрат» (AES > Квадрат)

7.3. С помощью переключателя “Номер раунда” установить 4 раундовый ключ.

7.4. Нажать кнопку “вычислить”.

7.5. Поместить текст из поля 1 в поле 2.

7.6. Нажать кнопку найти ключ.

7.7. Сравнить значения текстовых полей “Ваш ключ” и значения текстовых полей “Реальный ключ”.

7.8. Сделать выводы.

8. Содержание отчета

8.1. Цель работы.

8.2. Описание действий по каждому пункту.

8.3. Результаты проделанной работы (таблицы, графики, расчетная часть).

8.4. Выводы.

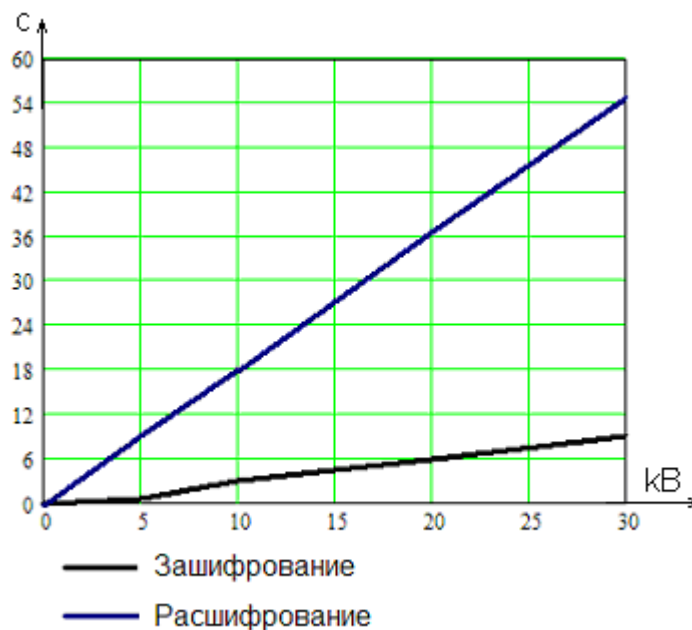
4 Контрольные вопросы

1. Что такое симметричное шифрование?
2. Что представляет собой стандарт AES (длина ключа, размер входного блока)?
3. Какой алгоритм выбран в качестве стандарта AES?
4. Что собой представляет архитектура данного стандарта?
5. Из чего состоит один раунд?
6. Сколько раундов шифрования предусмотрено стандартом?
7. Что быстрее шифрование или дешифрование? Почему?
8. Какие режимы шифрования применяются в стандарте AES?
9. Какие режимы быстрее при дешифровании? Почему?
10. Какие режимы лучше восстанавливают зашифрованную информацию при ошибке в одном символе? Почему?
11. С какой целью используется синхропосылка или вектор инициализации?
12. Что представляет собой атака Квадрат? Какие ее особенности?

ПРИЛОЖЕНИЕ

Исследование стандарт шифрования

Размер файла	5kB	10kB	20kB	30kB
Время зашифрования, с	1	3	6	9
Время расшифрования, с	9	18	37	55



Зависимости размера файла от времени его шифрования \ расшифрования
при проведении эксперимента на Athlon 2600+

Исследование быстродействия различных режимов шифрования

Размер файла	5kB	10kB	20kB	30kB
ECB время зашифрования	1	3	6	9
ECB время расшифрования	9	18	37	55
CBC время зашифрования	1	3	6	9
CBC время расшифрования	8	17	37	56
CFB время зашифрования	1	3	5	9
CFB время расшифрования	2	3	6	10
OFB время зашифрования	1	3	6	9

OFB время расшифрования	2	3	7	9
----------------------------	---	---	---	---

Зашифруем разными режимами слово пример, затем в шифротексте изменим 1 символ, получим:

	Исходный текст	Исходный шифротекст	Измененный шифротекст	Результат
ECB	пример	63891BCDBE14D6E7 B827C0AF68EF5419	63891BCDBE14D6E7 B827C0AF68EF5410	ЛМЎ!ня«□ □ m;5J9w
CBC	пример	000000000000000000 000000000000006389 1BCDBE14D6E7B82 7C0AF68EF5419	100000000000000000 000000000000006389 1BCDBE14D6E7B82 7C0AF68EF5419	япример
CFB	пример	000000000000000000 0000000000000008919 A3380A7A2D3A894D FB58CB352A2F	000000000000000000 0000000000000068919 A3380A7A2D3A894D FB58CB352A2F	□ ЗЮSÿ”□ [ж<Яе□ (s
OFB	пример	000000000000000000 000000000000008919 A3380A7A2D3A894D FB58CB352A2F	000000000000100000 00000000000008919 A3380A7A2D3A894D FB58CB352A2F	μZi'a□ Ij·jOŃC□ &Ú

Исследование свойств различных режимов шифрования

Файл «Свойства режимов» содержит несколько раз повторяющееся слово.

Например, *задание задание задание задание задание задание задание задание*.

Каждый режим при этом покажет свои особенности:

1. ECB

задание задание задание задание задание задание задание задание

D3D5451C8296E7E653A82652AED4E56A
D3D5451C8296E7E653A82652AED4E56A
D3D5451C8296E7E653A82652AED4E56A
A8F11E11A02FF140F0936291695512CA

2. CBC

00000000000000000000000000000000
D3D5451C8296E7E653A82652AED4E56A
F84E11957A1A02F67B97ED2A9DE04DD2
00D20F40BEE5D0C7EEF499531BEF8B8C
9BB124631613C483AE57E40BCD63C2AC

3. CFB

00000000000000000000000000000000
8109AF340262C91B6FAC1EB927DCCE0E
6D376DFF127D70059697E293142F5DA6
DBC4EAC4CA79A6139E944EB9B374DDAD
4128D7EF70BE71BB9F57B9D64DF4EC68

4. OFB

00000000000000000000000000000000
8109AF340262C91B6FAC1EB927DCCE0E
107559AABF0A7BF7F433F71ACD01689C
46EC128DE2351660B490505F601D5A93
A02767750D304B01733AEE7046216D8B

В данном примере синхропосыла равна 00000000000000000000000000000000

Исследование пораундовой работы алгоритма

Используем слово «пример»

Зашифрование:

Зашифрование

Входной блок EFF0E8ECE5F001010101010101010101

Раундовый ключ: 00000000000000000000000000000000

Состояние после сложения с ключом: EFF0E8ECE5F001010101010101010101

Раунд №1

Состояние после SubBytes: DF8C9BCED98C7C7C7C7C7C7C7C7C7C

Состояние после ShiftRows: DF8C7C7CD97C7CCE7C7C9B7C7C8C7C7C

Состояние после MixColumns: 2A242F729F6B14F79B4EA99B77878C8C

Раундовый ключ: 62636363626363636263636362636363

Состояние после сложения с ключом:

48474C11FD087794F92DCAF815E4EFEF

Раунд №2

Состояние после SubBytes: 52A029825430F52299D874415969DFDF

Состояние после ShiftRows: 523074DF54D8DF829969292259A0F541

Состояние после MixColumns: 5F71F0178607B4E49912C4B4FD47CB3C

Раундовый ключ: 9B9898C9F9FBFBA9B9898C9F9FBFBA9

Состояние после сложения с ключом:

C4E968DE7FFC4F4E028A5C7D04BC3096

Раунд №3

Состояние после SubBytes: 1C1E451DD2B0842F777E4AFF2650490

Состояние после ShiftRows: 1CB04A90D27E041D7765452FF21E84FF

Состояние после MixColumns: 292993E5243F832D2B5DE9E7A6A6E572

Раундовый ключ: 90973450696CCFFAF2F457330B0FAC99

Состояние после сложения с ключом:

B9BEA7B54D534CD7D9A9BED4ADA949EB

Раунд №4

Состояние после SubBytes: 56AE5CD5E3ED290E35D3AE4895D33BE9

Состояние после ShiftRows: 56EDAEE9E3D33BD535D35C0E95AE2948

Состояние после MixColumns: C797DC705DC6226756624CCCB9E1B1B3

Раундовый ключ: EE06DA7B876A1581759E42B27E91EE2B

Состояние после сложения с ключом:

2991060BDAAC37E623FC0E7EC7705F98

Раунд №5

Состояние после SubBytes: A5816F2B57919A8E26B0ABF3C651CF46

Состояние после ShiftRows: A591AB4657B0CF2B26516F8EC6819AF3

Состояние после MixColumns: 143CB342814D1FD05EBB2053669966B7

Раундовый ключ: 7F2E2B88F8443E098DDA7CBBF34B9290

Состояние после сложения с ключом:

6B1298CA790921D9D3615CE895D2F427

Раунд №6

Состояние после SubBytes: 7FC94674B601FD3566EF4A9B2AB5BFCC

Состояние после ShiftRows: 7F014ACCB6EFBF7466B546352AC9FD9B

Состояние после MixColumns: 7B6FA54996DDA0797BE800337224B467

Раундовый ключ: EC614B851425758C99FF09376AB49BA7

Состояние после сложения с ключом:

970EEEC82F8D5F5E217090418902FC0

Раунд №7

Состояние после SubBytes: 88AB284B134103E698F001F2AD6015BA

Состояние после ShiftRows: 884101BA13F0154B986028E6ADAB03F2

Состояние после MixColumns: 73B31EAC739C144645C6992C56170DBB

Раундовый ключ: 217517873550620BACAF6B3CC61BF09B

Состояние после сложения с ключом:

52C6092B46CC764DE969F210900CFD20

Раунд №8

Состояние после SubBytes: 00B401F15A4B38E31EF989CA60FE54B7
Состояние после ShiftRows: 004B89B75AF954F11EFE01E360B438CA
Состояние после MixColumns: E3A180B701BE03BAC719DC00F591E1A3
Раундовый ключ: 0EF903333BA9613897060A04511DFA9F
Состояние после сложения с ключом:
ED5883843A176282501FD604A48C1B3C

Раунд №9

Состояние после SubBytes: 556AEC5F80F0AA1353C0F6F24964AFEB
Состояние после ShiftRows: 55F0F6EB80C0AF5F5364EC13496AAAF2
Состояние после MixColumns: BC447434B0AEE44AF5A7C15B748A61E4
Раундовый ключ: B1D4D8E28A7DB9DA1D7BB3DE4C664941
Состояние после сложения с ключом:
0D90ACD63AD35D90E8DC728538EC28A5

Раунд №10

Состояние после SubBytes: D76091F680664C609B86409707CE3406
Состояние после ShiftRows: D7664006808634F69BCE916007604C97
Раундовый ключ: B4EF5BCB3E92E21123E951CF6F8F188E
Состояние после сложения с ключом:
63891BCDBE14D6E7B827C0AF68EF5419

Расшифрование

Разшифрование

Входной блок 63891BCDBE14D6E7B827C0AF68EF5419
Раундовый ключ: B4EF5BCB3E92E21123E951CF6F8F188E
Состояние после сложения с ключом: D7664006808634F69BCE916007604C97

Раунд №1

Состояние после invShiftRows: D76091F680664C609B86409707CE3406
Состояние после invSubBytes: 0D90ACD63AD35D90E8DC728538EC28A5
Раундовый ключ: B1D4D8E28A7DB9DA1D7BB3DE4C664941

Состояние после сложения с ключом:

BC447434B0AEE44AF5A7C15B748A61E4

Состояние после invMixColumns: 55F0F6EB80C0AF5F5364EC13496AAAF2

Раунд №2

Состояние после invShiftRows: 556AEC5F80F0AA1353C0F6F24964AFEB

Состояние после invSubBytes: ED5883843A176282501FD604A48C1B3C

Раундовый ключ: 0EF903333BA9613897060A04511DFA9F

Состояние после сложения с ключом:

E3A180B701BE03BAC719DC00F591E1A3

Состояние после invMixColumns: 004B89B75AF954F11EFE01E360B438CA

Раунд №3

Состояние после invShiftRows: 00B401F15A4B38E31EF989CA60FE54B7

Состояние после invSubBytes: 52C6092B46CC764DE969F210900CFD20

Раундовый ключ: 217517873550620BACAF6B3CC61BF09B

Состояние после сложения с ключом:

73B31EAC739C144645C6992C56170DBB

Состояние после invMixColumns: 884101BA13F0154B986028E6ADAB03F2

Раунд №4

Состояние после invShiftRows: 88AB284B134103E698F001F2AD6015BA

Состояние после invSubBytes: 970EEEC82F8D5F5E217090418902FC0

Раундовый ключ: EC614B851425758C99FF09376AB49BA7

Состояние после сложения с ключом:

7B6FA54996DDA0797BE800337224B467

Состояние после invMixColumns: 7F014ACCB6EFBF7466B546352AC9FD9B

Раунд №5

Состояние после invShiftRows: 7FC94674B601FD3566EF4A9B2AB5BFCC

Состояние после invSubBytes: 6B1298CA790921D9D3615CE895D2F427

Раундовый ключ: 7F2E2B88F8443E098DDA7CBBF34B9290

Состояние после сложения с ключом:

143CB342814D1FD05EBB2053669966B7

Состояние после invMixColumns: A591AB4657B0CF2B26516F8EC6819AF3

Раунд №6

Состояние после invShiftRows: A5816F2B57919A8E26B0ABF3C651CF46

Состояние после invSubBytes: 2991060BDAAC37E623FC0E7EC7705F98

Раундовый ключ: EE06DA7B876A1581759E42B27E91EE2B

Состояние после сложения с ключом:

C797DC705DC6226756624CCCB9E1B1B3

Состояние после invMixColumns: 56EDAEE9E3D33BD535D35C0E95AE2948

Раунд №7

Состояние после invShiftRows: 56AE5CD5E3ED290E35D3AE4895D33BE9

Состояние после invSubBytes: B9BEA7B54D534CD7D9A9BED4ADA949EB

Раундовый ключ: 90973450696CCFFAF2F457330B0FAC99

Состояние после сложения с ключом:

292993E5243F832D2B5DE9E7A6A6E572

Состояние после invMixColumns: 1CB04A90D27E041D7765452FF21E84FF

Раунд №8

Состояние после invShiftRows: 1C1E451DD2B0842F777E4AFF2650490

Состояние после invSubBytes: C4E968DE7FFC4F4E028A5C7D04BC3096

Раундовый ключ: 9B9898C9F9FBFBA9B9898C9F9FBFBA

Состояние после сложения с ключом:

5F71F0178607B4E49912C4B4FD47CB3C

Состояние после invMixColumns: 523074DF54D8DF829969292259A0F541

Раунд №9

Состояние после invShiftRows: 52A029825430F52299D874415969DFDF

Состояние после invSubBytes: 48474C11FD087794F92DCAF815E4EFEF

Раундовый ключ: 62636363626363636263636362636363

Состояние после сложения с ключом:

2A242F729F6B14F79B4EA99B77878C8C

Состояние после invMixColumns: DF8C7C7CD97C7CCE7C7C9B7C7C8C7C7C

Раунд №10

Состояние после invShiftRows: DF8C9BCED98C7C7C7C7C7C7C7C7C7C

Состояние после invSubBytes: EFF0E8ECE5F001010101010101010101

Раундовый ключ: 00000000000000000000000000000000

Состояние после сложения с ключом: EFF0E8ECE5F001010101010101010101

4. Рекомендуемая литература

1. Метлицкий Ю.В. Разработка программного комплекса для визуализации и анализа стандарта криптографической защиты AES, МИФИ 2003 год.
2. www.intuit.ru, Криптографические основы информации.
3. Зензин О.С., Иванов М.А. Стандарт криптографической защиты – AES. Конечные поля. – М: КУДИЦ-ОБРАЗ, 2002-176с.

Лабораторная работа 5. Исследование отечественных стандартов хэш-функции (ГОСТ Р 34.11-94) и электронной цифровой подписи (ЭЦП ГОСТ Р 34.10-2001)

1. Цель работы

Целью данной лабораторной работы является изучение процессов вычисления функции хэширования ГОСТ Р 34.11-94, формирования и проверки электронной цифровой подписи (ЭЦП) ГОСТ Р 34.10-2001.

2 Методические указания

2.1 Основные понятия и определения

При обмене электронными документами по сети связи существенно снижаются затраты на обработку и хранение документов, убыстряется их поиск. Но при этом возникает проблема аутентификации автора документа и самого документа, т.е. установления подлинности автора и отсутствия изменений в полученном документе. В обычной (бумажной) информатике эти проблемы решаются за счет того, что информация в документе и рукописная подпись автора жестко связаны с физическим носителем (бумагой). В электронных документах на машинных носителях такой связи нет.

Целью аутентификации электронных документов является их защита от возможных видов злоумышленных действий, к которым относятся:

- *активный перехват* – нарушитель, подключившись к сети, перехватывает документы (файлы) и изменяет их;
- *маскарад* – абонент *С* посылает документ абоненту *В* от имени абонента *А*;
- *рenegатство* – абонент *А* заявляет, что не посылал сообщения абоненту *В*, хотя на самом деле послал;
- *подмена* – абонент *В* изменяет или формирует новый документ и заявляет, что получил его от абонента *А*;

- *повтор* – абонент С повторяет ранее переданный документ, который абонент А посылал абоненту В.

Эти виды злоумышленных действий могут нанести существенный ущерб банковским и коммерческим структурам, государственным предприятиям и организациям, частным лицам, применяющим в своей деятельности компьютерные информационные технологии.

При обработке документов в электронной форме совершенно непригодны традиционные способы установления подлинности по рукописной подписи и оттиску печати на бумажном документе. Принципиально новым решением является электронная цифровая подпись (ЭЦП).

Электронная цифровая подпись используется для аутентификации текстов, передаваемых по телекоммуникационным каналам. Функционально она аналогична обычной рукописной подписи и обладает ее основными достоинствами:

- удостоверяет, что подписанный текст исходит от лица, поставившего подпись;
- не дает самому этому лицу возможности отказаться от обязательств, связанных с подписанным текстом;
- гарантирует целостность подписанного текста.

Цифровая подпись представляет собой относительно небольшое количество дополнительной цифровой информации, передаваемой вместе с подписываемым текстом.

Система ЭЦП включает две процедуры:

1. процедуру постановки подписи;
2. процедуру проверки подписи.

В процедуре постановки подписи используется секретный ключ отправителя сообщения, в процедуре проверки подписи – открытый ключ отправителя.

При формировании ЭЦП отправитель, прежде всего, вычисляет хэш-функцию $h(M)$ подписываемого текста M . Вычисленное значение хэш-функции $h(M)$ представляет собой один короткий блок информации t , характеризующий весь текст M в целом. Затем число t шифруется секретным ключом отправителя. Получаемая при этом пара чисел представляет собой ЭЦП для данного текста M .

При проверке ЭЦП получатель сообщения снова вычисляет хэш-функцию $t = h(M)$ принятого по каналу текста M , после чего при помощи открытого ключа отправителя проверяет, соответствует ли полученная подпись вычисленному значению t хэш-функции.

Принципиальным моментом в системе ЭЦП является невозможность подделки ЭЦП пользователя без знания его секретного ключа подписывания.

В качестве подписываемого документа может быть использован любой файл.

2.2 Функция хэширования

Функция хэширования (хэш-функция, или дайджест-функция) представляет собой отображение, на вход которого подается сообщение переменной длины M , а выходом является строка фиксированной длины $h(M)$. Иначе говоря, хэш-функция h принимает в качестве аргумента сообщение (документ) M произвольной длины и возвращает хэш-значение $h(M)$ фиксированной длины.

Хэш-значение $h(M)$ – это сжатое двоичное представление (дайджест) основного сообщения M произвольной длины. Функция хэширования позволяет сжать подписываемый документ M до нескольких десятков или сотен бит (например, 128 или 256 бит), тогда как M может быть размером в мегабайт или более. Следует отметить, что значение хэш-функции $h(M)$ зависит от документа M сложным образом и не позволяет восстановить сам документ M .

Функция хэширования может использоваться для обнаружения модификации сообщения, то есть служить в качестве криптографической

контрольной суммы (также называемой кодом обнаружения изменений или кодом аутентификации сообщения). В этом качестве функция хэширования применяется при формировании и проверке электронной цифровой подписи [1].

Для того чтобы функция хэширования могла быть надлежащим образом использована в процессе аутентификации, она должна обладать следующими свойствами.

1. Хэш-функция может быть применена к аргументу любого размера.
2. Выходное значение хэш-функции имеет фиксированный размер.
3. Хэш-функцию $h(x)$ достаточно просто вычислить для любого x .
4. Хэш-функция должна быть чувствительна к всевозможным изменениям в тексте M .
5. Хэш-функция должна обладать свойством необратимости, иначе говоря, задача подбора документа M' , который обладал бы требуемым значением хэш-функции, должна быть вычислительно неразрешима; то есть для любого y с вычислительной точки зрения невозможно найти такое x , при котором $h(x) = y$.
6. Вероятность того, что значения хэш-функций двух различных документов (вне зависимости от их длин) совпадут, должна быть ничтожно мала; то есть для любого фиксированного x с вычислительной точки зрения невозможно найти такое $x' \neq x$, при котором $h(x') = h(x)$.

Теоретически допустимо, что два различных сообщения могут быть сжаты в одну и ту же свертку (так называемая коллизия или “столкновение”). Поэтому для обеспечения стойкости функции хэширования необходимо предусмотреть способ избежания столкновений. Полностью столкновений избежать нельзя, поскольку в общем случае количество возможных сообщений превышает количество возможных выходных значений функции хэширования. Однако вероятность столкновения должна быть низкой.

Свойство 5 эквивалентно тому условию, что $h(M)$ является односторонней функцией. Свойство 6 гарантирует, что не может быть найдено другое

сообщение, дающее ту же свертку. Это предотвращает подделку и также позволяет использовать значение хэш-функции в качестве криптографической контрольной суммы для аутентификации пользователей и проверки целостности сообщения.

Большинство хэш-функций строится на основе однонаправленной функции f , которая образует выходное значение длиной n при задании двух входных значений длиной n . Этими входами являются блок исходного текста M_i и хэш-значение H_{i-1} предыдущего блока текста (рисунок 2.1): $H_i = f(M_i, H_{i-1})$.

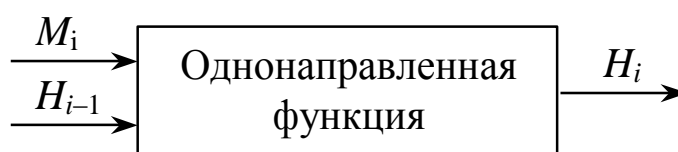


Рис. 2.1. Построение однонаправленной хэш-функции

Хэш-значение, вычисляемое при вводе последнего блока текста, становится хэш-значением всего сообщения M .

В результате однонаправленная хэш-функция всегда формирует выход фиксированной длины n (независимо от длины входного текста).

Примерами хэш-функций являются SHA-1 (Secure Hash Algorithm) и ГОСТ Р 34.11-94.

Российский стандарт ГОСТ Р 34.11-94 определяет алгоритм и процедуру вычисления хэш-функции для любых последовательностей двоичных символов, применяемых в криптографических методах обработки и защиты информации. Этот стандарт базируется на блочном алгоритме шифрования ГОСТ 28147-89, хотя в принципе можно было бы использовать и другой блочный алгоритм шифрования с 64-битовым блоком и 256-битовым ключом.

Сообщения с произвольной длины можно сжать, используя хэш-функцию с фиксированным размером входа, при помощи двух методов:

- последовательного (итерационного);

- параллельного.

Создатели ГОСТ Р 34.11 – 94 пошли по первому пути и использовали метод последовательного хэширования, использующий хэш-функцию с фиксированным размером входа $h: \{01\}^{2n} \rightarrow \{01\}^n$ (рисунок 2.2), то есть функцию сжатия с коэффициентом 2.

$\{01\}^n$ и $\{01\}^{2n}$ – множества двоичных строк длиной n и $2n$ бит соответственно.

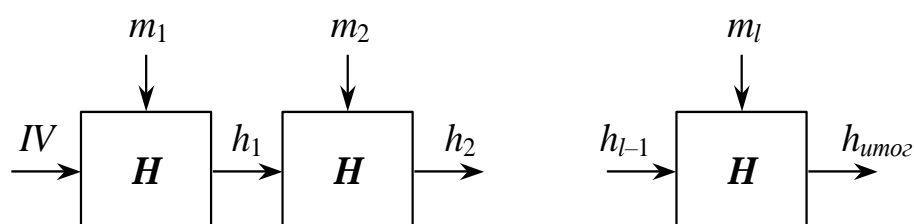


Рис. 2.2. Метод последовательного хэширования

Если необходимо хэшировать сообщение $m = (m_1, m_2, \dots, m_l)$, то хэширование выполняется следующим образом:

$$\begin{aligned}
 h_0 &= IV, \\
 h_i &= H(m_i, h_{i-1}), \text{ для } i=1, 2, \dots, l \\
 h_{\text{итого}} &= h_l.
 \end{aligned}$$

Здесь H – функция сжатия, а h_i – переменная сцепления.

Если последний блок меньше чем n бит, то он набивается одним из существующих методов до достижения длины кратной n . В отличие от стандартных предпосылок, что сообщение разбито на блоки и произведена набивка последнего блока, если необходимо, до начала хэширования, то в ГОСТ Р 34.11 – 94 процедура хэширования ожидает конца сообщения. Набивка производится следующим образом: последний блок сдвигается вправо, а затем набивается нулями до достижения длины в 256 бит (рисунок 2.3).

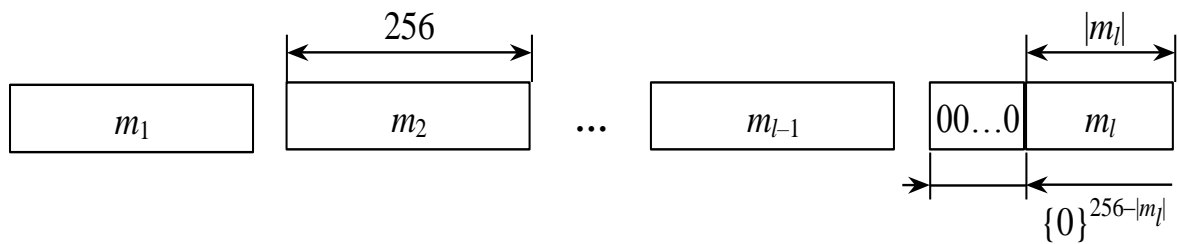
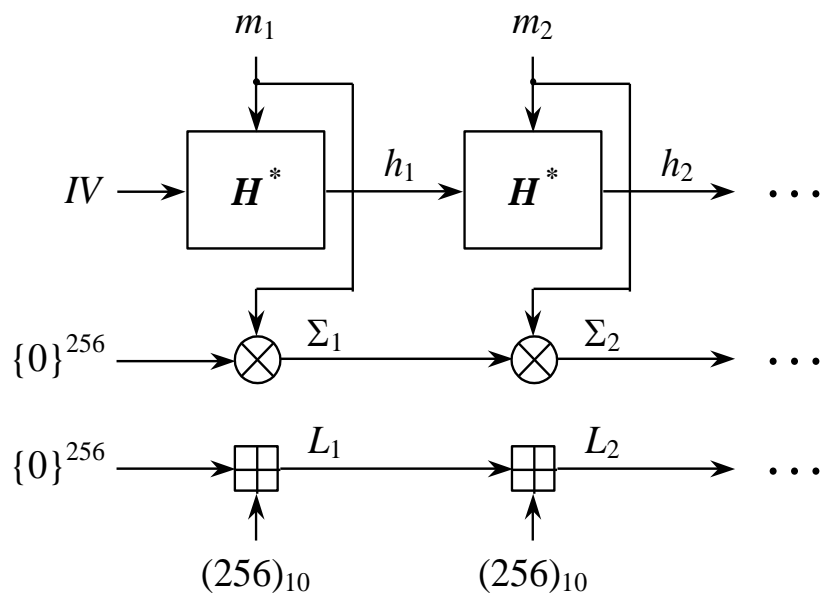


Рис. 2.3. Набивка сообщения

Указывать в передаваемом сообщении, сколько было добавлено нулей к последнему блоку, не требуется, так как длина сообщения участвует в хэшировании.

Параллельно рассчитываются контрольная сумма представляющая собой сумму всех блоков сообщения (последний суммируется уже набитым) по правилу $A+B \bmod 2^k$, где $k = |A|=|B|$, а $|A|$ и $|B|$ битовые длины слов A и B (далее на рисунках и в тексте эту операцию будем обозначать значком \otimes), и битовая длина хэшируемого сообщения приводимая по $\bmod 2^{256}$, которые в финальной функции сжатия используются для вычисления итогового хэша (рисунок 2.4) [2].



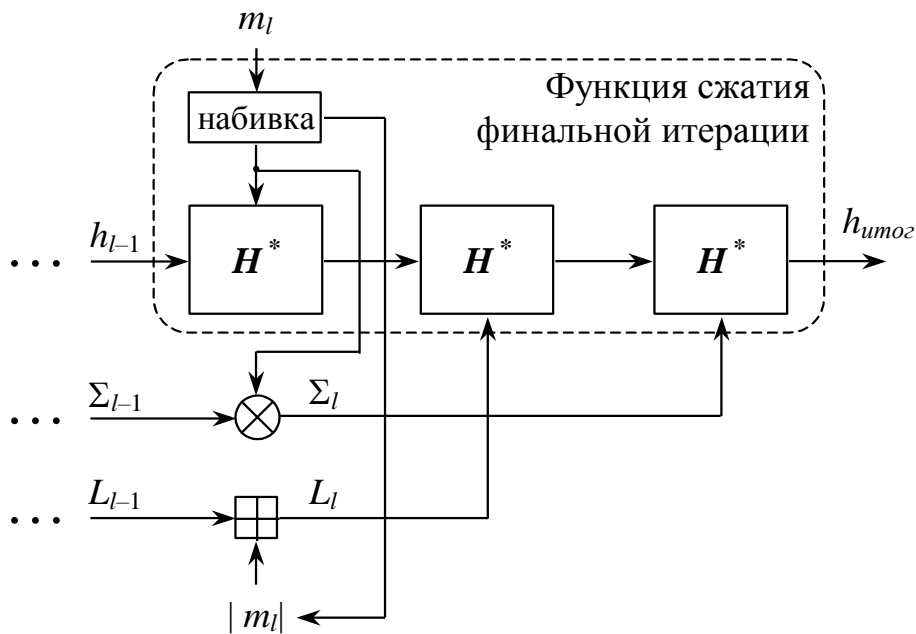


Рис. 2.4. Общая схема функции хэширования по ГОСТ Р 34.11 – 94

Согласно ГОСТ Р 34.11 – 94 IV – произвольное фиксированное слово длиной 256 бит ($IV \in \{01\}^{256}$). В таком случае, если он априорно не известен верифицирующему целостность сообщения, то он должен передаваться вместе с сообщением с гарантией целостности. При небольших сообщениях можно усложнить задачу противнику, если IV выбирается из небольшого множества допустимых величин (но при этом увеличивается вероятность угадывания хэш-величины противником). Также он может задаваться в рамках организации, домена как константа (обычно как в тестовом примере).

Функция сжатия внутренних итераций (по ГОСТ “шаговая функция хэширования”) H^* отображает два слова длиной 256 бит в одно слово длиной 256 бит:

$$H^*: \{01\}^{256} \times \{01\}^{256} \rightarrow \{01\}^{256},$$

и состоит из трех процедур (рисунок 2.5):

- генерирования ключей;
- перемешивающего преобразования;
- шифрующего преобразования.

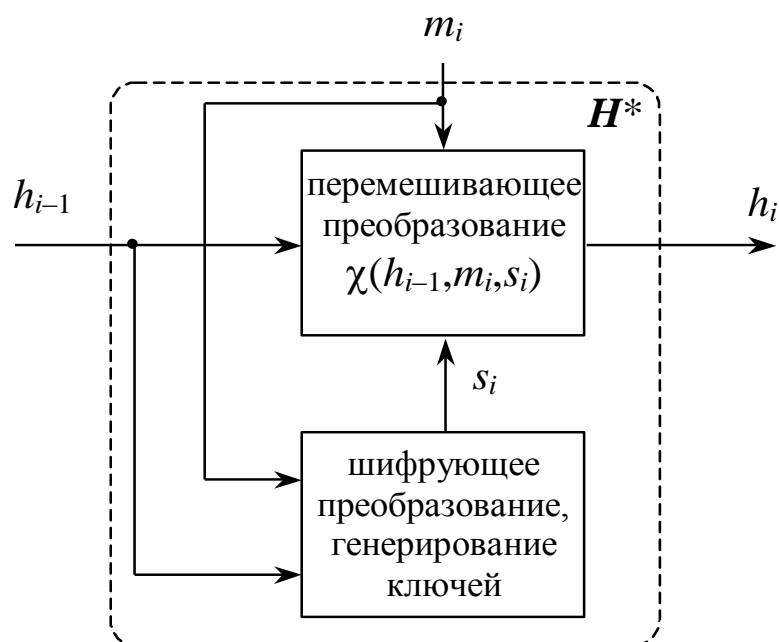


Рис. 2.5. Структура функции сжатия в ГОСТ Р 34.11 – 94

2.2.1. Генерирование ключей

Данная подпрограмма использует следующие функции и константы:

- Константы

$$C_2 = C_4 = \{0\}^{256} \text{ и } C_3 = 1^{80} 8^{16} 0^{24} 1^{16} 0^8 (0^8 1^8)^2 1^8 0^8 (0^8 1^8)^4 (1^8 0^8)^4,$$

где степень обозначает количество повторений 0 или 1.

- Функции

- $P: \{01\}^{256} \rightarrow \{01\}^{256}$.

Пусть $X = \xi_{32} \parallel \xi_{31} \parallel \dots \parallel \xi_1$, где \parallel – конкатенация,

тогда $P(X) = \xi_{\varphi(32)} \parallel \xi_{\varphi(31)} \parallel \dots \parallel \xi_{\varphi(1)}$, где $\varphi(i+1+4(k-1)) = 8i+k$, $i = 0 \dots 3$,

$k = 1 \dots 8$.

- $A: \{01\}^{256} \rightarrow \{01\}^{256}$.

Пусть $X = x_4 \parallel x_3 \parallel x_2 \parallel x_1$,

тогда $A(X) = (x_1 \oplus x_2) \parallel x_4 \parallel x_3 \parallel x_2$.

При вычислении ключей реализуется следующий алгоритм (рисунок 2.6):

1. $j = 1$, $U = m_i$, $V = h_{i-1}$;

2. $W = U \oplus V, K_1 = P(W)$;
3. $j = j + 1$;
4. Проверить условие $j = 5$. При положительном исходе перейти к шагу 7, при отрицательном – к шагу 5.
5. $U = A(U) \oplus C_j, V = A(A(V)), W = U \oplus V, K_j = P(W)$;
6. Перейти к шагу 3;
7. Выход (K_1, K_2, K_3, K_4) .

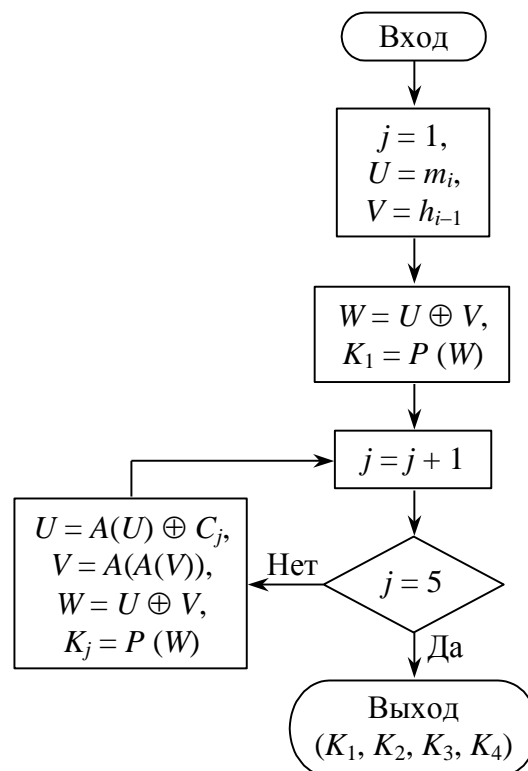


Рис. 2.6. Программа генерирования ключей

2.2.2. Шифрующее преобразование

Основным функциональным предназначением является получение s_i из h_{i-1} .

Пусть $h_{i-1} = h_{i-1}^4 \parallel h_{i-1}^3 \parallel h_{i-1}^2 \parallel h_{i-1}^1$, где $h_{i-1}^j \in \{01\}^{64}, j = 1 \dots 4$, а

$$s_i = s_i^4 \parallel s_i^3 \parallel s_i^2 \parallel s_i^1, \text{ где } s_i^j \in \{01\}^{64}, j = 1 \dots 4.$$

Тогда

$$s_i^j = E_{K_j}(h_{i-1}^j),$$

где $j = 1 \dots 4$, E_{K_j} – шифрование по ГОСТ 28147 – 89 в режиме простой замены.

Схематически это изображено на рисунке 2.7.

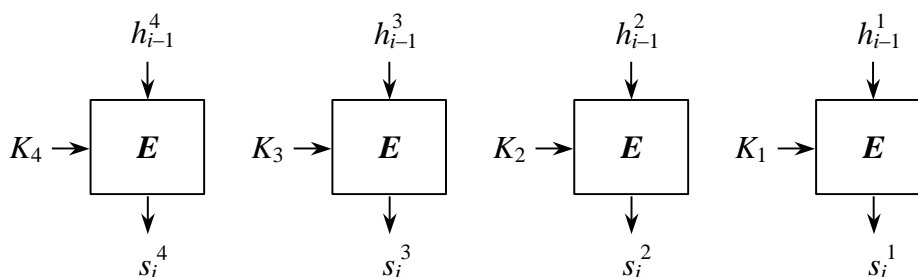


Рис. 2.7. Шифрующее преобразование

ГОСТ 28147-89 предусматривает три следующих режима шифрования данных [3]:

- простая замена,
- гаммирование,
- гаммирование с обратной связью,

и один дополнительный – режим выработки имитовставки.

В любом из этих режимов данные обрабатываются блоками по 64 бита, на которые разбивается массив, подвергаемый криптографическому преобразованию, именно поэтому ГОСТ 28147-89 относится к блочным шифрам.

Рассмотрим режим простой замены.

Оригинал ГОСТа 28147–89 содержит описание алгоритмов нескольких уровней. На самом верхнем находятся практические алгоритмы, предназначенные для шифрования массивов данных и выработки для них имитовставки. Все они опираются на три алгоритма низшего уровня – *базовые циклы*. В свою очередь, каждый из базовых циклов представляет собой многократное повторение одной единственной процедуры, называемой для

определенности далее в настоящей работе *основным шагом криптопреобразования.*

В соответствии с принципом Кирхгоффа, которому удовлетворяют все современные известные широкой общественности шифры, секретность ключевой информации обеспечивает секретность зашифрованного сообщения. В ГОСТ 28147-89 ключевая информация состоит из двух структур данных. Помимо собственно *ключа*, необходимого для всех шифров, она содержит еще и *таблицу замен*:

1. **Ключ** является массивом из восьми 32-битных элементов кода, далее в настоящей работе он обозначается символом K : $K = \{K_i\}_{0 \leq i \leq 7}$. В ГОСТ 28147-89 элементы ключа используются как 32-разрядные целые числа без знака: $0 \leq K_i < 2^{32}$. Таким образом, размер ключа составляет $32 \cdot 8 = 256$ бит или 32 байта.
2. **Таблица замен** является матрицей 8×16 , содержащей 4-битовые элементы, которые можно представить в виде целых чисел от 0 до 15. Строки *таблицы замен* называются *узлами замен*, они должны содержать различные значения, то есть каждый узел замен должен содержать 16 различных чисел от 0 до 15 в произвольном порядке. В настоящей статье таблица замен обозначается символом H : $H = \{H_{i,j}\}_{\substack{0 \leq i \leq 7 \\ 0 \leq j \leq 15}}$, $0 \leq H_{i,j} \leq 15$. Таким образом, общий объем таблицы замен равен: $8 \text{ узлов} \times 16 \text{ элементов/узел} \times 4 \text{ бита/элемент} = 512 \text{ бит}$ или 64 байта.

Основной шаг криптопреобразования по своей сути является оператором, определяющим преобразование 64-битового блока данных. Дополнительным параметром этого оператора является 32-битовый блок, в качестве которого используется какой-либо элемент ключа. Схема алгоритма основного шага приведена на рисунке 2.8.

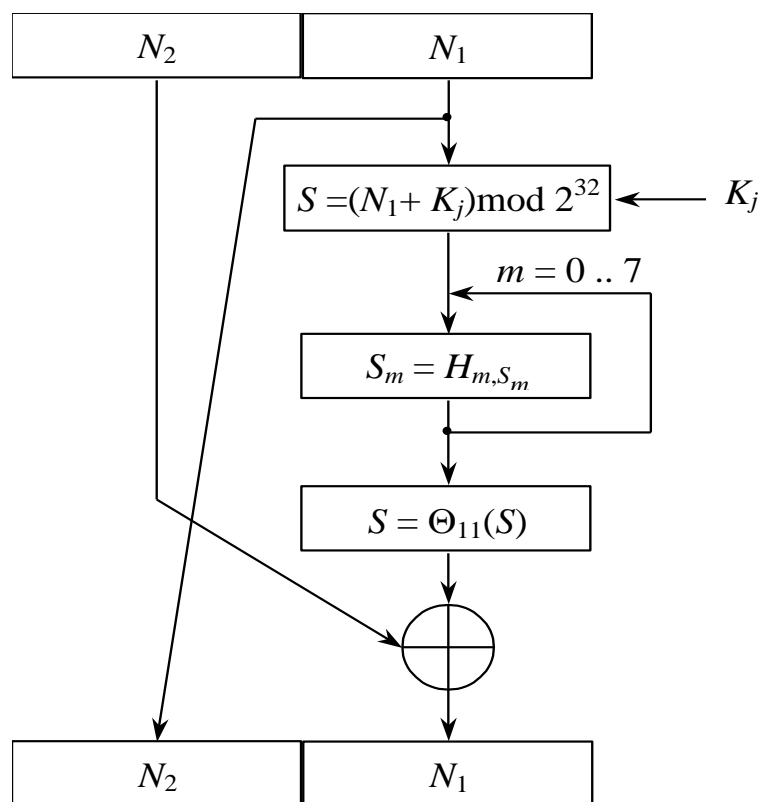


Рис. 2.8. Схема основного шага криптопреобразования алгоритма ГОСТ 28147-89

Ниже даны пояснения к алгоритму основного шага:

1. Определяет исходные данные для основного шага криптопреобразования:
 - N – преобразуемый 64-битовый блок данных. В ходе выполнения шага его младшая (N_1) и старшая (N_2) части обрабатываются как отдельные 32-битовые целые числа без знака. Таким образом, можно записать $N = (N_1, N_2)$.
 - K_j – 32-битовый элемент ключа;
2. Сложение с ключом: младшая половина преобразуемого блока складывается по модулю 2^{32} с используемым на шаге элементом ключа, результат передается на следующий этап;
3. Поблочная замена: 32-битовое значение, полученное на предыдущем этапе, интерпретируется как массив из восьми 4-битовых блоков кода:

$$S = (S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_7).$$

Далее значение каждого из восьми блоков заменяется на новое, которое выбирается по таблице замен следующим образом: значение блока S_i заменяется на S_j -ый по порядку элемент (нумерация с нуля) i -того узла замен (т.е. i -той строки таблицы замен, нумерация также с нуля). Другими словами, в качестве замены для значения блока выбирается элемент из таблицы замен с номером строки, равным номеру заменяемого блока, и номером столбца, равным значению заменяемого блока как 4-битового целого неотрицательного числа. Теперь становится понятным размер таблицы замен: число строк в ней равно числу 4-битных элементов в 32-битном блоке данных, то есть восьми, а число столбцов равно числу различных значений 4-битного блока данных, равному, как известно, 2^4 , шестнадцати.

4. Циклический сдвиг на 11 бит влево: результат предыдущего этапа сдвигается циклически на 11 бит в сторону старших разрядов и передается на следующий этап. На схеме алгоритма символом Θ_{11} обозначена функция циклического сдвига своего аргумента на 11 бит в сторону старших разрядов.
5. Побитовое сложение: значение S_j , полученное на предыдущем этапе, побитно складывается по модулю 2 со старшей половиной преобразуемого блока.
6. Сдвиг по цепочке: младшая часть преобразуемого блока сдвигается на место старшей, а на ее место помещается результат выполнения предыдущего этапа.
7. Полученное значение преобразуемого блока возвращается как результат выполнения алгоритма основного шага криптопреобразования.

ГОСТ 28147-89 относится к классу блочных шифров, то есть единицей обработки информации в нем является блок данных. Следовательно, вполне

логично ожидать, что в нем будут определены алгоритмы для криптографических преобразований, то есть для зашифрования, расшифрования и «учета» в контрольной комбинации одного блока данных. Именно эти алгоритмы и называются *базовыми циклами* ГОСТа 28147-89, что подчеркивает их фундаментальное значение для построения этого шифра.

Базовые циклы построены из *основных шагов* криптографического преобразования. В процессе выполнения основного шага используется только один элемент ключа, в то время как ключ ГОСТ 28147-89 содержит восемь таких элементов. Следовательно, чтобы ключ был использован полностью, каждый из базовых циклов должен многократно выполнять основной шаг с различными его элементами. Вместе с тем кажется вполне естественным, что в каждом базовом цикле все элементы ключа должны быть использованы одинаковое число раз, по соображениям стойкости шифра это число должно быть больше одного. Поэтому базовые циклы заключаются в многократном выполнении *основного шага* с использованием разных элементов ключа и отличаются друг от друга только числом повторения шага и порядком использования ключевых элементов. Ниже приведен этот порядок для различных циклов.

1. Цикл зашифрования 32-З:

$K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0.$

2. Цикл расшифрования 32-Р:

$K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0, K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0, K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0.$

Каждый из циклов имеет собственное буквенно-цифровое обозначение, соответствующее шаблону «*n-X*», где первый элемент обозначения (*n*), задает число повторений основного шага в цикле, а второй элемент обозначения (*X*), буква, задает порядок зашифрования («З») или расшифрования («Р») в

использовании ключевых элементов. Этот порядок нуждается в дополнительном пояснении.

Цикл расшифрования должен быть обратным циклу зашифрования, то есть последовательное применение этих двух циклов к произвольному блоку должно дать в итоге исходный блок, что отражается следующим соотношением: $C_{32-P}(C_{32-Z}(T))=T$, где T – произвольный 64-битный блок данных, $C_X(T)$ – результат выполнения цикла X над блоком данных T . Для выполнения этого условия для алгоритмов, подобных ГОСТ 28147-89, необходимо и достаточно, чтобы порядок использования ключевых элементов соответствующими циклами был взаимно обратным. В справедливости записанного условия для рассматриваемого случая легко убедиться, сравнив приведенные выше последовательности для циклов 32-З и 32-Р. Из сказанного вытекает одно интересное следствие: свойство цикла быть обратным другому циклу является взаимным, то есть цикл 32-З является обратным по отношению к циклу 32-Р. Другими словами, зашифрование блока данных теоретически может быть выполнено с помощью цикла расшифрования, в этом случае расшифрование блока данных должно быть выполнено циклом зашифрования. Из двух взаимно обратных циклов любой может быть использован для зашифрования, тогда второй должен быть использован для расшифрования данных, однако стандарт ГОСТ 28147-89 закрепляет роли за циклами и не предоставляет пользователю права выбора в этом вопросе.

Схемы базовых циклов приведены на рисунке 2.9. Каждый из них принимает в качестве аргумента и возвращает в качестве результата 64-битный блок данных, обозначенный на схемах N .

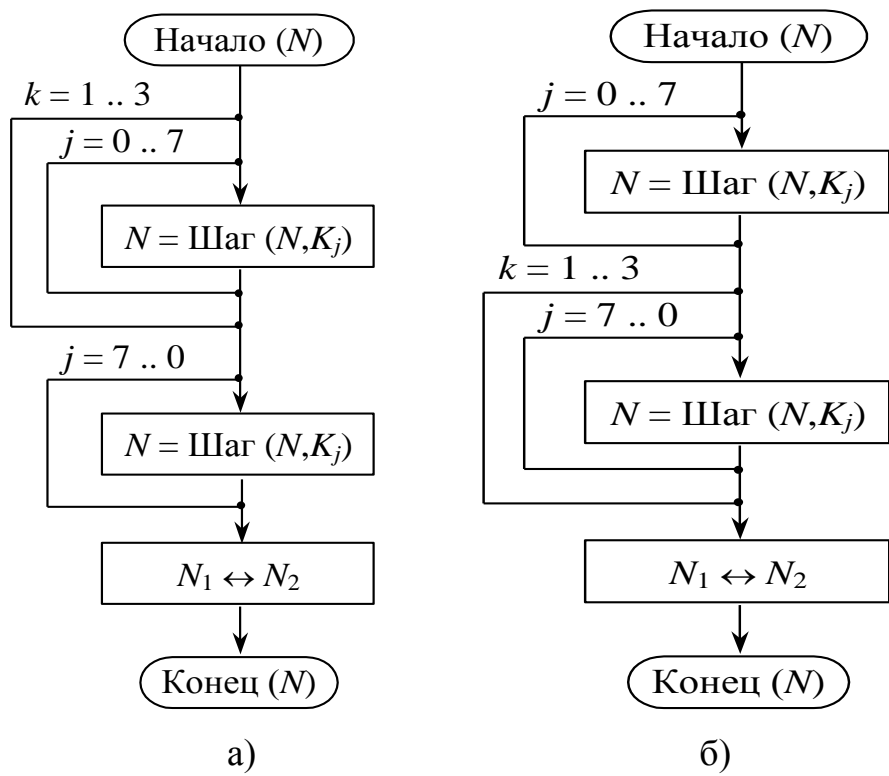


Рис. 2.9. Схемы цикла зашифрования 32-3 (а) и расшифрования 32-Р(б)

Символ Шаг(N, K_j) обозначает выполнение основного шага криптопреобразования для блока N с использованием ключевого элемента X .

В конце базовых циклов шифрования старшая и младшая часть блока результата меняются местами, это необходимо для их взаимной обратимости.

Зашифрование в режиме простой замены заключается в применении цикла 32-3 к блокам открытых данных, расшифрование – цикла 32-Р к блокам зашифрованных данных. Это наиболее простой из режимов, 64-битовые блоки данных обрабатываются в нем независимо друг от друга. Схемы алгоритмов зашифрования и расшифрования в режиме простой замены приведены на рисунке 2.10, они тривиальны и не нуждаются в комментариях.

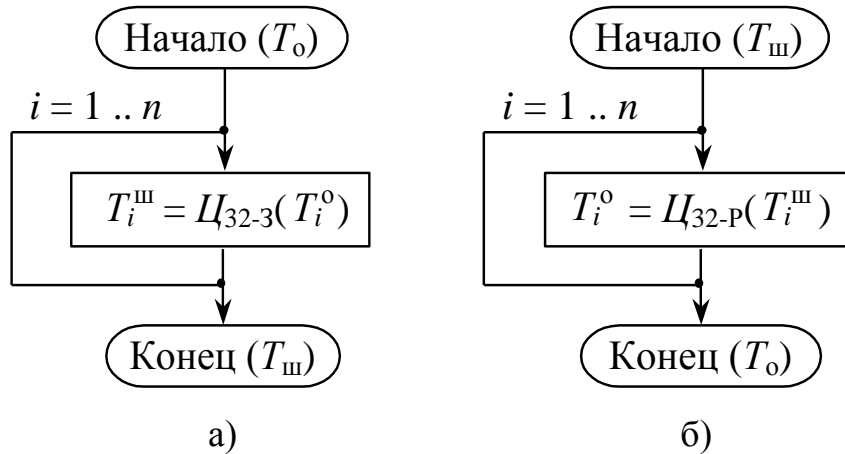


Рис. 2.10. Алгоритмы зашифрования (а) и расшифрования (б) данных в режиме простой замены

Размер массива открытых или зашифрованных данных, подвергющийся соответственно зашифрованию или расшифрованию, должен быть кратен 64 битам: $|T_o|=|T_{\text{ш}}|=64 \cdot n$, после выполнения операции размер полученного массива данных не изменяется.

2.2.3. Перемешивающее преобразование

Перемешивающее преобразование имеет вид

$$h_i = \chi(m_i, h_{i-1}, s_i) = \psi^{61}(h_{i-1} \oplus \psi(m_i \oplus \psi^{12}(s_i))),$$

где ψ^j – j -я степень преобразования ψ .

Схематически данное преобразование представлено на рисунке 2.11.

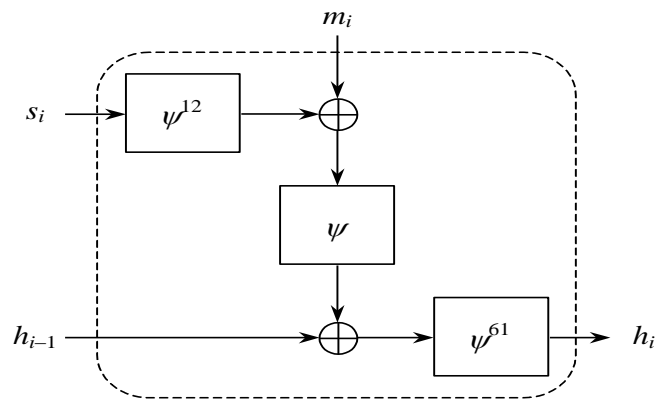


Рис. 2.11. Перемешивающее преобразование ГОСТ Р 34.11 – 94

Причем s_i выводится из h_{i-1} (рисунок 2.5).

Функция $\psi: \{01\}^{256} \rightarrow \{01\}^{256}$ преобразует слово $\eta_{16} \parallel \eta_{15} \parallel \dots \parallel \eta_1 \in \{01\}^{16}$, $i = 1 \dots 16$ в слово $\eta_1 \oplus \eta_2 \oplus \eta_3 \oplus \eta_4 \oplus \eta_{13} \oplus \eta_{16} \parallel \eta_{15} \parallel \dots \parallel \eta_2$ (рисунок 2.12).

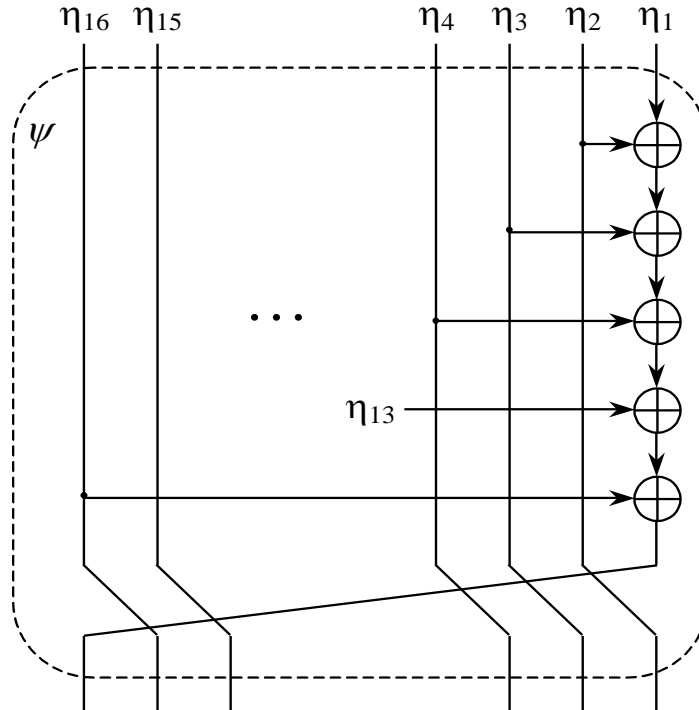


Рис. 2.12. Функция ψ перемешивающего преобразования

Функция сжатия финальной итерации производит итоговую хэш-величину, зависящую от результата хэширования последовательным методом, контрольной суммы по $\text{mod } 2$ и длины сообщения в битах приведенной по $\text{mod } 2^{256}$:

$$m'_i \times h_{l-1} \times \sum_l \times L_l \rightarrow h_{\text{итог}},$$

где $m'_i, h_{l-1}, \sum_l, L_l \in \{01\}^{256}$, m'_i – последний набитый нулями (если необходимо) блок.

Сначала вычисляется битовая длина последнего (ненабитого) блока сообщения $|m_i| \leq 256$ бит. Если $|m_i| < 256$, то производится его набивка справа нулями до достижения длины 256 бит и получается новый блок $m'_i = \{0\}^{256-|m_i|} \parallel m_i$. Вычисляются итоговая контрольная сумма $\sum_l = \sum_{l-1} \otimes m'_i$ и длина всего

сообщения $L_l = L_{l-1} + |m'| \pmod{2^{256}}$. Параллельно выполняется последняя итерация $h_l = H^*(m', h_{l-1})$. Затем вычисляются перемешивающие преобразования $h_{l+1} = H^*(L_l, h_l)$ и $h_{итог} = H^*(\Sigma_l, h_{l+1})$, давая в результате итоговую хэш-величину $h_{итог}$.

Теперь дадим формальное описание алгоритма вычисления хэш-функции ГОСТ Р 34.11 – 94:

Вход: двоичное сообщение M , блоки замен для шифрования в режиме простой замены по ГОСТ 28147-89, начальный вектор $IV \in \{01\}^{256}$.

Выход: хэш-величина $h_{итог}$ для сообщения M .

1. *Инициализация алгоритма:* $h = IV, \Sigma = \{0\}^{256}, L = \{0\}^{256}$.
2. *Функция сжатия внутренних итераций:* Пока $|M| > 256$ выполняем следующее:
 - 2.1. $h = H^*(m_s, h)$ (итерация метода последовательного хэширования);
 - 2.2. $L = L + 256 \pmod{2^{256}}$ (итерация вычисления длины сообщения);
 - 2.3. $\Sigma = \Sigma \otimes m_s$ (итерация вычисления контрольной суммы).
3. *Иначе (функция сжатия финальной итерации)*
 - 3.1. $L = L + |m| \pmod{2^{256}}$ (вычисление полной длины сообщения);
 - 3.2. $m' = \{0\}^{256-|m|} || m$ (набивка последнего блока);
 - 3.3. $\Sigma = \Sigma \otimes m'$ (вычисление контрольной суммы сообщения);
 - 3.4. $h = H^*(m', h)$;
 - 3.5. $h = H^*(L, h)$;
 - 3.6. $h_{итог} = H^*(\Sigma, h)$.
4. *Выход* ($h_{итог}$)

Данная хэш-функция определена стандартом ГОСТ Р 34.11-94 для использования совместно с российским стандартом электронной цифровой подписи.

2.3. ЭЦП и эллиптические кривые

Абсолютное большинство разрабатываемых и используемых в мире ЭЦП базируется на использовании несимметричных криптографических преобразований, выполняемых в кольцах (RSA), полях Галуа (DSA, ГОСТ Р 34.10-94) и группах точек эллиптических кривых (ECDSA, ГОСТ Р 34.10-2001). Опыт применения и проведения исследований ЭЦП, базирующихся на преобразованиях в кольцах и полях, показали, что они практически исчерпали себя. Для обеспечения требуемой стойкости, при использовании данных ЭЦП, необходимо увеличивать длину ключа цифровых подписей. Однако увеличение длины ключа повышает требование этих криптосистем к вычислительным возможностям ЭВМ, что не всегда является приемлемым (не все организации в состоянии поменять весь парк компьютеров для осуществления приемлемого уровня быстродействия обновленных алгоритмов электронной подписи). Для разрешения этого противоречия разработаны и начали внедряться новые или модифицированные криптографические преобразования, выполняемые в группах точек эллиптических кривых.

Использование математического аппарата эллиптических групп позволяет уменьшить длину ключа, что в свою очередь позволяет повысить скорость алгоритма формирования и проверки подписи.

2.3.1. Эллиптические кривые

Эллиптические кривые применяются в криптографии с 1985 года. Интерес к ним обусловлен тем, что на их основе обеспечиваются те же криптографические свойства, которыми обладают числовые или полиномиальные криптосистемы, но при существенно меньшем размере ключа.

Пусть задано простое число $p > 3$. Тогда *эллиптической кривой* E , определённой над конечным простым полем F_p , называется кривая, которая задается уравнением вида

$$y^2 \equiv x^3 + ax + b \pmod{p}, \quad (6.1)$$

где $a, b \in F_p$, и $4a^3 + 27b^2$ не сравнимо с нулем по модулю p .

Эллиптическая кривая E состоит из всех точек (x, y) , $x, y \in F_p$, удовлетворяющих тождеству (6.1), и бесконечно удаленной точки O , которая называется *нулевой точкой*.

Точки эллиптической кривой будем обозначать $Q = (x, y)$ или просто Q . Две точки эллиптической кривой равны, если равны их соответствующие x - и y -координаты.

Инвариантом эллиптической кривой называется величина $J(E)$, удовлетворяющая тождеству

$$J(E) \equiv 1728 \cdot \frac{4a^3}{4a^3 + 27b^2} \pmod{p}$$

Коэффициенты a, b эллиптической кривой E , по известному инварианту $J(E)$, определяются следующим образом

$$\begin{cases} a \equiv 3k \pmod{p}, \\ b \equiv 2k \pmod{p}, \end{cases} \text{ где } k \equiv \frac{J(E)}{1728 - J(E)} \pmod{p}, J(E) \neq 0 \text{ или } 1728.$$

Таким образом, эллиптическая кривая E может быть однозначно задана либо своим инвариантом $J(E)$, либо коэффициентами $a, b \in F_p$.

Введем операцию сложения на множестве всех точек эллиптической кривой E .

Пусть E – эллиптическая кривая, определённая над конечным простым полем F_p и заданная уравнением (6.1). Пусть P и Q – две точки на E . Определим сумму $(P + Q)$ по следующим правилам:

1. Если P выступает в роли нулевого элемента, т.е. $P = O$, то для любой точки Q на эллиптической кривой $Q + O = Q$.
2. Отрицание $-P$ – это точка с той же x -координатой, но с отрицанием y -координаты, т.е. $-(x, y) = (x, -y)$. Если $Q = -P$, то определяем сумму $Q + P = O$.

3. Если P и Q имеют различные x -координаты, то для их сложения следует провести через эти точки прямую и найти точку R пересечения ее с эллиптической кривой (рисунок 2.13).

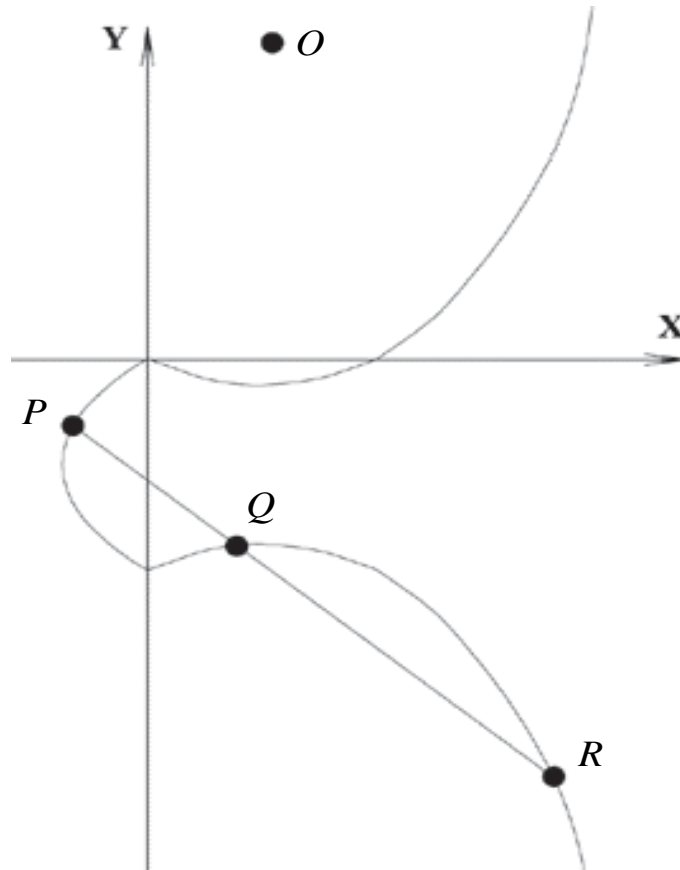


Рис. 2.13. Сложение точек на эллиптической кривой

Если эта прямая не является касательной к кривой в точках P или Q , то существует только одна такая точка R . Сумма любых трех точек эллиптической кривой, лежащих на одной прямой, равна O : $P + Q + R = O$. Используя второе правило, получим $P + Q = -R$. Если прямая является касательной к кривой в точке P или Q , то полагаем $R = P$ или $R = Q$, соответственно.

Если $P = (x_1, y_1)$, $Q = (x_2, y_2)$, то сумма $P + Q = (x_3, y_3)$ определяется по следующим формулам:

$$\begin{cases} x_3 \equiv \lambda^2 - x_1 - x_2 \pmod{p}, \\ y_3 \equiv \lambda(x_1 - x_3) - y_1 \pmod{p}, \end{cases}$$

где $\lambda \equiv \frac{y_2 - y_1}{x_2 - x_1} \pmod{p}$ – угловой коэффициент прямой, проведенной через точки P и Q .

4. Последний вариант – это $P = Q$.

Чтобы удвоить точку P , следует провести касательную в точке P и найти другую точку R пересечения с эллиптической кривой. Тогда $P + P = 2P = -R$.

Если $P = (x_1, y_1)$, то $2P = (x_3, y_3)$ определяется по следующим формулам:

$$\begin{cases} x_3 \equiv \lambda^2 - 2x_1 \pmod{p}, \\ y_3 \equiv \lambda(x_1 - x_3) - y_1 \pmod{p}, \end{cases}$$

где $\lambda \equiv \frac{3x_1^2 + a}{2y_1} \pmod{p}$.

Введенная таким образом операция сложения подчиняется всем обычным правилам сложения, в частности коммутативному и ассоциативному законам.

Точка Q называется *точкой кратности k* , или просто – *кратной точкой* эллиптической кривой E , если для некоторой точки P выполнено равенство:

$$Q = \underbrace{P + \dots + P}_k = kP.$$

2.3.2. Российский стандарт ЭЦП ГОСТ Р 34.10-2001

Новый отечественный стандарт цифровой подписи обозначается как ГОСТ Р 34.10-2001. В нем используются следующие параметры:

- простое число p – модуль эллиптической кривой, удовлетворяющее неравенству $p > 2^{255}$. Верхняя граница данного числа должна определяться при конкретной реализации схемы цифровой подписи;
- эллиптическая кривая E , задаваемая своим инвариантом $J(E)$ или коэффициентами $a, b \in F_p$;

- целое число m – порядок группы точек эллиптической кривой E ;
- простое число q – порядок циклической подгруппы группы точек эллиптической кривой E , для которого выполнены следующие условия:

$$\begin{cases} m = nq, & n \in \mathbb{Z}, \quad n \geq 1, \\ 2^{254} < q < 2^{256}; \end{cases} \quad (3.26)$$

- точка $P \neq O$ эллиптической кривой E , с координатами (x_p, y_p) , удовлетворяющая равенству $qP = O$.

Каждый пользователь схемы цифровой подписи должен обладать личными ключами:

- ключом подписи – целым числом d , удовлетворяющим неравенству $0 < d < q$;
- ключом проверки – точкой эллиптической кривой Q с координатами (x_q, y_q) , удовлетворяющей равенству $dP = Q$.

На приведенные выше параметры схемы цифровой подписи накладываются следующие требования:

- должно быть выполнено условие $p^t \neq 1 \pmod{q}$, для всех целых $t = 1, 2, \dots, B$, где B удовлетворяет неравенству $B \geq 31$;
- должно быть выполнено неравенство $m \neq p$;
- инвариант кривой должен удовлетворять условию $J(E) \neq 0$ или 1728.

Для получения цифровой подписи под сообщением M необходимо выполнить следующие действия (рисунок 2.14):

1. вычислить хэш-значение сообщения M : $m = h(M)$;
2. определить $e \equiv m \pmod{q}$. Если $e = 0$, то определить $e = 1$;
3. сгенерировать случайное (псевдослучайное) целое число k , удовлетворяющее неравенству $0 < k < q$;

4. вычислить точку эллиптической кривой $kP = (x_1, y_1)$ и определить $r \equiv x_1 \pmod{q}$. Если $r = 0$, то вернуться к шагу 3;
5. вычислить значение $s \equiv (ke + rd) \pmod{q}$. Если $s = 0$, то вернуться к шагу 3.

Подпись для сообщения M составит пара чисел (r, s) .

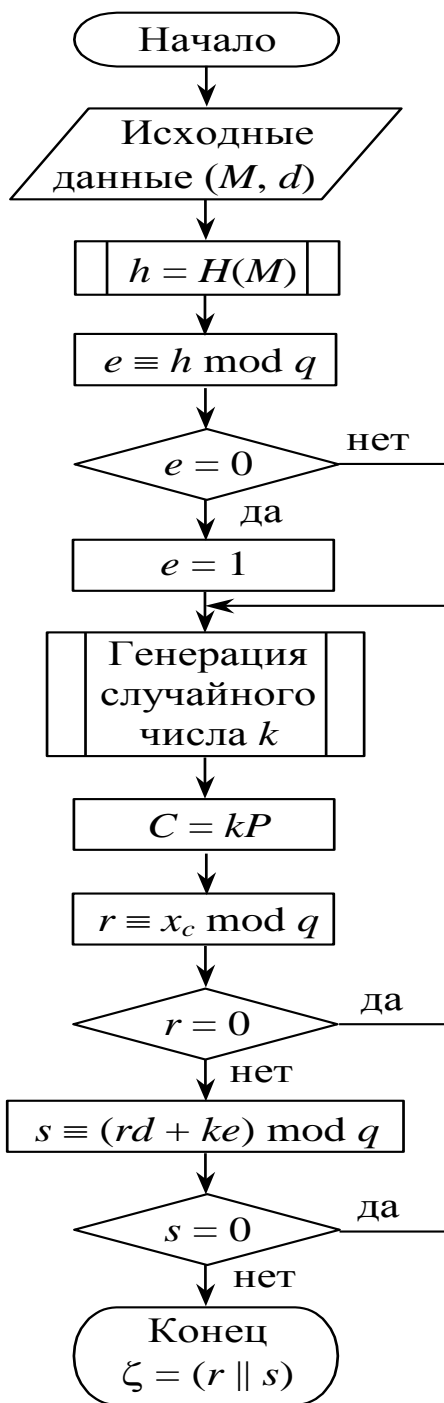


Рис. 2.14. Процесс формирования цифровой подписи ГОСТ Р 34.10-2001

Для проверки цифровой подписи под полученным сообщением M необходимо выполнить следующие действия (рисунок 2.15):

1. Если выполнены неравенства $0 < r < q$, $0 < s < q$, то перейти к следующему шагу. В противном случае подпись неверна;
2. вычислить хэш-значение полученного сообщения M : $m = h(M)$;
3. определить $e \equiv m \pmod{q}$. Если $e = 0$, то определить $e = 1$;
4. вычислить значение $w = e^{-1} \pmod{q}$;
5. вычислить значения $u_1 \equiv sw \pmod{q}$, $u_2 \equiv -rw \pmod{q}$;
6. вычислить точку эллиптической кривой $u_1P + u_2Q = (x_1, y_1)$ и определить $v \equiv x_1 \pmod{q}$;
7. если выполнено равенство $R = r$, то подпись принимается, в противном случае, подпись неверна.

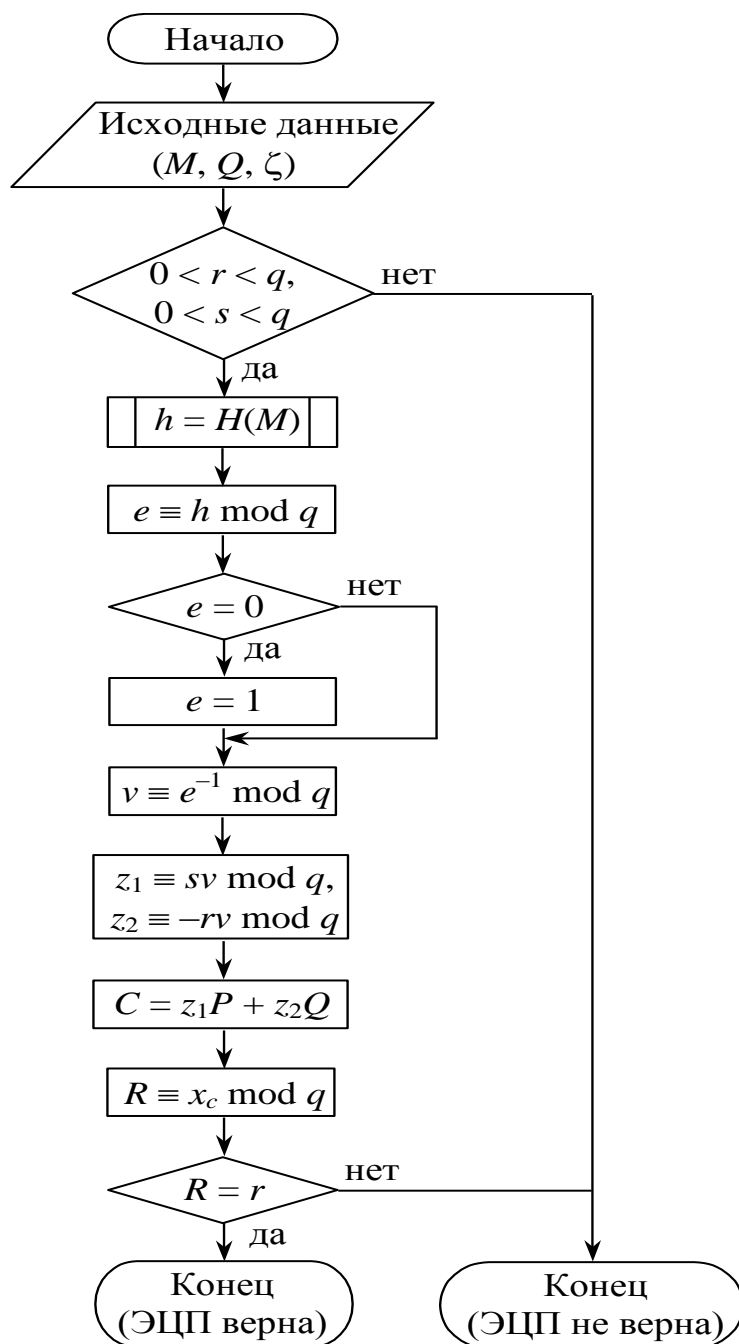


Рис. 2.15. Процесс проверки цифровой подписи ГОСТ Р 34.10-2001

2.4. Интерфейс учебного программного комплекса

Главное окно учебного программного комплекса (рисунок 2.16) содержит меню, состоящее из нескольких пунктов:

- ВЫЧИСЛЕНИЯ;
- СПРАВКА;
- ВЫХОД.

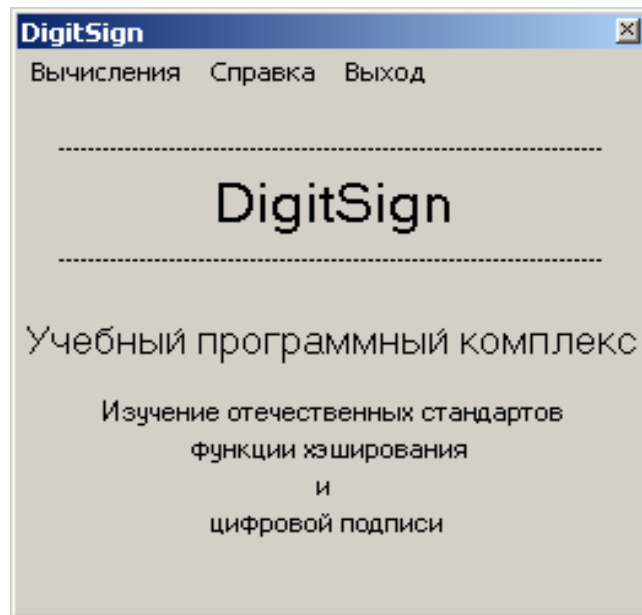


Рис. 2.16. Интерфейс главного окна учебного программного комплекса

Пункт меню “*Вычисления*” содержит следующие опции:

- **Вычисление хэш-функции.**

При выборе данной опции открывается окно “Вычисление хэш-функции” (рисунок 2.17), которое позволяет определить хэш-значение произвольного файла при заданных пользователем начальном хэш-векторе и таблице замен.

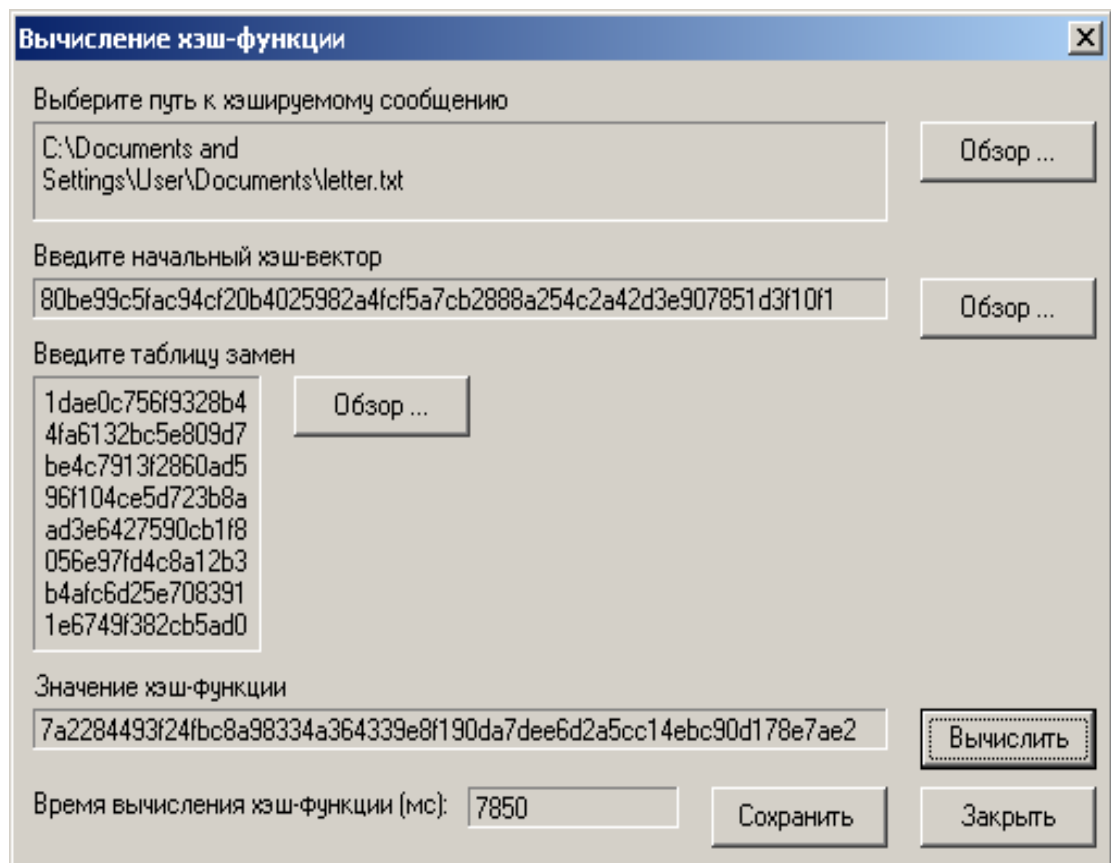


Рис. 2.17. Окно “Вычисление хэш-функции”

Полученный результат может быть сохранен в файл. Также пользователь получает информацию о времени хэширования сообщения.

- Генератор псевдослучайных чисел.
При выборе данной опции открывается окно “Генератор псевдослучайных чисел” (рисунок 2.18).

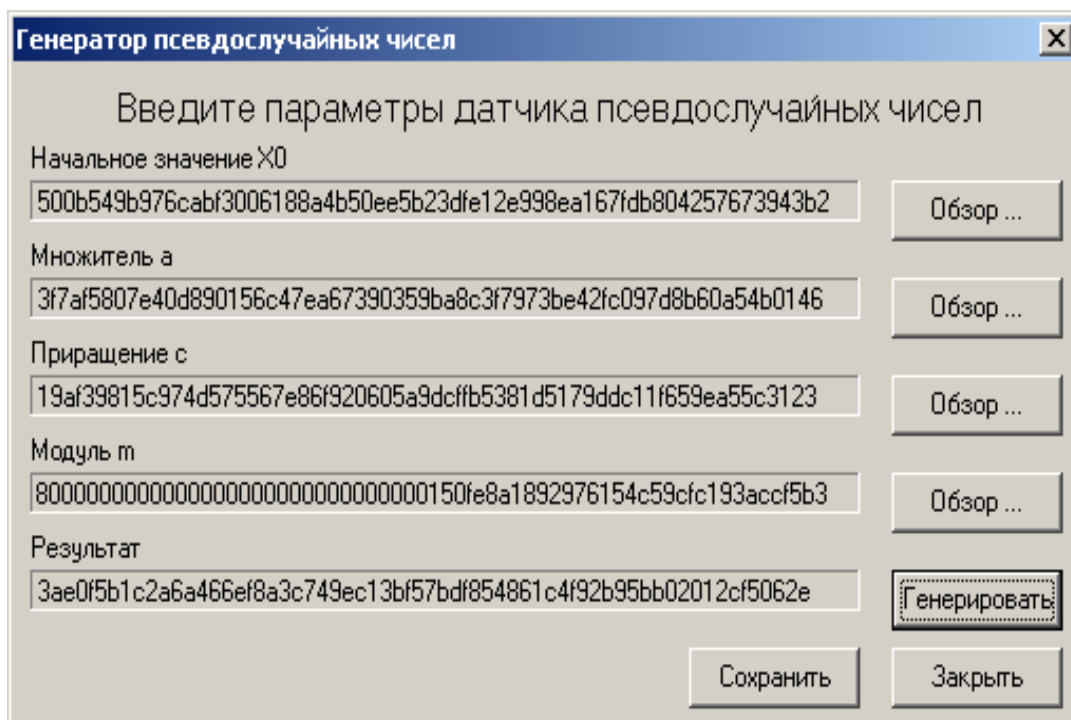


Рис. 2.18. Окно “Генератор псевдослучайных чисел”

Пользователь имеет возможность ввести параметры своего генератора, произвести вычисления и сохранить полученный результат в файл. В качестве генератора псевдослучайных чисел используется функция вида

$$X_{n+1} = (a \cdot X_n + c) \bmod m, n \geq 0,$$

где X_0 – начальное значение, $X_0 \geq 0$,

a – множитель, $a \geq 0$,

c – приращение, $c \geq 0$

m – модуль, $m > X_0$, $m > a$, $m > c$,

n – порядковый номер очередного целого числа, порождаемого данной функцией.

Она называется линейной конгруэнтной (совпадающей) последовательностью.

- Вычисление открытого ключа.

При выборе данной опции открывается окно “Вычисление открытого ключа” (рисунок 2.19).

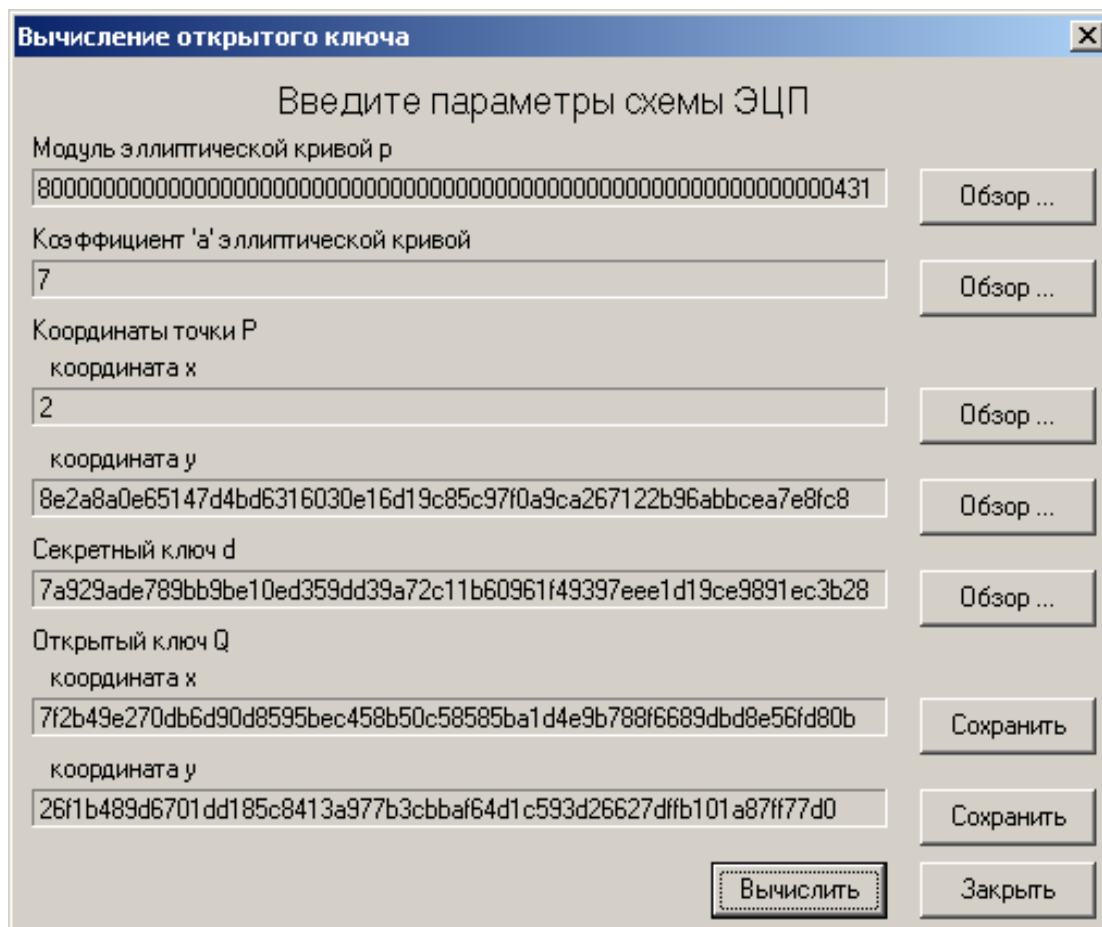


Рис. 2.19. Окно “Вычисление открытого ключа”

Пользователь имеет возможность вычислить открытый ключ для имеющегося у него секретного ключа и параметров схемы ЭЦП.

- Формирование ЭЦП.
При выборе данной опции открывается окно “Формирование ЭЦП” (рисунок 2.20).

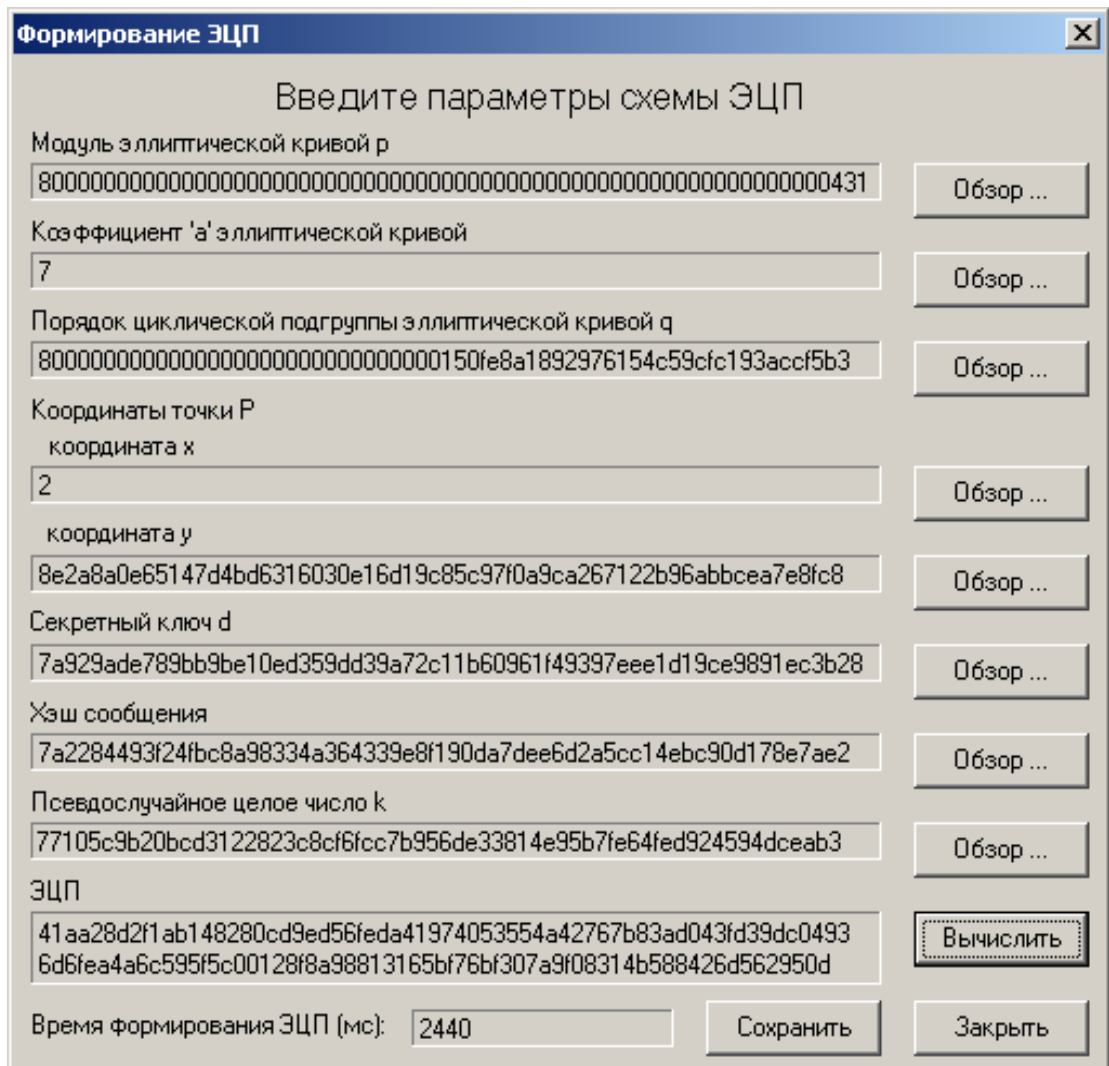


Рис. 2.20. Окно “Формирование ЭЦП”

Пользователь имеет возможность, после ввода всех необходимых для расчетов параметров, вычислить цифровую подпись и сохранить полученный результат в файл.

- Проверка ЭЦП.

При выборе данной опции открывается окно “Проверка ЭЦП” (рисунок 2.21).

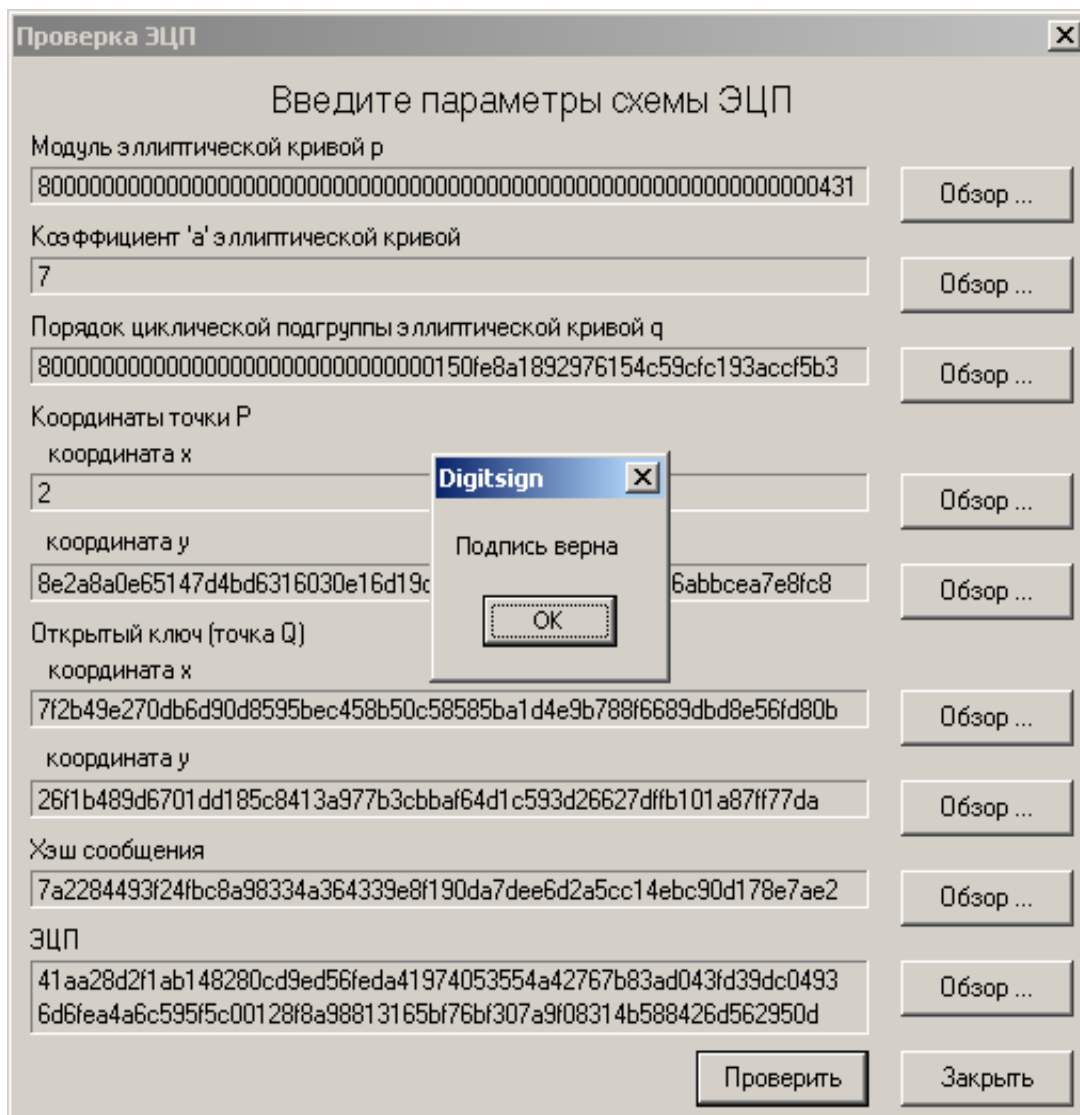


Рис. 2.21. Окно “Проверка ЭЦП”

Пользователь имеет возможность, после ввода всех необходимых для расчетов параметров, проверить подлинность цифровой подписи.

Пункт меню “Справка” содержит следующие опции:

- Задание на лабораторную работу.
Данная опция позволяет открыть файл справки, содержащий задание на лабораторную работу.
- Справка.
Данная опция позволяет открыть файл справки, содержащий информацию о хэш-функции и цифровых подписях.

- О программе.

Пункт меню “Выход” позволяет выйти из программы.

В каталоге с программой расположены тестовые значения параметров псевдослучайного генератора (X_0, a, c, m), хэш-функции (начальный хэш-вектор, таблица замен) и схемы ЭЦП (p, a, P, q), которые необходимо использовать при выполнении лабораторной работы.

3. Задание на лабораторную работу

1. Изучение функции хэширования ГОСТ Р 34.11-94.

1.1. Ознакомьтесь с теорией по функциям хэширования.

1.2. Откройте окно “Вычисление хэш-функции”.

1.3. Задайте начальные данные для вычисления хэш-функции:

- путь к файлу, который подвергается сжатию;
- начальный хэш-вектор;
- таблицу замен.

1.4. Вычислите хэш-функцию.

1.5. Повторите пункты 1.3 – 1.4 для файлов размером от 0,1 до 5 Мбайт.

Определите время хэширования. Полученные результаты занесите в первые две строки таблицы.

l , Мбайт					
t_h , мс					
$t_{ЭЦП}$, мс					
t_{Σ} , мс					

1.6. По полученным результатам постройте график зависимости времени вычисления хэш-значения от размера файла $t_h = f(l)$.

1.7. Сделайте выводы по полученным результатам.

2. Изучение цифровой подписи ГОСТ Р 34.10-2001.

2.1. Ознакомьтесь с теорией по цифровым подписям.

2.2. Вычислите секретный ключ.

2.2.1. Откройте окно “Программный датчик случайных чисел”.

2.2.2. Введите параметры датчика:

- начальное значение X_0 ;
- множитель a ;
- приращение c ;
- модуль m .

2.2.3. Вычислите секретный ключ.

2.2.4. Сохраните полученное значение в файл.

2.3. Вычислите открытый ключ.

2.3.1. Откройте окно “Вычисление открытого ключа”.

2.3.2. Задайте параметры, необходимые для вычисления.

2.3.3. Вычислите открытый ключ.

2.3.4. Сохраните полученные результаты в файл.

2.4. Вычислите цифровую подпись.

2.4.1. Выполните предварительное вычисление случайного числа k , необходимого для формирования цифровой подписи. Для этого следует выполнить действия, описанные в п. 2.2.

2.4.2. Откройте окно “Формирование цифровой подписи”.

2.4.3. Задайте параметры, необходимые для вычисления.

2.4.4. Сформируйте цифровую подпись.

2.4.5. Сохраните полученные результаты в файл.

2.5. Передайте студенту, с которым вы работаете в паре, свой открытый ключ, файл и соответствующую ему цифровую подпись при помощи дискеты или сети и получите от него такой же набор.

2.6. Проверьте цифровую подпись.

2.6.1. Откройте окно “Проверка цифровой подписи”.

2.6.2. Задайте параметры, необходимые для вычисления.

2.6.3. Осуществите проверку.

- 2.7. Сформируйте цифровые подписи для хэш-значений, вычисленных в п. 1.5. Определите время формирования ЭЦП. Полученные результаты занесите в третью строку таблицы.
 - 2.8. Рассчитайте суммарное время вычисления ЭЦП и занесите результаты расчетов в таблицу.
 - 2.9. По полученным результатам постройте графики зависимостей $t_{\text{ЭЦП}} = f(l)$ и $t_{\Sigma} = f(l)$ на одном графике с $t_h = f(l)$.
 - 2.10. Сделайте выводы по полученным результатам.
3. Оформите отчет по проделанной работе.

4. Контрольные вопросы

1. Что такое асимметричная криптографическая система? В чем ее преимущества перед симметричной системой? В чем недостатки?
2. Что такое хэш-функция? Какими свойствами она должна обладать и в чем они заключаются?
3. Опишите процесс формирования хэш-функции ГОСТ Р 34.11-94.
4. Что такое “функция сжатия внутренних итераций”? В чем она заключается?
5. Что такое цифровая подпись? В чем ее отличие от рукописной подписи?
6. Опишите процесс формирования цифровой подписи ГОСТ Р 34.10-2001.
7. Опишите процесс проверки цифровой подписи ГОСТ Р 34.10-2001.
8. От каких злоумышленных действий позволяет защититься ЭЦП?

5. Рекомендуемая литература

1. Романец Ю.В., Тимофеев П.А., Шаньгин В.Ф. Защита информации в компьютерных системах и сетях. – М.: Радио и связь, 2001. – 376 с.

2. Шефановский Д.Б ГОСТ Р 34.11-94. Функция хэширования. Краткий анализ // http://ssl.stu.neva.ru/psw/crypto/GOST_R_34.11-94_Pub.pdf
3. Винокуров А. Алгоритм шифрования ГОСТ 28147-89, его использование и реализация для компьютеров платформы Intel x86 // http://www.enlight.ru/crypto/download/articles/vinokurov/28147_89.zip
4. Алферов А.П., Зубов А.Ю., Кузьмин А.С., Черемушкин А.В. Основы криптографии – М.: Гелиос АРВ, 2001. – 480 с.
5. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. – М.: ТРИУМФ, 2002 – 816 с.

Лабораторная работа 6. Исследование алгоритмов защиты данных с помощью PGP

1. Цель работы

Целью данной лабораторной работы является получение практических навыков по управлению ключами, шифрованию, расшифрованию, электронному подписанию документов/файлов и их безвозвратному удалению с диска посредством криптографической программы Pretty Good Privacy (PGP).

2. Методические указания

Все пользователи Интернета должны отчётливо понимать, что отправка обычного незашифрованного электронного послания аналогична отправке открытки почтой неэлектронной: такое сообщение может быть прочитано кем угодно и где угодно на участке между отправителем и получателем даже без всякой нужды осуществлять его целенаправленный перехват. Копия сообщения остаётся в кэше сервера вашего Интернет-провайдера, сетевые серверы у вас на работе, в университете или в Интернет-кафе, не говоря о бесплатных почтовых службах вроде mail.ru, также сохраняют копию, копии остаются на всех серверах, через которые сообщение проходит по пути к адресату. Системные администраторы этих серверов могут по своему желанию прочитать письмо и переслать его, кому захотят. Спецслужбы крупных государств в рабочем порядке сканируют электронную почту на предмет подозрительных ключевых слов и фраз. Деловые послания могут представлять интерес для ещё большего круга лиц – от конкурентов до организованной преступности – и ставки в этой игре оказываются гораздо выше.

С помощью PGP вы можете зашифровать сообщение для своего адресата, даже если никогда прежде с ним не общались. Все эти организации и люди смогут по-прежнему получить доступ к зашифрованному письму, но уже не

будут иметь ни малейшего представления о его содержании, словно вы поместили его в непроницаемый конверт.

Pretty Good Privacy (буквально – "очень неплохая защита приватности"), свободно распространяемая криптографическая программа. Простая в использовании, PGP сегодня стоит на страже частной жизни миллионов пользователей Интернета.

PGP была придумана американским математиком и программистом Филипом Зиммерманом (Philip Zimmermann) в 1991 году. Получилась бесплатная программа для массового пользователя на основе алгоритма с открытым ключом. Она сразу стала набирать популярность. Правительство США пыталось бороться с распространением PGP (да и вообще стойкой гражданской криптографии). Зиммерману даже было предъявлено обвинение в "незаконном экспорте вооружений". Но энтузиасты со всех стран мира уже повсюду размещали PGP у себя на сайтах, изучали код программы и переводили ее на другие языки. В конце концов власти махнули рукой на это дело, и сегодня PGP работает на миллионах компьютеров.

2.1. Основы

2.1.1. Интерфейс

PGP предоставляет пользователю ряд альтернативных путей для доступа функциям программы, выбор конкретного зависит от решаемой вами задачи.

Существует четыре способа, которыми вы можете добраться до нужных функций и компонентов программы:

- Иконка PGPtray.
- Контекстное меню Проводника Windows.
- Меню Пуск.
- Плагины в email-клиентах.

Наиболее удобный и часто используемый путь – это иконка PGPTray (🔒, в зависимости от версии Windows может быть либо золотой, либо серой), расположенная в системном трее – в правой части панели задач Windows рядом с системными часами (рисунок 2.1). Оттуда вы можете быстро произвести любые операции шифрования с содержимым буфера обмена или активного окна, например, email-клиента, текстового редактора или формы на интернет-сайте, получить доступ к меню настроек PGP или к любому из компонентов программы.

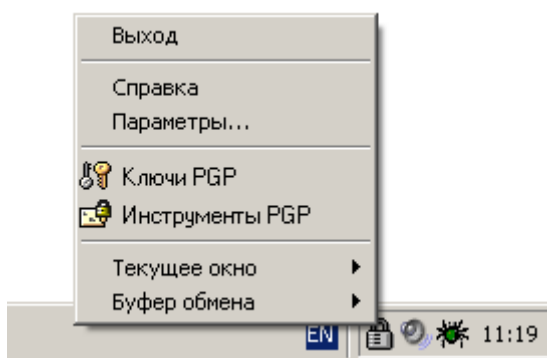


Рис. 2.1.

Нажав на эту иконку, вам открывается следующее меню:

- *Выход* – скрыть иконку PGPTray.
- *Справка* – справочник по работе с программой.
- *Параметры* – меню настроек программы.
- *Ключи PGP* – позволяет быстро получить доступ к соответствующему компоненту программы.
- *Инструменты PGP* – предоставляет доступ к основным функциям шифрования, уничтожения файлов и очистки свободного пространства дисков.
- *Текущее окно* – операции с содержимым активного окна позволяют зашифровать, расшифровать, поставить или сверить электронную подпись с текстовой информацией используемого в данный момент приложения (например, текстового редактора).

- *Буфер обмена* – аналогичные операции с содержимым буфера обмена позволяют, кроме перечисленного выше, быстро очистить или отредактировать находящийся в нём текст.

Вы можете воспользоваться средствами шифрования PGP из Проводника, нажав правой кнопкой на имя того или иного файла или папки, а затем в появившемся контекстном меню из пункта *PGP* выбрав нужную операцию (рисунок 2.2). Это весьма удобный способ работы с файлами.

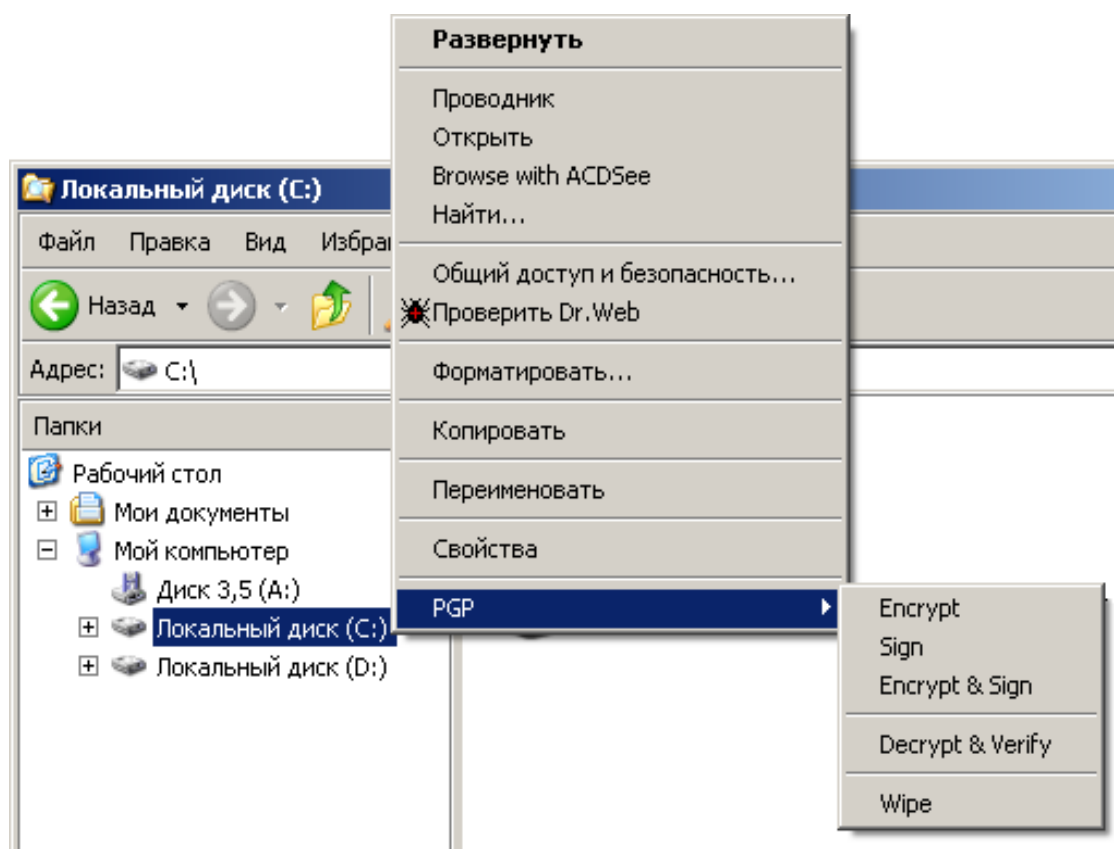


Рис. 2.2.

Содержимое контекстного меню и список доступных функций напрямую зависят от выбранного объекта.

- **Для дисков и папок.** Нажав правой кнопкой на любой из накопителей или папку и выбрав в главном контекстном меню пункт *PGP*, вы можете сделать следующее:

- Зашифровать (*Encrypt*), расшифровать (*Decrypt*), поставить (*Sign*) или сверить (*Verify*) электронные подписи с содержащихся на диске / в папке файлов.
- Удалить файлы с диска или папку со всем ее содержимым (*Wipe*).
- **Для файлов.** Нажав правой кнопкой на файл и выбрав в главном контекстном меню пункт *PGP*, в зависимости от типа файла вы можете сделать следующее:
 - Если выбран любой незашифрованный файл, вы можете уничтожить его (*Wipe*), зашифровать (*Encrypt*), подписать (*Sign*).
 - Если выбран зашифрованный / подписанный файл, вы можете уничтожить его (*Wipe*) или расшифровать / сверить подпись (*Decrypt & Verify*).
 - Если выбран файл в ASCII-формате (*.asc), вы можете уничтожить его (*Wipe*) или расшифровать / сверить подпись (*Decrypt & Verify*). При выборе последнего варианта для файла, содержащего материал ключа, вам будет предложено импортировать его на свою связку.
 - Если выбран файл связки открытых или закрытых ключей (*.pkg или *.skr соответственно), вы можете уничтожить его (*Wipe*) или импортировать содержащиеся в нём ключи на свою связку.

2.1.2. Компоненты PGP

Основными компонентами PGP являются ключи PGP (PGPkeys) и инструменты PGP (PGPtools).

Менеджер PGPkeys имеет средства управления вашими парами открытых и закрытых ключей, а также открытыми ключами ваших корреспондентов (рисунок 2.3).

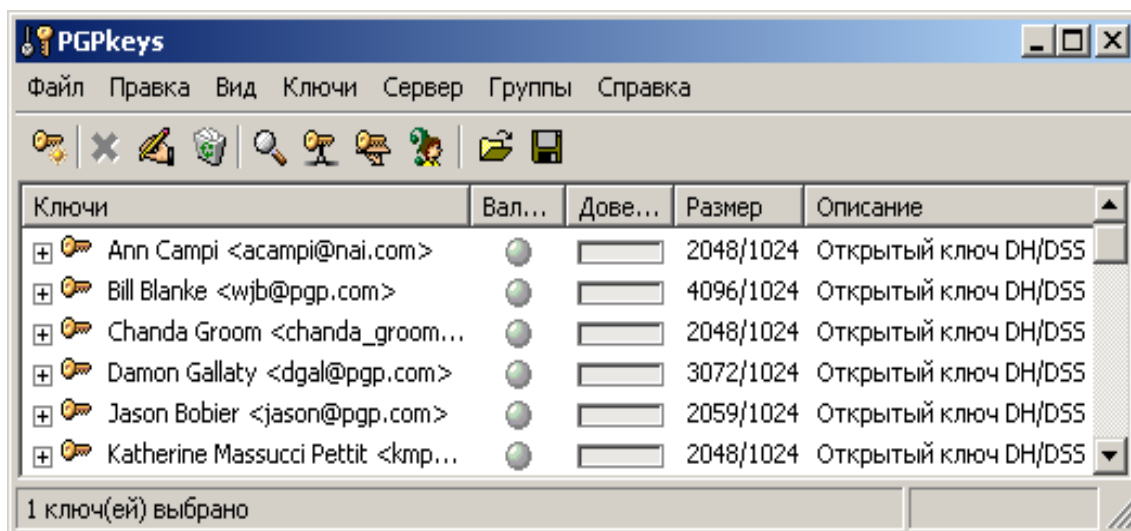


Рис. 2.3.

Чтобы открыть компонент PGPkeys:

1. нажмите иконку PGPtray (🔑);
2. из меню выберите *Ключи PGP*.

PGPtools – это небольшое плавающее окошко, предоставляющее доступ к основным функциям шифрования, уничтожения файлов и очистки свободного пространства дисков (рисунок 2.4).

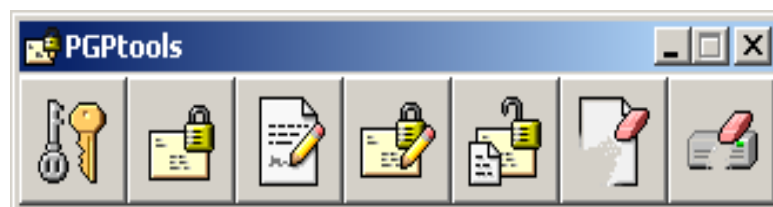


Рис. 2.4.

Чтобы открыть компонент PGPtools:

1. нажмите иконку PGPtray (🔑);
2. из меню выберите *PGPtools*.

2.1.2. Настройка

Изначальная настройка PGP пригодна для большинства пользователей. Тем не менее, предпочтительно перенастроить программу под специфику ваших условий и нужд до начала её эксплуатации. Ниже приведено описание всех опций настройки программы и рекомендации, могущие помочь в выборе конкретного варианта. Если сомневаетесь, какой вариант выбрать, опирайтесь на здоровую параною: чуть более строгие меры безопасности не доставят много дискомфорта, но помогут надёжнее сберечь информацию.

Будьте очень внимательны: некорректная настройка способна сильно повлиять на функциональность и безопасность работы программы.

Открыть меню настроек программы можно двумя разными способами:

- Нажать иконку PGPtray (🔒) > *Параметры*.
- В окне любого компонента PGP в строке меню выбрать *Правка* > *Настройки*.

2.1.2.1. Общие

Вкладка *Общие* содержит основные настройки PGP, связанные с общим функционированием программы (рисунок 2.5).

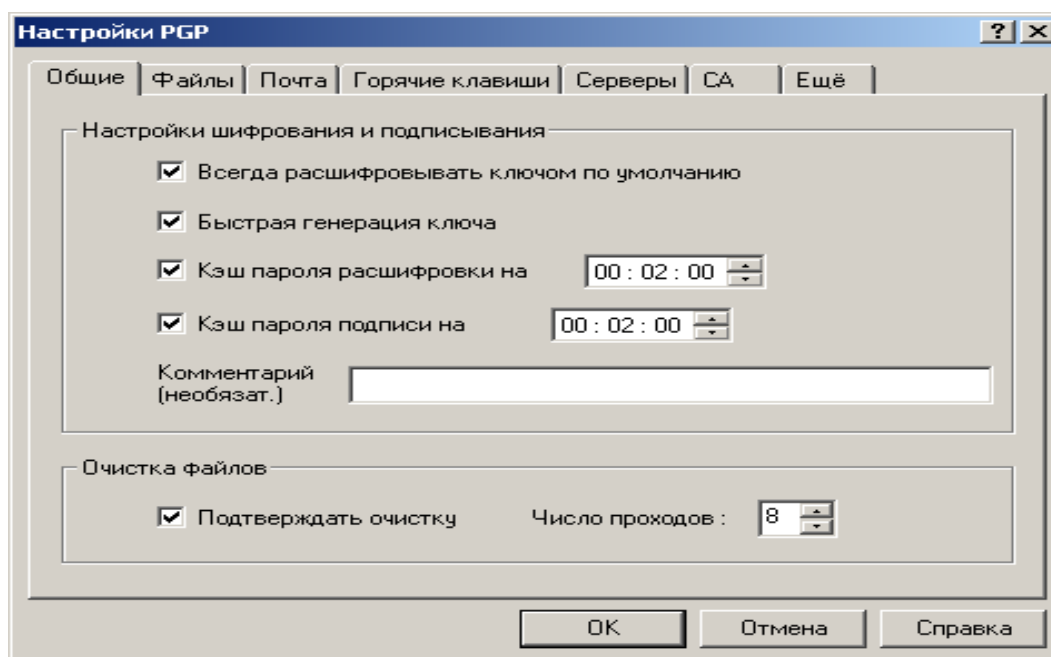


Рис. 2.5.

Раздел Настройки шифрования и подписывания

- *Всегда расшифровывать ключом по умолчанию.*
- *Быстрая генерация ключа* – быстрая генерация ключей Diffie-Hellman / DSS. При включении данной опции, программа будет в несколько раз быстрее создавать ключи DH/DSS, используя набор предварительно рассчитанного математического материала, лежащего в их основе, вместо его вычисления с нуля. Учтите, эта опция имеет значение только для генерации ключей DH/DSS, но не для RSA вследствие особенностей самих алгоритмов. Считается, что знание этого предрассчитанного материала не даёт взломщику преимуществ для осуществления атаки на ключи DH/DSS, и включение опции не несёт ущерба безопасности; но если вам от этого всё равно неуютно, можете эту опцию отключить, что и рекомендуется сделать, если вы не планируете генерировать очень много ключевых пар (чего, как правило, не требуется).
- *Комментарий* – сюда можно вписать короткий комментарий, который будет отображаться во всех зашифрованных или подписанных вами сообщениях в поле *Comment* после служебного заголовка *BEGIN PGP MESSAGE* или *BEGIN PGP SIGNATURE*. Комментарий никак не влияет на безопасность и сам может быть любым образом изменён вами или посторонним уже в зашифрованном или подписанном сообщении, поскольку не входит в сообщение, а является только служебным блоком данных. Для этой опции рекомендации отсутствуют: пишите или не пишите на своё усмотрение.
- *Кэш пароля расшифровки/подписи на ...* Если вы покинете рабочее место не перезагрузив компьютер или не очистив кэш, любой посторонний человек сможет беспрепятственно расшифровать ваши файлы и подделать подпись! В этом режиме программа будет хранить введённую ключевую фразу в памяти только указанный здесь срок. Это

удобно, когда нужно расшифровать / подписать сразу несколько файлов, но не хочется несколько раз подряд вводить длинную ключевую фразу.

Раздел Очистка файлов

Здесь можно настроить параметры уничтожения файлов (удаления без возможности восстановления).

- *Число подходов* – количество проходов очистки определяет, сколько раз сектора диска, содержавшие удаляемый файл, будут перезаписаны случайными данными. Обычно достаточно 3 (столько, например, предусматривает инструкция Минобороны США 5220.22). Для крайне ценных файлов увеличьте параметр до 9-12. Большее число проходов повышает надёжность уничтожения данных и снижает риск их намеренного восстановления, но и делает процесс стирания крайне долгим. Рекомендации таковы:
 - 1-3 прохода – для использования на домашнем компьютере;
 - 10-12 проходов – для использования в коммерческой и бизнес-сфере;
 - 16-18 проходов – для использования в военной сфере;
 - 26-28 проходов – для максимальной надёжности (может потребовать вплоть до нескольких часов работы для удаления достаточно крупного файла).

Коммерческие фирмы, специализирующиеся на восстановлении информации, могут восстановить данные, которые были перезаписаны примерно до девяти раз. Но вы должны учесть, что если ваша информация представляет чрезвычайную ценность (вероятно, ценность государственного масштаба), даже максимальное число проходов не обеспечит должного уровня надёжности. Питер Гутман, разработавший методику стирания информации, реализованную в PGP, рекомендует один достаточно надёжный способ уничтожения столь ценных данных:

сжечь носитель информации, пепел растереть в порошок и развеять по ветру.

- *Подтверждать очистку* – если включить, PGP будет выдавать предупреждение перед уничтожением файлов с просьбой подтвердить ваши намерения – последний шанс передумать. Рекомендую включить, поскольку специально или случайно уничтоженные данные будет исключительно трудно восстановить не только злоумышленнику, но и вам самим.

2.1.2.2. Файлы

Вкладка *Файлы* содержит местоположение связок ключей и пула случайных чисел (рисунок 2.6).

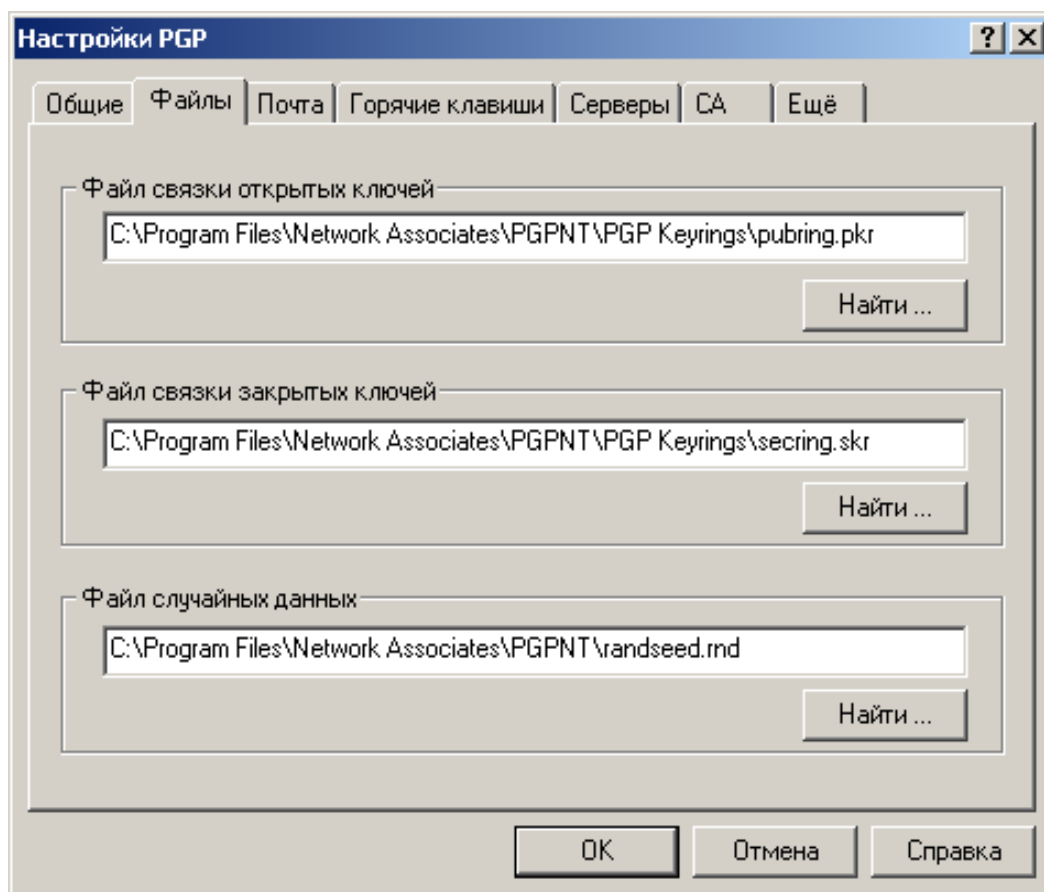


Рис. 2.6.

Раздел “Файл связки открытых ключей” указывает директорию и файл, где находится ваша связка открытых ключей; именно на ней хранятся ваши открытые ключи и открытые ключи всех ваших корреспондентов. Если необходимо, переместите файл в каталог, где он не будет случайно удалён вами или кем-то посторонним. В то же время не стоит помещать файл на внешний носитель, тем более на CDR-диск, поскольку для редактирования, добавления новых и удаления ненужных ключей файл должен быть доступен для записи.

Раздел “Файл связки закрытых ключей” указывает директорию и файл, где находится связка ваших закрытых ключей. **ВНИМАНИЕ:** удаление или порча этого файла приведёт к фактической потере всей зашифрованной информации! Крайне желательно переместить закрытые ключи на аппаратное криптографическое устройство (смарт-карту или USB-токен) либо всю связку – на внешний носитель, скажем, дискету, ZIP или CD-RW диск: в этом случае, во-первых, будет снижен риск случайного или злонамеренного удаления файла и, во-вторых, все закрытые ключи всегда будут под вашим физическим контролем (при условии надёжного хранения носителя), и могут быть использованы как обыкновенный ключ от дверного замка: подключаете к компьютеру, расшифровываете / подписываете что-либо, отключаете от компьютера и прячете.

В разделе “Файл случайных данных” указан путь к файлу с пулом псевдослучайных чисел, используемых программой для генерации сеансовых ключей и другого криптографического материала. Этот файл не содержит сколь-нибудь ценных данных, он лишь накапливает показатели энтропии. Чтобы воспользоваться этими показателями для проведения атаки на шифртекст, взломщику по меньшей мере придётся взломать гамма-генератор, обновляющий содержимое этого файла. Всё же стоит разместить его на логическом диске в

оперативной памяти компьютера, дабы он (файл, а не компьютер) уничтожался при каждой перезагрузке.

2.1.2.3. Электронная почта

Вкладка *Почта* содержит настройки интеграции с почтовым клиентом (рисунок 2.7).

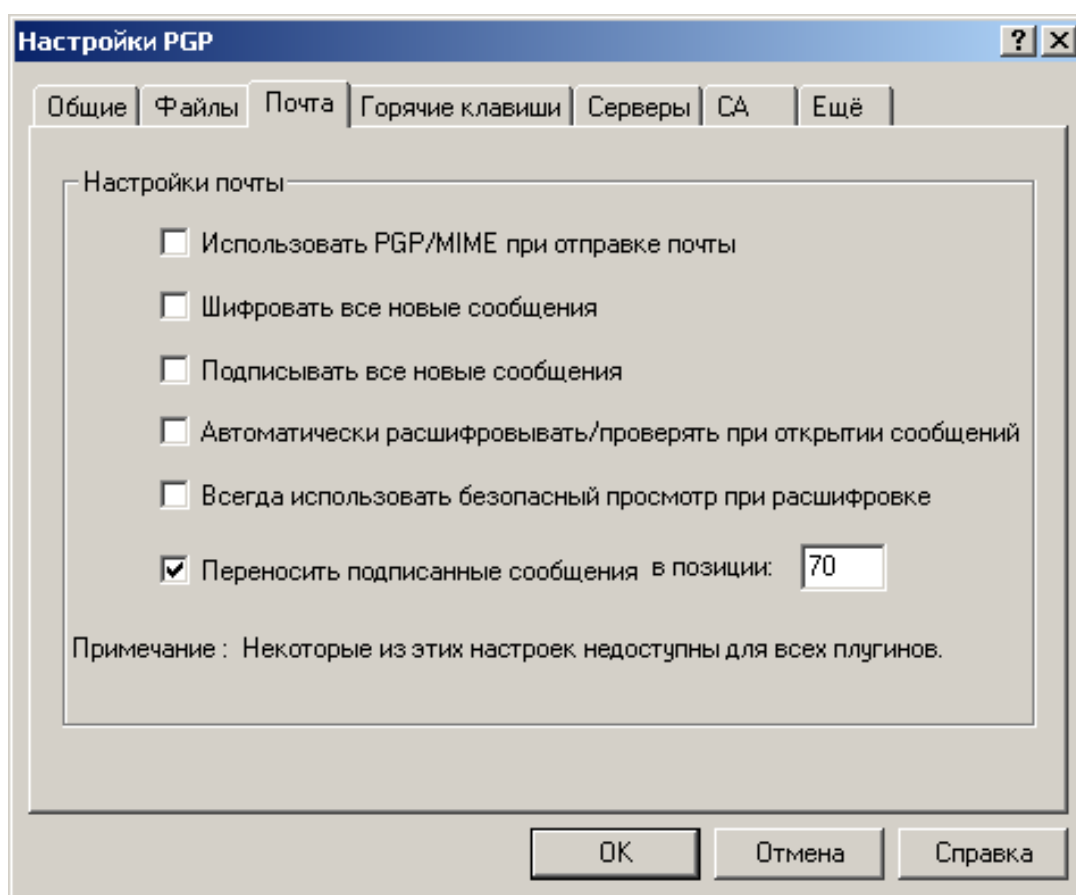


Рис. 2.7.

Если наряду с PGP вы установили плагин для интеграции с почтовым клиентом, здесь можно настроить некоторые параметры обработки почты. Не все из этих опций поддерживаются каждой почтовой программой. Кроме того, перед включением той или иной опции убедитесь, что и почтовая программа получателя также её поддерживает.

- *Использовать PGP/MIME при отправке почты* – если вы и ваши корреспонденты используете email-клиент Qualcomm Eudora, включение

этой опции позволит PGP автоматически зашифровывать всё содержимое письма (включая вложенные файлы и т.д.) и отправлять его в особом MIME-формате OpenPGP. Получателю нужно будет только нажать на ярлычок в пришедшем сообщении, и оно расшифруется с сохранением оригинального форматирования и всяческих украшательств в виде картинок и HTML-шаблонов. Данная опция (как для отправки, так и для получения) поддерживается только email-клиентом Qualcomm Eudora! Если вы не уверены в обратном, рекомендую отключить.

- *Шифровать все новые сообщения* – автоматически зашифровывать сообщения перед отправкой открытым ключом получателя. Поддерживается всеми email-клиентами. Если вам приходится часто пересылать зашифрованную корреспонденцию, лучше включить.
- *Подписывать все новые сообщения* – автоматически подписывать сообщения перед отправкой. Поддерживается всеми email-клиентами. Рекомендация аналогична предыдущей.
- *Автоматически расшифровывать / проверять при открытии сообщения* – автоматически расшифровывать / сверять подписи с открываемых сообщений. Поддерживается большинством email-клиентов.
- *Всегда использовать безопасный просмотр при расшифровке* – если включить, то любой расшифрованный текст (не только письма) будет выводиться в специальном окне *Secure Viewer*, используя шрифт, предотвращающий так называемую TEMPEST-атаку (удалённый съём информации по электромагнитному излучению монитора); кроме того, в этом случае сообщение невозможно будет сохранить в виде открытого текста. Для большинства случаев эту опцию лучше отключить (ещё и потому, что этот поставляемый с PGP TEMPEST-защитный шрифт не имеет кириллических символов).

- *Переносить подписанные сообщения в позиции ...* – производить на указанном символе в строчке жёсткий перенос (возврат каретки). Не меняйте установленное по умолчанию 70, если твёрдо не уверены, что в вашей почтовой программе используется меньший показатель, в противном случае сделайте эту цифру ниже, иначе ваш email-клиент, отформатировав текст перед отправкой, повредит цифровую подпись и шифртекст.

2.1.2.4. Горячие клавиши

Вкладка *Горячие клавиши* содержит настройки клавиш быстрого доступа к функциям PGP (рисунок 2.8).

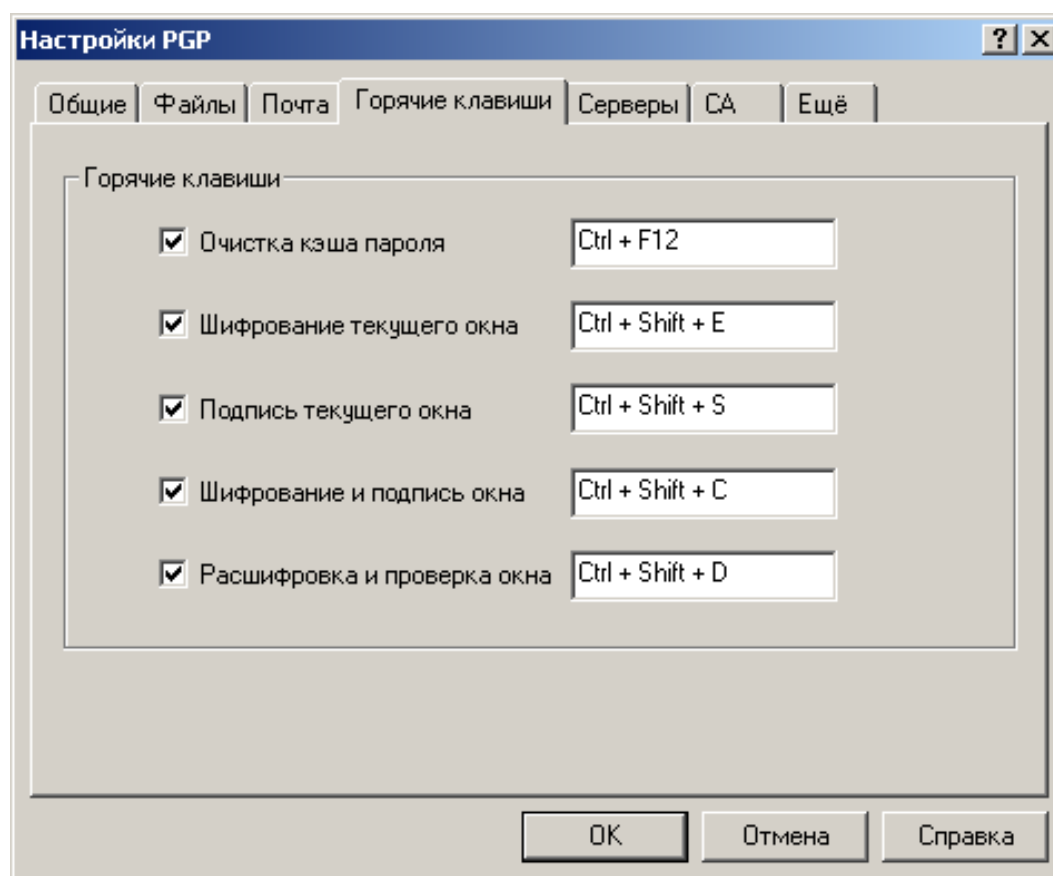


Рис. 2.8.

Здесь можно указать комбинации клавиш для быстрого выполнения тех или иных основных операций: шифрования, подписания и пр.

- *Очистка кэша пароля* – быстрая очистка кэша от ключевых фраз. Если вы включили режим кэширования ключевых фраз, обязательно нажимайте эту комбинацию клавиш, когда отлучаетесь от компьютера.
- *Шифрование текущего окна* – зашифровать содержимое активного окна.
- *Подпись текущего окна* – подписать содержимое активного окна.
- *Шифрование и подпись окна* – зашифровать и подписать содержимое активного окна.
- *Расшифровка и проверка окна* – расшифровать / сверить подпись с содержимого активного окна.

2.1.2.5. Серверы

Вкладка *Серверы* содержит настройки списка серверов-депозитариев (рисунок 2.9).

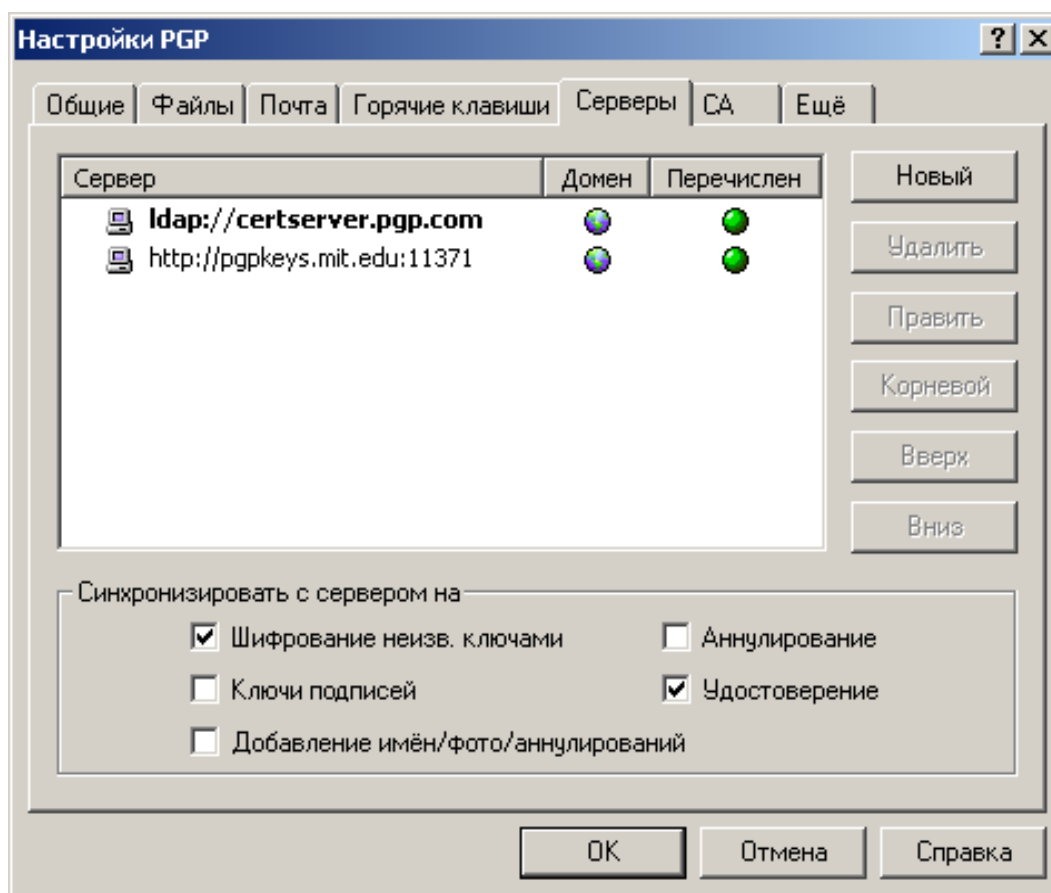


Рис. 2.9.

Кнопки справа от списка серверов позволяют вносить в него следующие изменения:

- *Новый* – добавить в список новый сервер-депозитарий.
- *Удалить* – удалить из списка указанный сервер.
- *Править* – редактировать параметры выделенного сервера.
- *Корневой* – сделать выбранный сервер корневым (или доменным). В корпоративной среде таковой используется для специфических задач, в частности, для обновления списков рассылки, настроек программы, доверенных поручителей и т.д. Для частных пользователей эта опция не представляет ценности.
- *Вверх* и *Вниз* – сдвинуть сервер в списке вверх или вниз. Поиск ключей на серверах в ходе синхронизации происходит в приоритетном порядке: если ключ не найден на первом сервере, производится поиск на втором и т.д. Поэтому крупные общественные серверы-депозитарии (как `ldap://certserver.pgp.com`) лучше оставить на самом верху.

Настройки в разделе “Синхронизировать с сервером на” позволяют указать, в каких случаях будет производиться синхронизация ключей с серверами. Желательно включить их все.

- *Шифрование неизв. ключами* – если включить, при отправке электронного письма человеку, открытого ключа которого нет на вашей связке, PGP попытается подключиться к депозитарию и самостоятельно найти ключ по email-адресу получателя. (Только при использовании почтового плагина.)
- *Ключи подписей* – если включить, при подписании чужого открытого ключа PGP сначала обновит его с сервера, а затем отправит на сервер подписанную вами копию.
- *Добавление имен / фото / аннулирований* – аналогично предыдущей опции, если вы внесёте в сертификат своего ключа новую запись (имя

или фото) либо добавьте т.н. "отменителя", PGP обновит ключ с сервера, а затем загрузит на сервер внесённые вами изменения.

- *Аннулирование* – после аннулирования открытого ключа PGP синхронизирует его с сервером, дабы в дальнейшем ваши корреспонденты не могли его применять.
- *Удостоверене* – если сверяете с сообщения или файла чужую ЭЦП, к которой на вашей связке не может быть найден подходящий открытый ключ, PGP попытается связаться с сервером и найти ключ по номеру ID.

2.1.2.6. Центр сертификации

Вкладка *CA (Certificate Authority)* содержит настройки установки соединения с Центром сертификации (ЦС) и объединения с инфраструктурой PKI. В основном эта функция применяется в корпоративной среде с развёрнутой PKI, основанной на стандарте X.509. Для частных пользователей она может не представлять интереса.

2.1.2.7 Дополнительные

Вкладка *Ещё* содержит расширенные настройки PGP (рисунок 2.10).

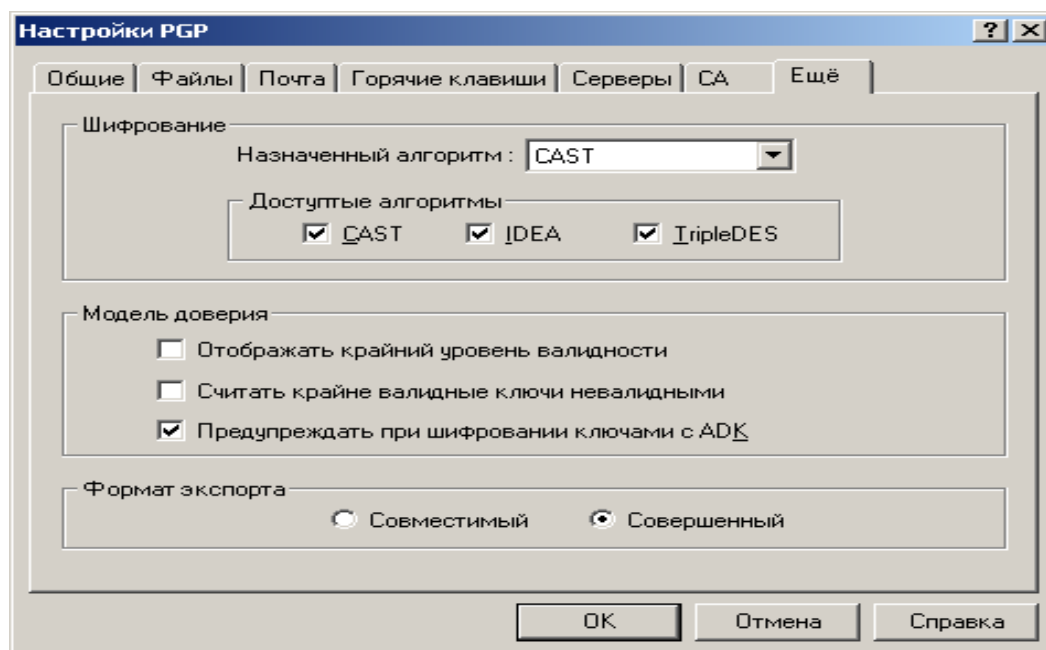


Рис. 2.10.

Раздел Шифрование

Настройки используемых шифровальных алгоритмов. Учтите, выбранные здесь настройки ложатся в материал генерируемых вами ключей, поэтому для уже существующих ключевых пар вы не обнаружите никаких изменений.

- *Назначенный алгоритм* – предпочтительный алгоритм симметричного шифрования. Следующая сгенерированная вами ключевая пара будет предпочтительно использовать указанный здесь блочный шифр. Иными словами, программа PGP отправителя зашифрует сообщение, предназначенное вам, именно этим алгоритмом, если и сама его поддерживает. Кроме того, этот же шифр будет применяться в дальнейшем и для симметричного шифрования с помощью обычного пароля.

По умолчанию выставлен CAST, в самых ранних был IDEA. С точки зрения обычного пользователя практических различий между ними нет – все чрезвычайно стойки. Некоторые были изобретены раньше, другие несколько новее:

- *CAST* – разработанный в 1993 году шифр со 128-битовым ключом и 64-битовым блоком. Дизайн основан на формальной архитектуре DES. Совершенно устойчив к линейному и дифференциальному криптоанализу, может быть взломан только "в лоб". Имеет множество модификаций, часть которых была признана ненадёжными. В PGP реализован стойкий вариант CAST5.
- *Triple-DES* – он же 3DES или тройной DES. Базовый алгоритм DES был разработан IBM в середине 1970-х и принят в качестве государственного стандарта шифрования США (и весьма распространился по миру). 3DES – это его вариация, в которой базовый DES выполняется трижды на одном блоке данных. В PGP он реализован в режиме EDE (зашифрование-расшифрование-зашифрование) с тремя независимыми подключами. Длина общего

ключа – 168 бит, оперирует на 64-битовых блоках. Теоретическая расчётная стойкость такого алгоритма к лобовой атаке составляет 112 бит, практическая – по меньшей мере 129 бит, что, вкупе с его проверенной годами надёжностью, крайне хороший показатель.

- *IDEA* – опубликованный в 1990, именно он лёг в основу первых версий PGP. Имеет ключи длиной 128 бит и оперирует на 64-битовых блоках открытого и шифртекста. Построен на концепции смешения операций различных алгебраических групп, а именно: XOR, сложение по модулю 2^{16} и умножение по модулю $2^{16}+1$. В ослабленных вариантах может быть подвержен криптоаналитическим атакам, но в базовом, который реализован в PGP, – нет.
- *Доступные алгоритмы* – допустимые симметричные алгоритмы. Как и предыдущая опция, данные установки ложатся в материал следующих генерируемых ключевых пар, а также используются вашей программой для шифрования отправляемых сообщений. Назначение этих опций в следующем: если программа PGP отправителя не поддерживает алгоритм, указанный у вас в качестве предпочтительно, она воспользуется одним из отмеченных здесь алгоритмов в качестве альтернативы. Выбор осуществляет в приоритетном порядке слева направо: скажем, если отправитель не поддерживает CAST, он зашифрует сообщение с помощью IDEA; если он не поддерживает CAST, а IDEA выключил, считая недостаточно надёжным, программа воспользуется 3DES, и т.д.

Снимайте галочки с перечисленных алгоритмов только в том случае, если у вас возникнут серьёзные и обоснованные (!) опасения в их стойкости, например, если станет доподлинно известно, что один из них был взломан! Если снять галочку с того или иного алгоритма, вы запретите вашей программе шифровать с его помощью сообщения для корреспондентов, а

корреспонденты не смогут с его помощью шифровать сообщения для вас, если те же настройки легли в материал вашего открытого ключа.

Раздел Модель доверия

Ряд расширенных настроек отношений доверия PGP.

- *Отображать крайний уровень валидности* – показывать или нет частичный уровень достоверности ключей. Если включить, уровень достоверности (подлинности) ключей в окне PGPkeys будет показан в виде шкалы с тремя состояниями: недостоверен (пустая шкала), частично достоверен (наполовину заполненная шкала), достоверен (целиком заполненная шкала). Если опция выключена, уровень достоверности будет показан как зелёный (достоверный ключ) или серый (недостоверный ключ) кружок. Если вам необходимо иметь полное представление о состоянии ключей на вашей связке – включите.
- *Считать крайне валидные ключи невалидными* – расценивать частично достоверные ключи как недостоверные. Если включено, при попытке зашифровать сообщение частично достоверным ключом, будет показано окно выбора ключей *Key Selection Dialog*, дабы предупредить о состоянии достоверности открытого ключа получателя.
- *Предупреждать при шифровании ключами с ADK* – выдавать ли предупреждение при шифровании, если ключ получателя содержит дополнительный ключ расшифрования (ADK). ADK используются в корпоративной среде, чтобы в определённых случаях иметь возможность расшифровать информацию своих служащих. Таким образом, наличие ADK говорит о том, что в этих определённых чрезвычайных случаях доступ к отправляемой информации могут иметь третьи лица, а не только фактический получатель.

Раздел Формат экспорта

Выбор формата экспортируемых со связки ключей:

- *Совместимый* – совместимый с версиями PGP до 6.x. Будет экспортирован только сам ключ и связанные с ним текстовые поля сертификата.
- *Совершенный* – новый формат, совместимый с PGP 6.x и выше. Кроме ключа будут экспортированы фотографии и прочее.

2.2. Управление ключами

Первая задача, возникающая после инсталляции и настройки программы – это генерация пары "открытый ключ / закрытый ключ". Именно асимметричные ключи позволят вам беспрепятственно обмениваться зашифрованными и подписанными сообщениями с людьми, живущими в любом конце света.

В рамках дальнейшего описания будут использованы следующие термины:

- *Ключевая пара* – асимметричная пара "открытый ключ / закрытый ключ".
- *Ключ* – в зависимости от контекста может подразумевать открытый ключ или ключевую пару.
- *Ключевая фраза, парольная фраза, пароль* – уникальная последовательность символов и/или слов, позволяющая использовать закрытый ключ асимметричной ключевой пары.
- *Связка* – связки открытых и закрытых ключей; pkr-, skr-файлы, указанные в настройках программы.
- *Основной ключ* – ключ, указанный как "ключ по умолчанию".
- *"Отменитель"* – designated revoker, человек, уполномоченный вами при необходимости аннулировать ваш ключ.

PGP предоставляет на выбор следующие типы асимметричных ключей: *Diffie-Hellman / DSS* (или просто *DH/DSS*) и *RSA*. В старых версиях PGP (до 5.0)

применялись только ключи RSA, использующие для шифрования и цифровой подписи асимметричный алгоритм RSA. В PGP 5.0 были добавлены ключи Diffie-Hellman, использующие шифрование по схеме Эльгамала и подписание по стандарту DSS.

При генерации нового ключа вам придётся сделать выбор его типа:

- Если вам нужна полная функциональность последних версий PGP и широкая совместимость вплоть до PGP 5.0, выбирайте *DH/DSS*. В большинстве случаев это предпочтительно. Однако следует помнить, что размер ключа подписания DSS всегда равен 1024 битам, независимо от размера ключа шифрования DH.
- Если вы не хотите быть скованными ограничением на 1024-битовый ключ подписания, выбирайте тип *RSA*.
- Если вы планируете общаться с людьми, использующими исключительно старые версии PGP (до 5.0), выбирайте *RSA*. Для целей совместимости эти ключи имеют ограничение длины до 2048 бит и не поддерживают множества новых функций (фотографические удостоверения, "отменителей" и пр.).

Также не забывайте, что вы можете сгенерировать столько ключей различных типов, сколько пожелаете.

Если вы прежде не использовали PGP и не имеете готовых связок ключей, которые указали в ходе инсталляции, то первое, что нужно сделать после установки и настройки программы – это создать свой первый ключ.

Хотя это может показаться увлекательным занятием, не создавайте больше одной ключевой пары, если в ином нет явной необходимости! Тому есть ряд причин. Во-первых, не пройдёт много времени, как вы окончательно в них запутаетесь. Во-вторых, что более важно, ни один ключ не может быть надёжнее защищающей его ключевой фразы. Нет смысла создавать множество ключей с идентичными ключевыми фразами (а много разных и *хороших* вы вряд ли запомните), поскольку взлом любой из них будет равносильен взлому всех.

Наконец, в-третьих, при необходимости отправить вам сообщение, незнакомый корреспондент может столкнуться с проблемой выбора ключа для зашифрования.

Чтобы сгенерировать новую ключевую пару сделайте следующее:

1. Откройте менеджер PGPkeys.
2. Нажмите иконку *Создать новую пару ключей* (🔑) в панели инструментов менеджера. Появится окно генерации ключа с описанием того, что такое открытые и закрытые ключи.
3. Нажмите кнопку *Далее* для продолжения.
4. В поле *Полное имя* введите своё имя, а в поле *Адрес Email* – адрес электронной почты. Несмотря на то, что указывать своё настоящее имя не обязательно, это может помочь корреспондентам идентифицировать данный открытый ключ как принадлежащий вам. То же касается и email-адреса, по которому ваш ключ будет проще отыскать на сервере и упростит корреспондентам отправку вам сообщений.
5. В меню *Тип пары ключей* выберите тип создаваемого ключа.
6. В поле *Размер пары ключей* укажите размер создаваемого ключа в битах. Более крупный ключ потребует больше времени на генерацию и на дальнейшие операции зашифрования / расшифрования, в то же время предоставляя большую степень надёжности. Если передаваемая вами информация не представляет ценность, сопоставимую с ценой проведения чрезвычайно дорогостоящей криптоаналитической атаки, будет более чем достаточно выставленных по умолчанию 2048 бит.
7. В разделе *Срок годности ключа* укажите дату истечения срока действия создаваемого ключа. Выберите либо установленное по умолчанию *“Вечный”* ключ (бессрочный), либо укажите определённую дату, с которой ключевая пара не сможет применяться для задач зашифрования и подписания (тем не менее, ею можно будет продолжать пользоваться для расшифрования и сверки ЭЦП). *“Вечный”* ключ является

предпочтительным. Если, однако, вы планируете использовать данный ключ только определённый период (например, в течение срока действия контракта с работодателем), укажите здесь дальнюю границу этого периода.

8. Нажмите *Далее*.

9. В окне выбора ключевой фразы введите в оба представленных поля пароль, которым хотите защитить свой новый закрытый ключ.

Парольная фраза – это единственный и по этой причине самый главный механизм защиты закрытого ключа от несанкционированного использования. Вся надёжность PGP упирается в качество выбранной вами на этом этапе ключевой фразы. В порядке меры предосторожности программа скрывает вводимые символы. Если вам от этого неудобно и вы уверены, что в помещении нет посторонних глаз, снимите галочку со *Скрыть*.

ВНИМАНИЕ: Если вы позднее забудете введённую на этом этапе парольную фразу, никто не сможет помочь вам воспользоваться закрытым ключом данной ключевой пары, и вся зашифрованная с её помощью информация фактически будет утеряна!

В целях совместимости крайне не рекомендуется использовать для ключевой фразы кириллицу и другие нелатинские национальные буквенные символы.

Если же вы считаете, что их использование необходимо, протестируйте созданный ключ на не представляющей ценности информации и убедитесь, что можете свободно её расшифровать, прежде чем применять ключ по назначению.

10. Нажмите *Далее*.

11. Если введённая на предыдущем этапе ключевая фраза не соответствует нормам безопасности, PGP выдаст предупреждение. Вернитесь назад и

устраните проблему, ибо её игнорирование повлечёт серьёзные проблемы с защищённостью ключа.

12. Движения вашей мышки и нажатия на клавиши создают множество случайной информации (энтропии), обязательной для генерации ключей. Однако бывает так, что PGP не успевает накопить достаточно энтропии до начала генерации ключа. В таком случае появится окно сбора случайных данных *PGP Random Data*: просто подвигайте мышкой и понажимайте на произвольные клавиши, пока шкала не заполнится целиком. Если же всё нормально, PGP приступит к формированию ключа.
13. В зависимости от мощности компьютера и от длины создаваемого ключа на этот этап может потребоваться разное количество времени: от нескольких секунд до десятков минут. Дождитесь, пока не появится сообщение *Завершено*. После этого можете нажать кнопку *Далее* и затем *Готово*.

PGP самостоятельно разместит открытый и закрытый ключи в соответствующих файлах связки, а имя ключа появится в окне менеджера PGPkeys.

Зашифровать файл, а после обнаружить, что не можешь его расшифровать – это болезненный опыт, тем не менее, помогающий понять, как правильно выбирать ключевую фразу, которую удастся запомнить.

Большинство приложений с ограничением доступа предлагают использовать в качестве пароля слово из трёх-восьми букв. Очень нежелательно использовать подобные пароли по ряду причин. Во-первых, они сильно уязвимы к атакам "по словарю", когда взломщик заставляет компьютер перебирать все слова из словаря, пока не угадает пароль. Во-вторых, они могут взломаны полным перебором всех возможных комбинаций букв, печатных символов и цифр.

Чтобы защититься от подобного рода атак рекомендуется создавать парольное слово, состоящее из заглавных и строчных букв, цифр, знаков препинания и пробелов. В результате получается пароль, который довольно сложно подобрать, но ещё труднее запомнить. Использование в составе пароля множества произвольных небуквенных символов повышает его стойкость к атакам "по словарю", но и затрудняет его запоминание, что, рано или поздно, может привести к катастрофической потере информации по той лишь причине, что вы не сможете расшифровать собственные файлы.

С другой стороны, ключевая фраза, или *осмысленный пароль*, – это последовательность логически связанных слов, обычно, длинное предложение, которое гораздо менее уязвимо к "словарным" атакам. Однако, если вы не выберете в качестве ключевой фразы нечто, давно хранящееся в долгосрочной памяти мозга, то едва ли сможете запомнить её буквально.

Выбор ключевой фразы под влиянием обстоятельств скорее всего приведёт к тому, что вы начисто её забудете; не поможет и попытка "зазубрить" – так устроена память. Выберите что-то, уже находящееся в вашей долгосрочной памяти. *Это не должна быть* фраза, которой вы с кем-то недавно делились или которую часто любите повторять, и не должен быть известный афоризм или цитата, поскольку всё это будет со временем подобрано опытным взломщиком. Можете построить ключевую фразу на ассоциации или ассоциативном ряде, задав себе вопрос, ответ на который знаете только вы. Но это должно быть нечто, давно и глубоко хранящееся в вашем мозге, однако и не что-то очевидное и легко предсказуемое. Альтернативный вариант – это мнемотехнические методики, но они требуют определённой практики и опыта. Постарайтесь несколько "усилить" результат заглавными буквами в произвольных местах и небуквенными символами, только не переусердствуйте.

Разумеется, если вы будете столь недальновидны, что запишите результат на листке бумаги и положите его в ящик письменного стола, не имеет большого значения, сколь хорошую ключевую фразу вы придумаете.

Сгенерировав новую ключевую пару немедленно сделайте несколько её резервных копий на разных внешних носителях! (В действительности, PGP сам предложит вам это сделать, когда вы закроете окно PGPkeys. Ни в коем случае не пренебрегайте этой рекомендацией!) Игнорирование этого требования приводит к неоправданному риску потери всех ценных данных. Если что-то случится с единственным файлом связки закрытых ключей, никто во всём мире не поможет вам расшифровать ваши файлы.

Кроме резервного копирования `pkc`- и `skr`-файлов связки ключей, обратите особое внимание на то, где хранится ваш закрытый ключ. Хотя закрытый ключ защищён ключевой фразой, известной только вам, посторонний может узнать её, например, просто подсмотрев из-за спины, какие клавиши вы нажимаете, или перехватив нажатия клавиш через локальную сеть или даже через Интернет, а затем воспользоваться закрытым ключом, чтобы расшифровывать вашу информацию и подделывать подпись.

Чтобы избежать подобных сценариев, храните закрытый ключ только на своём компьютере. Если ваш компьютер подключён к локальной сети, убедитесь, что файлы связок не подлежат автономному резервному копированию на носители, к которым могут получить доступ посторонние лица. Учитывая лёгкость, с которой злоумышленник может проникнуть в компьютер через сеть, установите дополнительные защитные барьеры в виде межсетевых экранов и антивирусных программ. Работая со сверхценной информацией, разместите свой закрытый ключ на дискете или, что предпочтительнее, на смарт-карте, которую можно использовать аналогично ключу от дверного замка, подключая к компьютеру, только когда нужно подписать или расшифровать информацию.

Ещё одна мера предосторожности заключается в переименовании связки закрытых ключей (`skr`-файла) и перемещении её в отдельный от открытых

ключей каталог. Для этого воспользуйтесь вкладкой *Файлы* меню настроек программы.

2.3. Управление связкой ключей

Ключи, созданные вами или полученные от корреспондентов, хранятся на связках, по сути представляющих собой два файла (базы данных): один содержит открытые ключи и по умолчанию назван *pubring.pkr*, другой предназначен для закрытых и называется *secring.skr*. Изначально эти файлы хранятся в каталоге с программой PGP или в папке “Мои документы\PGP”.

В некоторых случаях вам может потребоваться изучить атрибуты ключей и их сертификатов или изменить их параметры. Скажем, получив от корреспондента открытый ключ, вы захотите установить его тип, проверить отпечаток и по содержащимся на сертификате подписям определить достоверность. Затем вы решите сами подписать этот ключ, чтобы указать на его подлинность, и настроить уровень доверия владельцу в заверении других ключей.

Порой может возникнуть необходимость изменить ключевую фразу вашего собственного закрытого ключа или найти чей-то открытый ключ на общественном сервере.

Для выполнения всех перечисленных и некоторых других мероприятий служит менеджер PGPkeys.

2.3.1. Основы PGPkeys

Окно менеджера PGPkeys содержит список всех ваших ключевых пар и чужих открытых ключей, добавленных вами на связку. В верхней части окна расположена панель инструментов, предназначенная для выполнения наиболее обычных задач, и строка меню, предоставляющая доступ к дополнительным функциям.

Большинство операций с ключами может быть выполнено четырьмя разными способами:

- Через иконки в панели инструментов.
- Через строку меню в верхней части окна.
- Через контекстное меню по нажатию правой кнопкой на имя ключа или составляющих его элементов.
- С помощью горячих клавиш PGPkeys (они отображены напротив соответствующих функций в меню в верхней части окна).

Все эти способы совершенно равноправны.











2.3.1.1. Атрибуты ключей

Наряду с именами ключей окно PGPkeys отображает некоторые из их параметров и атрибутов. В меню *Вид* вы можете указать, какие атрибуты будут отображаться в окне менеджера, а в самом окне при желании можете изменить порядок расположения столбцов атрибутов (перетащив столбец за шапку) и сортировку списка ключей по любому из атрибутов (нажав левой кнопкой на шапку нужного столбца).

Окно PGPkeys может показывать следующие параметры ключей:








- **Ключи (Keys)** – этот атрибут представлен набором пиктографических изображений, обозначающих различные параметры ключа. Также он содержит имя владельца, сведения сертификата ключа и имена его поручителей. В таблице 2.1 приведено описание пиктограмм, соответствующих атрибуту “Ключи”.

Таблица 2.1

	Золотой ключ и человечек обозначают принадлежащую пользователю пару "открытый ключ / закрытый ключ" типа Diffie-Hellman / DSS.
	Серый ключ и человечек обозначают принадлежащую пользователю пару "открытый ключ / закрытый ключ" типа RSA.
	Золотой ключ обозначает открытый ключ типа Diffie-Hellman / DSS.
	Серый ключ обозначает открытый ключ типа RSA.
	Пара ключей обозначает разделённый ключ. Такой может использоваться для расшифрования / подписания только после объединения.
	Тусклый ключ обозначает временно деактивированный открытый ключ. Такой не может использоваться для зашифрования. Это удобно при большом количестве открытых ключей на связке, сильно захламляющих окно <i>Key Selection Dialog</i> .
	Серый ключ на золотой карте обозначает сохранённый на смарт-карте ключ типа RSA.
	Ключ с красным запрещающим знаком обозначает аннулированный открытый ключ. Это значит, что он либо был скомпрометирован, либо по иным причинам более не используется владельцем.
	Ключ с часиками обозначает просроченный открытый ключ, чей период действия уже истёк.
	Два человечка обозначают группу открытых ключей списка рассылки.

В таблице 2.2 приведены пиктограммы, обозначающие содержимое сертификата:




Таблица 2.2

	Конверт обозначает обычное имя в сертификате ключа; как правило, это просто имя и email-адрес владельца ключа. Конверт может быть жёлтым или серым в зависимости от типа ключа (DH/DSS или RSA).
	Конверт с красным запрещающим знаком обозначает аннулированную запись сертификата.
	Картинка обозначает фотографическое удостоверение в сертификате.
	Карандаш (или шариковая ручка) обозначает подпись, подтверждающую ту или иную запись сертификата ключа. Иконка карандаша без дополнительных символов – это неэкспортируемая подпись, заверяющая ключ только на связке пользователя.
	Карандаш с синей стрелкой обозначает экспортируемую со связки подпись. Такая используется как поручительство пользователя в подлинности ключа и данной записи сертификата
	Карандаш с красным запрещающим знаком обозначает отозванную подпись.
	Тусклый карандаш обозначает неверную или повреждённую подпись.

- **Достоверность (Validity)** – обозначает степень убеждённости в том, что открытый ключ действительно принадлежит предполагаемому владельцу. Зависит от состава подписей, заверяющих данный ключ, и уровней доверия пользователя поручителям (людям, подписавшим





ключ). Ключ, подписанный непосредственно пользователем, становится полностью достоверным, исходя из логики программы, что пользователь не станет подписывать поддельный открытый ключ. Ключ, не имеющий подписей, считается недостоверным, о чём программа будет напоминать всякий раз при попытке зашифровать информацию данным ключом. Атрибут достоверности может быть показан либо в виде цветного кружка, либо в виде шкалы, в зависимости от установок параметра *Display marginal validity level* в настройках программы. Если опция выключена, то атрибут отображается следующим образом (таблица 2.3):

Таблица 2.3

	Серый кружок обозначает недостоверные ключи (и частично достоверные ключи, если опция <i>Treat marginally valid keys as invalid</i> в настройках программы включена).
	Зелёный кружок обозначает достоверные открытые ключи.
	Зелёный кружок и человечек обозначают безусловно достоверную ключевую пару пользователя.

- **Размер (Size)** – длина асимметричного ключа в битах. Для ключей типа DH/DSS и RSA отображается два числа: первое – длина подключа шифрования, второе – длина ключа подписания. Ключ подписания DSS всегда равен 1024 битам.
- **Доверие (Trust)** – обозначает указанный пользователем уровень доверия владельцу данного ключа в заверении чужих открытых ключей (таблица 2.4). Это влияет на степень достоверности чужих ключей, поручителем которых выступает владелец данного.

Таблица 2.4

	Пустая шкала говорит о том, что владелец данного ключа не имеет доверия и не может выступать поручителем (его подпись не учитывается при расчёте достоверности ключей).
	Частично заполненная шкала говорит о том, что подписанный владельцем данного ключа открытый ключ будет иметь частичную достоверность.
	Полностью заполненная шкала обозначает, что владелец данного ключа имеет полное доверие в заверении других открытых ключей, и любой ключ, подписанный им, считается программой достоверным.
	Заштрихованная шкала указывает на безусловно доверенный ключ пользователя.

- **Описание (Description)** – краткое описание объекта в колонке **Ключи:** тип и состояние ключа, тип удостоверения, вид подписи и т.п.
- **ID ключа (Key ID)** – уникальный идентификационный номер, помогающий отличить несколько открытых ключей с одинаковыми именами владельца (в очень редких случаях сами номера ID у разных ключей совпадают).
- **Создание (Creation Date)** – указывает дату, когда ключ был сгенерирован. Иногда можно исходить из этой информации при анализе подлинности ключа. Если он был создан довольно давно, маловероятно, что его станут подменять, поскольку за прошедшее время оригинальные копии должны были получить широкое распространение.

Но никогда не полагайтесь на этот показатель как на единственный параметр анализа (его крайне легко сфальсифицировать)!

- *Срок годности (Expiration Date)* – указывает дату, когда ключ станет неприменим для новых криптографических задач, либо “Вечный” (*Never*), т.е. неограниченный срок действия.
- *ADK* – наличие дополнительных ключей расшифрования. Серым или зелёным кружком показывает, содержит ли конкретный ключ ADK.

Также в меню *Вид* можно включить или выключить показ панели инструментов (*Панель инструментов*).

2.3.1.2. Выбор основного ключа

Основной ключ пользователя, или ключ по умолчанию, используется программой, чтобы автоматически зашифровывать информацию не только для получателя, но и для вас самих, дабы в дальнейшем вы имели возможность, например, расшифровать и прочитать отправленное письмо. Подписывая сообщение или чей-то открытый ключ, PGP будет также предлагать использовать ваш ключ по умолчанию (разумеется, если вы пожелаете воспользоваться другим своим закрытым ключом, то сможете это сделать). В окне PGPkeys основной ключ выделен **жирным шрифтом**, дабы отличить его от остальных. Если вы используете несколько ключевых пар, выбор одной как основной сделает работу с PGP более удобной.

Чтобы выбрать основной ключ:

1. В окне PGPkeys выделите тот свой ключ, который хотите сделать основным.
2. В строке меню нажмите *Ключи > Назначить ключом по умолчанию*.

Имя ключа станет жирным, обозначая, что теперь он используется по умолчанию.

2.3.1.3. Импорт и экспорт ключей

Наиболее удобным способом обмена ключами является их пересылка через сервер-депозитарий, но иногда может потребоваться отправить открытый ключ в виде отдельного файла (например, через FTP-сервер). Или вы можете захотеть сохранить резервную копию отдельных ключей, а не связок целиком (чтобы зарезервировать связку, достаточно скопировать файлы `pubring.pkr` и `secring.skr`, расположение которых можно узнать во вкладке *Файлы* меню настроек программы). В этом случае можно экспортировать или копировать ключи в файл.

Если же ваш корреспондент изберёт в качестве способа передачи отправку открытого ключа по электронной почте, воспользуйтесь возможностью импортирования, чтобы добавить полученный ключ на свою связку. То же касается и восстановления резервных копий, и иных схожих задач.

Ниже приведены способы импортирования / экспортирования ключей.

Экспортирование ключа со связки в файл:

1. В окне `PGPkeys` выделите ключ, который хотите экспортировать. Можете экспортировать сразу группу ключей, выделив несколько нужных.
2. В строке меню нажмите *Ключи > Экспорт*.
3. Чтобы вместе с ключами экспортировать фото-удостоверения, отметьте опцию *Включить расширения 6.0*. Однако учтите, что в этом случае экспортированные ключи будут несовместимы с версиями PGP до 6.0.
4. Если в числе экспортируемых ключей присутствуют ваши ключевые пары, и кроме открытых вы хотите сохранить и закрытые ключи, отметьте галочкой опцию *Включить закрытые ключи*.
В этом случае будьте внимательны, чтобы экспортированный файл не попал в руки к посторонним.
5. Укажите имя файла и каталог, где хотите его сохранить.

Копирование материала ключа со связки позволяет позднее вставить его в любой текстовый файл или в тело письма. Весьма удобный способ для некоторых мероприятий. Но копировать таким образом материал закрытых ключей невозможно.

Копирование материал ключа со связки в документ:

1. В окне PGPkeys выделите ключ, который хотите копировать. Можете копировать сразу группу ключей, выделив несколько нужных.
2. В строке меню нажмите *Правка > Копировать*.

Материал ключа (или ключей) помещён в буфер обмена (рисунок 2.11). Теперь можете вставить его в любой текстовый документ простой функцией *Вставить* или комбинацией клавиш *Ctrl+V*.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----  
Version: PGPfreeware 6.5.8 for non-commercial use <http://www.pgp.com>  
Comment: Копированный блок открытого ключа  
  
mQCNBEFiQG8BBACyXzMkqNm0gEB6CejyiftiHtAbahwF79++mwGnA2nEWW+g7iUt  
Y9/ZXjpkEEM6Y3ZeOCiCt1SyOq4aqpIM+74BoBgoqCPkcQ7eFAOq3Ihkoiz3CmjR  
LLNo1yz5N12/Q6fU+U4LTAC3u4en+m+MavdLp85EhOEiEmPxiUcbgVE4BwARAQAB  
tAMxMjOJAK4EEAEACABgFAkFiQG8ICwIDCQgHAQoCGQEFGwMAAAACGkQ4+j9xbFm  
GcOKBQP+Nr17DYzatk7gtgXLJvjYbfnIEgTDww97kocBYOafq1ETDD3ObVkWp0F  
y8hOUpkpxBE4+C7F1CCv6CWpk96dN/F3w0v6XkHMeP170WOGhk7dJKaH70y0aWK0  
/mMNpnz5ZtzLvodANG6Q0ddHBvXuJW8KB13yOxVm0nT7AjbBmZi5AI0EQWJACAE  
AJ+kaOL9lYIbOMd/mfRw5yiAMrU/QdSLKhyzYf98OVuSTqDH1BLwXbA0vEN5U/al  
eKNWegfmFdXqRct9PXv8ECTzCVVxobo5IhXAOkL3nCyxAdadBgMV1BASdKZtd8QZ  
BcFeUEdSqcAVe992jXWs4xec0aIwlk2gRk3J3kFlgjGjABEBAAGJAKIEGAECAAwF  
AkFiQHAFGwwAAAAACGkQ4+j9xbFmGcOBagQAqF/ODSYBha5+dC/95PkTtPGulF7M  
BqWwrPv3luIPOSCmNCobHVZCRYs1507WLoirMwi7sSA8K00eWASbXJMtGKSXHxnq  
M5A3TD9rkPlCOvB8iS1jM6qPlguD9CTuIU6hSQGtGtSnZBQ51wgCTyg42aR/D5YZ  
KCmMilmVpwrKZqI=  
=lZr7  
-----END PGP PUBLIC KEY BLOCK-----
```

Рис. 2.11.

Импортирование ключа из файла на связку:

1. В строке меню PGPkeys нажмите *Ключи > Импорт*.

2. В меню *Тип файла (File Types)* выберите тип импортируемого файла. Это может быть *txt* или *asc* для текстового материала ключа PGP, файлы связок PGP с расширениями *pkc*, *skr*, *pubkc*, *seckr* и *pgp*.
3. Укажите импортируемый файл и нажмите кнопку *Открыть (Open)*.
4. В появившемся окошке *Укажите ключ(и)* отметьте те ключи из выбранного файла, которые хотите импортировать, и жмите *Импорт*.

Ключи будут присоединены к вашим связкам. Если среди импортированных были и закрытые ключи, программа предупредит, что им необходимо указать соответствующий уровень доверия. Для этого откройте свойства этих ключей (*Ключи > Свойства* в строке меню) и установите флажок на опцию *ИмPLICITное доверие*, наделяющую ключ безусловным уровнем доверия.

Если полученное вами письмо или текстовый файл содержат материал ключа, можете добавить его на связку следующим образом:

1. В полученном тексте выделите блок, начиная с заголовка “-----BEGIN PGP PUBLIC KEY BLOCK-----” (или “PRIVATE KEY BLOCK” для закрытого ключа) и заканчивая строкой “-----END PGP PUBLIC KEY BLOCK-----” и копируйте выделенный материал в буфер обмена (обычно можно просто нажать *Ctrl+C*).
2. В строке меню PGPkeys нажмите *Правка > Вставить*.
3. В появившемся окошке *Укажите ключ(и)* отметьте те ключи из полученного материала, которые хотите импортировать, и жмите *Импорт*.

2.3.1.4. Удаление ключей, подписей и сертификатов


Иногда может потребоваться удалить со связки ненужный ключ, заверяющую его подпись или запись из сертификата.

Удаление ключа со связки необратимо. Хотя вы можете повторно импортировать открытый ключ, добавить запись в сертификат или снова

заверить ключ прежде удалённой подписью, удаление закрытого ключа, не имеющего резервных копий, приведёт к фактической потере всей информации, зашифрованной соответствующим открытым ключом, поскольку эта информация более не сможет быть расшифрована!

Не забывайте, что удаление своего ключа со связки не аналогично его аннулированию. Если вы больше не собираетесь использовать ключевую пару, аннулируйте её и обновите на сервере, чтобы корреспонденты не использовали данный открытый ключ для отправки вам сообщений.

Чтобы удалить ключ, сертификат или подпись со связки:

1. В окне PGPkeys выделите объект, который хотите удалить.
2. В строке меню нажмите *Правка > Удалить* либо нажмите кнопку *Удалить выбранный элемент* () в панели инструментов.
3. На просьбу подтвердить удаление нажмите *Да*.

Выбранный объект будет удалён со связки.

2.3.1.5. Активирование / деактивирование ключей

Если количество ключей на вашей связке становится угрожающе велико, и поиск нужного для зашифрования письма оборачивается всё более трудной задачей, вы можете временно деактивировать ключи, которые не используете постоянно, но и не хотите удалять. С этого момента они не будут захламлять окно *Key Selection Dialog*.

Для деактивации открытых ключей корреспондентов:

1. В окне PGPkeys выделите ключ, который хотите деактивировать.
2. В строке меню нажмите *Ключи > Запретить*.

Пиктограмма ключа потускнеет, обозначая, что он временно отключён.

Чтобы снова активировать ключ для использования:

1. Выделите ключ, который хотите активировать.
2. В строке меню нажмите *Ключи > Разрешить*.

Пиктограмма станет обычной, а ключ – готовым к работе.

Если фраза об угрожающем количестве ключей на связке справедлива для ваших собственных ключевых пар, некоторыми из которых вы пользуетесь относительно редко, можете деактивировать и их. В этом случае конкретный ключ не сможет применяться для зашифрования и подписания данных, но вы сможете продолжать им пользоваться для расшифрования файлов и сверки своих ЭЦП.

Чтобы деактивировать ключевую пару:

1. В окне PGPkeys нажмите правой кнопкой на ключ, который хотите деактивировать > *Свойства ключа*.
2. В окне свойств ключа снимите галочку с опции *ИмPLICITное доверие*, а затем с *Разрешен*. Закройте окно свойств.

Изображение человечка с пиктограммы ключа пропадёт, а сам ключ потускнеет, обозначая, что он временно неактивен. Чтобы снова активировать ключевую пару:

1. Нажмите правой кнопкой на ключ, который хотите активировать > *Свойства ключа*.
2. В окне свойств ключа отметьте галочкой параметр *Разрешен*, а затем – *ИмPLICITное доверие*. Закройте окно свойств.

Пиктограмма станет обычной, а ключевая пара – готовой к работе.

2.3.2. Просмотр и настройка свойств ключей

Кроме просмотра наиболее общих атрибутов ключей непосредственно в окне менеджера PGPkeys, вы можете изучить и отредактировать дополнительные параметры любого ключа в окне его свойств.

Сведения в окне свойств ключа (*Свойства ключа*) разбиты по четырём вкладкам (рисунок 2.12):

- “*Общие*” содержит основные параметры и описание ключа;
- “*Подключи*” позволяет редактировать подключи шифрования;
- в “*Аннуляторы*” перечислены "отменители" данного ключа;

- в “ADK” указаны дополнительные ключи расшифрования.

Учтите, что вкладки “ADK” и “Аннуляторы” могут отсутствовать, если ключи ADK и “отменители” не были добавлены к данному ключу.

Чтобы открыть окно свойств, в менеджере PGPkeys нажмите правой кнопкой на имя ключа > *Свойства ключа*. Либо выделите нужный ключ и в панели инструментов нажмите кнопку *Показать свойства ключа или сертификата* (👤).

2.3.2.1. Основные свойства и смена ключевой фразы

Во вкладке основных свойств *Общие* (рисунок 2.12) содержатся следующие сведения и настройки.

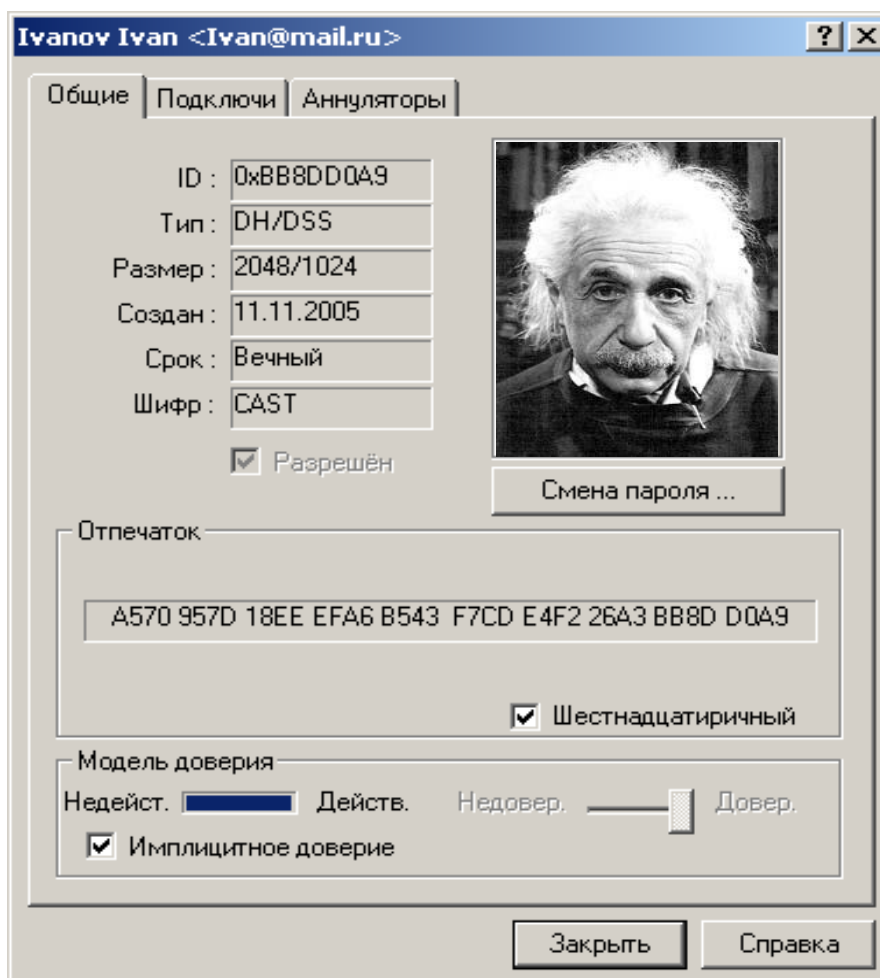


Рис. 2.12.

- Технические параметры ключа, а именно:

- *ID* – уникальный идентификационный номер ключа.
 - *Тип* – тип асимметричного ключа.
 - *Размер* – длина асимметричного ключа в битах.
 - *Создан* – дата создания ключа.
 - *Срок* – окончание срока действия ключа (*Вечный*, если не имеет ограничения).
 - *Шифр* – симметричный алгоритм, используемый для шифрования этим ключом.
- Опцию *Разрешен*, позволяющую активировать / деактивировать ключ.
 - Фото-удостоверение.
 - Кнопку *Смена пароля* для смены ключевой фразы либо *Соединение с ключом* для восстановления разделённого ключа в единое целое.
 - Параметры *Отпечаток* с отпечатком открытого ключа и *Модель доверия*, показывающий уровень достоверности открытого ключа и степень доверия его владельцу.

Регулярная смена ключевой фразы не является обязательной практикой для асимметричных ключей (важнее, чтобы она просто была очень надёжной): если кто-то получит ваш закрытый ключ в своё распоряжение, замена прежней ключевой фразы уже не избавит ключ от угрозы компрометации. Поэтому важно менять её только при угрозе компрометации самой ключевой фразы, например, если кто-то стоял у вас за спиной и, вероятно, мог подсмотреть за нажатиями клавиш, когда вы её набирали.

Но если злоумышленник уже похитил копию закрытого ключа, процедура смены пароля вас не спасёт. В этом случае немедленно изготовьте новую ключевую пару и перешифруйте все зашифрованные документы, файлы и корреспонденцию, уничтожив копии, зашифрованные скомпрометированным ключом.

Чтобы изменить текущую ключевую фразу:

1. Во вкладке *Общие* нажмите кнопку *Смена пароля*.

2. Введите текущую ключевую фразу и нажмите *ОК*.
3. В оба представленных поля введите новую ключевую фразу. Если хотите видеть, что набираете, снимите галочку с *Скрыть* (убедитесь, что в помещении нет посторонних). Нажмите *ОК*.

Если вы сменили ключевую фразу из-за подозрений её компрометации, после процедуры обязательно примите меры к уничтожению всех резервных копий своих связок и данной ключевой пары, а затем очистите свободное пространство диска, поскольку оставшиеся копии закрытого ключа по-прежнему защищены скопрометированной ключевой фразой!

2.3.2.2. Свойства подключей шифрования и их настройка

Каждая асимметричная ключевая пара по определению состоит из двух ключей: открытого и закрытого. В PGP версии 6.0 и выше появилась возможность создавать, удалять и аннулировать дополнительные подключи шифрования без необходимости жертвовать своей базовой ключевой парой и собранными на её сертификате подписями. В целом это похоже на превращение вашего базового ключа в своего рода связку с хранищимися на ней подключами. Существенное отличие лишь в том, что эти подключи используются только для зашифрования и расшифрования; для задач подписания информации служит только базовый закрытый ключ.

Основным назначением описанной функции является создание нескольких подключей шифрования, каждый из которых будет действовать в строго определённый период жизни базового ключа. Скажем, если вы сгенерировали базовый ключ со сроком жизни 3 года, можно создать ему три дополнительных подключа шифрования для каждого года жизни. Эта дополнительная система безопасности будет автоматически и регулярно заменять вам ключ шифрования без трудоёмкого процесса генерации и распространения нового открытого ключа. Но гораздо лучше не создавать

несколько подключей сразу, а добавлять каждый новый по мере необходимости, когда период действия текущего начинает подходить к концу. Так каждый новый подключ будет совершенно непредсказуем для взломщика, что в свою очередь, многократно повысит надёжность всей системы.

Для просмотра и настройки подключей шифрования в окне свойств откройте вкладку *Подключи* (рисунок 2.13).

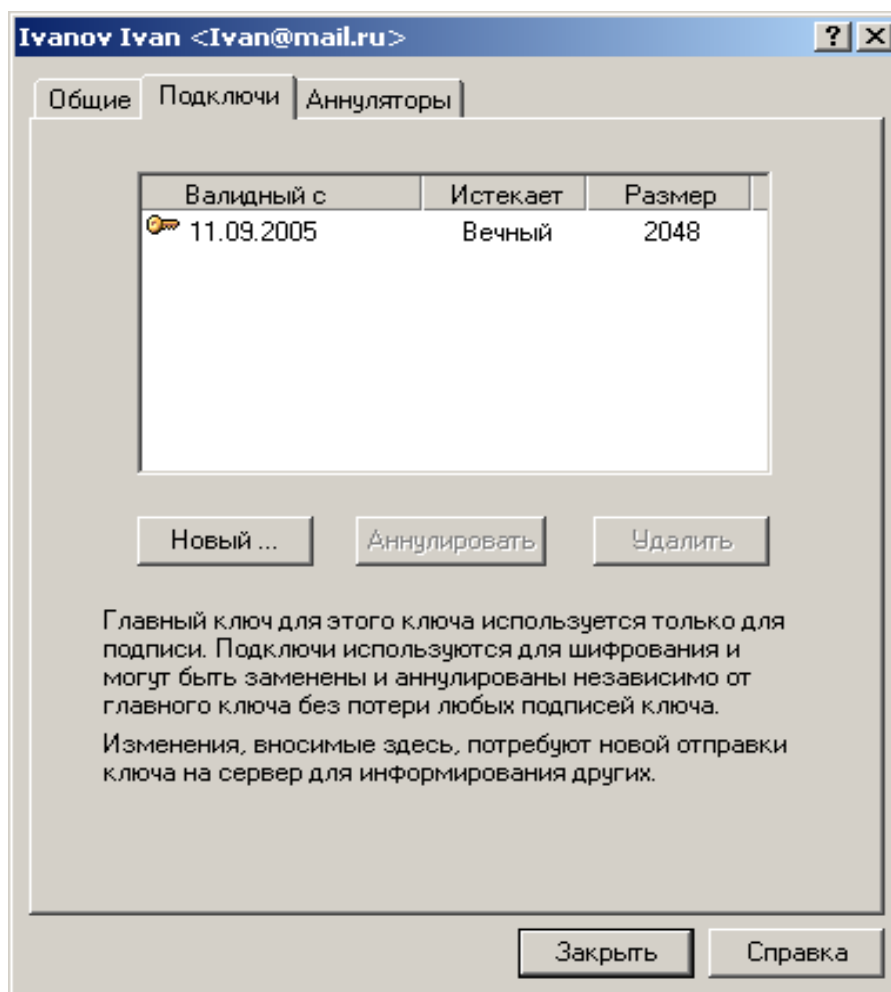


Рис. 2.13.

Чтобы создать новый подключ:

1. Во вкладке *Подключи* нажмите кнопку *Новый*.
2. В появившемся окне в поле *Размер ключа* введите необходимый размер под ключа в пределах 1024-4096 бит или выберите один из заданных в списке.

Не создавайте подключ меньшей длины, чем базовый ключ шифрования. Скажем, если ваш базовый ключ имеет длину 2048 бит, не создавайте подключи меньшего размера – это снизит стойкость вашей ключевой пары.

3. В поле *Начальная дата* укажите дату, когда данный подключ должен быть активирован.
4. Для параметра *Срок истекает* выберите либо *Никогда*, чтобы не ограничивать период действия подключа, либо *Дата* и укажите дату окончания его действия. Во избежание недоразумений при использовании нескольких подключей не допускайте совпадения, наложения и пересечения дат начала и окончания их действия.
5. Нажмите *ОК*.
6. Введите ключевую фразу и снова нажмите *ОК*.

Дополнительный подключ шифрования будет сгенерирован и добавлен к базовому ключу. Теперь вам нужно обновить ключ на сервере или самостоятельно передать его всем корреспондентам с тем, чтобы при шифровании они использовали новые сгенерированные подключи.

Если у вас возникли подозрения, что любой из подключей был скомпрометирован (обычно это относится к тому, который действует в настоящий момент), вы можете аннулировать его вместо аннулирования открытого ключа в целом.

Чтобы аннулировать подключ шифрования:

1. Во вкладке *Подключи* выделите нужный и нажмите кнопку *Аннулировать*.
2. PGP предупредит, что аннулирование подключа сделает невозможным зашифрование с его помощью любой информации. Если вы уверены в своих действиях, нажмите *Да*.
3. Введите ключевую фразу и нажмите *ОК*.

Обязательно обновите свой открытый ключ на сервере и разошлите соответствующие уведомления своим постоянным корреспондентам!

Чтобы удалить подключ с базового ключа:

1. Во вкладке *Подключи* выделите нужный и нажмите кнопку *Аннулировать*.
2. PGP предупредит, что удаление подключа носит необратимый характер и сделает невозможным расшифрование любой зашифрованной им информации (поскольку будет удалена и соответствующая часть с базового закрытого ключа).
3. Если действительно хотите это сделать, нажмите *Да*. *Обязательно обновите свой открытый ключ на сервере и разошлите обновлённые копии своим постоянным корреспондентам!*

2.3.2.3. Свойства "отменителя"

Вкладка *Аннуляторы* (рисунок 2.14) содержит список ключей, владельцы которых уполномочены при необходимости аннулировать данный открытый ключ.

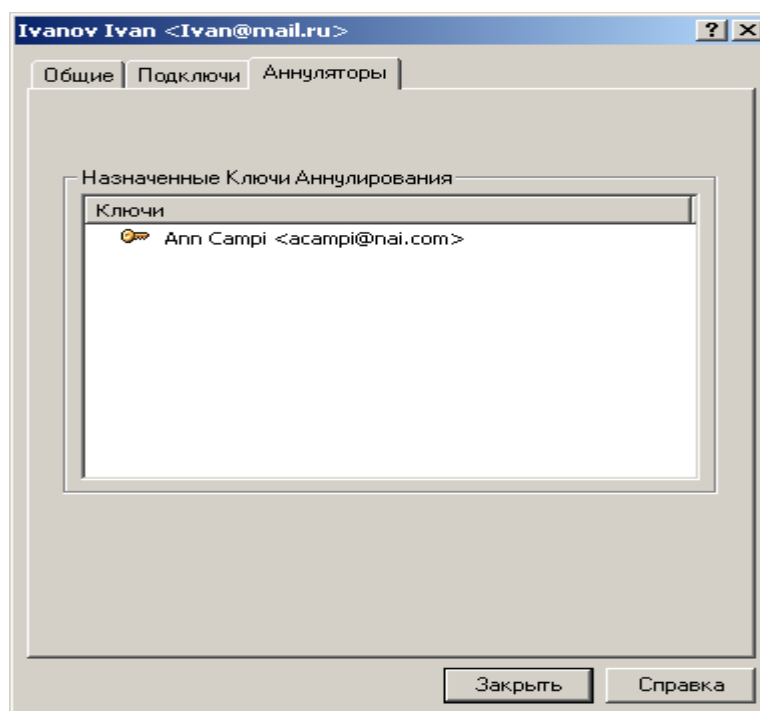


Рис. 2.14.

Если ключ "отменителя" отсутствует на вашей связке, он будет представлен строкой "*Неизвестный ключ*", за которой следует идентификационный номер. Выделите номер и нажмите кнопку *Update from Server*, чтобы загрузить копию открытого ключа с сервера.

Обязательно загружайте к себе на связку ключи всех уполномоченных вашими корреспондентами "отменителей". В противном случае PGP не сможет корректно проверять ключи на предмет "аннулированности" и в итоге вы можете использовать скомпрометированный ключ!

2.3.3. Сертификация открытых ключей

Хотя асимметричные криптосистемы являются лучшим решением для обмена ключами и зашифрованной информацией, они крайне уязвимы к атакам "человек в середине", когда злоумышленник пытается выдать свой поддельный открытый ключ за ключ вашего корреспондента, чтобы позднее перехватывать, читать и изменять пересылаемые сообщения. Взаимное заверение пользователями открытых ключей друг друга – это краеугольный камень распределённой модели доверия Web of Trust, лежащей в основе PGP и служащей мерой противодействия таким атакам.

Считать открытый ключ корреспондента априорно подлинным можно лишь в одном случае – если он вручил вам свой ключ на жёстком носителе при личной встрече или если очно передал вам отпечаток (не номер ID!) своего ключа. Но зачастую это невозможно, ведь через Интернет приходится общаться с людьми, живущими за тысячи километров. Специально для цели точной идентификации любого открытого ключа они снабжены так называемыми отпечатками. Цифровой отпечаток открытого ключа (fingerprint) – это хэш-значение его материала, столь же уникальное, сколь и сам ключ.

Лучший способ установить подлинность полученной вами копии открытого ключа корреспондента – позвонить ему и попросить прочитать отпечаток с оригинала, хранящегося на его связке (прочитать отпечаток должен

именно он вам, а не вы ему!). Маловероятно, что злоумышленник сможет перехватить такой произвольный звонок и провести активную атаку, попытавшись выдать себя за корреспондента. А если вам знаком голос корреспондента, это сделать будет практически невозможно.

Чтобы просмотреть отпечаток ключа, в менеджере PGPkeys нажмите правой кнопкой на имя ключа > *Свойства ключа*. Либо выделите нужный ключ и в панели инструментов нажмите кнопку *Показать свойства ключа или сертификата* (👤). В появившемся окне свойств ключа обратите внимание на *Отпечаток* (рисунок 2.15).

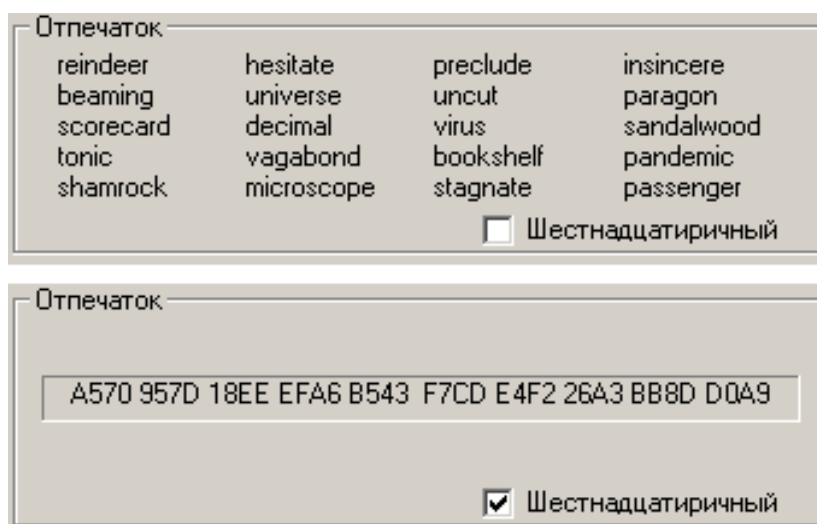


Рис. 2.15.

Отпечаток ключа может быть представлен в двух формах: в виде уникального списка слов или в виде уникальной буквенно-числовой последовательности.

По умолчанию отпечаток представлен уникальным списком т.н. биометрических слов. Эти слова по своему назначению аналогичны международному авиационному алфавиту (наверное, вы слышали в западных фильмах эти "альфа-зулу-фокстрот-гольф" и т.п.), предназначенному для безошибочной передачи буквенной информации по аудио-каналу с сильными помехами, но в отличие от того алфавита, содержащего всего 26 букв-слов,

биометрический словарь PGP включает 256 слов. Если вы решите сравнить отпечаток ключа, позвонив его владельцу, эти фонетически отчётливые слова позволят точно идентифицировать ключ даже по плохой междугородней связи и даже если вы или корреспондент не знаете английского языка.

Отметив галочкой опцию *Шестнадцатиричный*, вы отобразите отпечаток в виде шестнадцатеричного числа. Такой формат удобен для передачи отпечатка через Интернет или его размещения на своём веб-сайте. Можно выделить и копировать число, а затем вставить его в любой документ. Кроме того, отпечаток открытого ключа в шестнадцатеричной форме иногда печатают на обратной стороне визитных карточек.

Кроме непосредственно заверения чужого открытого ключа вы можете указать некоторый уровень доверия его владельцу в заверении других ключей и в выступлении в качестве их поручителя. Этот показатель считается вашим субъективным мнением о том, насколько данный пользователь компетентен в проверке подлинности открытых ключей, и насколько весомой вы считаете его подпись, заверяющую тот или иной ключ. Это значит, что если в будущем к вам в руки попадёт ключ, подписанный данным пользователем, он изначально будет для вас достоверным, хотя вы лично и не проверяли его подлинность. Поскольку показатель степени доверия является вашим субъективным конфиденциальным мнением, он не экспортируется вместе с ключом и действителен только на вашей связке. Чтобы установить степень доверия владельцу ключа убедитесь, что этот ключ вами подписан (экспортируемой или неэкспортируемой подписью). Затем:

1. В окне PGPkeys выделите ключ, и в строке меню выберите *Ключи > Свойства ключа*. В появившемся окне свойств ключа обратите внимание на раздел *Модель доверия* (рисунок 2.16).

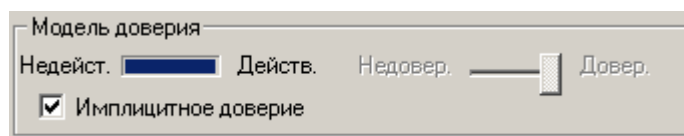


Рис. 2.16

2. Используйте регулятор *Уровень доверия*, чтобы установить нужный уровень доверия.

- По умолчанию уровень доверия ключа установлен на *Недовер*. (нет доверия): подписи этого ключа не будут приниматься в расчёт при вычислении достоверности других ключей.
- Если сдвинуть регулятор на средний уровень, подпись станет частично доверяемой и будет частично заверять другие ключи, т.е. одной этой подписи будет недостаточно, чтобы считать подписанные ключи достоверными: потребуется по меньшей мере две частично доверяемых подписи.
- Если вы считаете, что владелец этого ключа достаточно осторожен и с большой тщательностью проверяет подлинность подписываемых ключей, установите регулятор в положение *Довер*. (полное доверие), и подпись этого ключа будет заверять другие так же, как ваша собственная. Учтите, что ключ, заверенный вами с помощью подписи *Экспортируемая доверенного представителя* или *Неэкспортируемая псевдо-представителя*, уже имеет максимальный уровень доверия, снизить который невозможно.

3. Сделав выбор, закройте окно свойств, чтобы сохранить изменения.

Учтите, что вы не можете менять уровень доверия своих ключей, поскольку логика программы исходит из допущения, что вы полностью доверяете собственным действиям, и ставит вас в основу иерархии вашего дерева сертификации. Программа считает, что конкретный открытый ключ принадлежит вам, если находит на связке соответствующий ему закрытый. Полная ключевая пара всегда имеет уровень доверия *ИмPLICITное доверие* – безусловное доверие.

2.3.4. Редактирование сертификата ключа

Сертификат каждого открытого ключа PGP содержит по меньшей мере одну идентифицирующую запись (удостоверение), позволяющую соотнести ключ с владельцем или с одним из его реквизитов: адресом электронной почты, номером ICQ и пр. Новый только что сгенерированный ключ имеет лишь одну такую запись. Но если вы хотите использовать данный ключ для различных email-адресов и других средств связи, хотите добавить фото-удостоверение как дополнительный способ опознавания, то в любой момент можете это сделать.

Обычная запись сертификата OpenPGP включает имя или псевдоним владельца ключа и, по желанию, его email-адрес.

Чтобы добавить обычную запись в сертификат ключа:

1. В окне PGPkeys выделите нужный ключ, в строке меню нажмите *Ключи > Добавить > Имя*.
2. В появившемся окне *Имя нового пользователя* в поле *Новое имя для добавления к ключу* введите своё имя и в поле *Новый адрес для добавления к ключу* – адрес электронной почты. Нажмите *ОК*.
3. Введите ключевую фразу и снова *ОК*.

Новая идентифицирующая запись будет внесена в сертификат ключа. Если вы захотите сделать только что добавленную или любую другую запись сертификата главной (имя и email-адрес главной записи отображаются в имени ключа напротив иконки 🗉, а сама запись стоит первой в списке), выберите нужную, в строке меню нажмите *Ключи > Установить как основное имя* и введите ключевую фразу.

Не забывайте, что после редактирования открытого ключа или внесения любых изменений в содержание его сертификата, ключ нужно обновить на сервере.

Кроме адреса электронной почты сертификат ключа может включать любые другие идентификационные сведения.

Чтобы внести в сертификат иные записи с различными типами идентификации:

1. В окне PGPkeys выделите нужный ключ, в строке меню нажмите *Ключи* > *Добавить* > *Имя*.
2. В появившемся окне *Имя нового пользователя* в поле *Новое имя для добавления к ключу* введите своё имя и в поле *Новый адрес для добавления к ключу* – идентификационные сведения. Если добавляете свой номер ICQ для шифрования переговоров при помощи плагина, укажите ID в следующем формате: *ICQ:номер*. Нажмите *ОК*.
3. Введите ключевую фразу и снова *ОК*.

Не забудьте обновить ключ на сервере.

В PGP 6.0 и выше вы также можете добавить в сертификат ключа типа DH/DSS фотографическое удостоверение.

Никогда не опирайтесь на фото-удостоверения для определения подлинности полученного открытого ключа! Используйте их только для первичной идентификации.

Чтобы добавить фото-удостоверение:

1. В окне PGPkeys выделите нужный ключ, в строке меню нажмите *Ключи* > *Добавить* > *Фотографию* (рисунок 2.17).

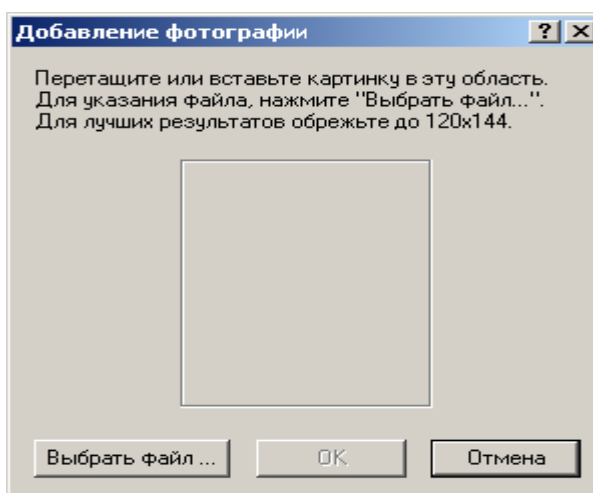


Рис. 2.17.

2. Можете добавить фотографию в появившееся окно *Добавление фотографии* тремя разными способами: её можно копировать и вставить (используя клавиши *Ctrl+C* / *Ctrl+V*), можно перетащить графический файл из Проводника, можно нажать кнопку *Выбрать файл* и выбрать его вручную. Так или иначе, используйте картинку в формате jpeg или bmp (в последнем случае программа конвертирует файл в jpeg автоматически) и, для лучшего качества, с габаритами 120x144 пикселей.

3. Выбрав картинку нажмите *ОК*.

4. Введите ключевую фразу и снова *ОК*.

Фотография будет добавлена к сертификату в виде отдельной записи. Теперь обновите ключ на сервере, чтобы каждый пользователь мог её увидеть в окне свойств полученной им копии вашего ключа.

Если хотите заменить текущую фотографию (при замене будут потеряны все подписи, заверяющие текущее фото-удостоверение):

1. В окне PGPkeys разверните список записей сертификата нужного ключа.

Найдите запись вида “ Фотография”.

2. Выделите эту запись, в строке меню нажмите *Правка > Удалить*.

3. Добавьте новую фотографию, как было описано выше. По окончании не забудьте обновить ключ на сервере.

2.3.5. Аннулирование ключа

Если по какой-то причине у вас возникнет подозрение или станет доподлинно известно, что ваша ключевая пара была скомпрометирована, нужно немедленно уведомить всех текущих и будущих корреспондентов не использовать данный открытый ключ для обмена информацией с вами, ибо велик шанс её попадания в чужие руки. Наиболее быстрый и удобный способ сделать это – выпустить сертификат аннулирования ключа (Key Revocation Certificate, KRC). Он будет присоединён к открытому ключу и аннулирует его,

а после импортирования этой копии ключа на связку корреспондент не сможет использовать его для зашифрования информации.

Аннулированная ключевая пара не может использоваться для зашифрования и подписания информации. Но ей можно продолжать пользоваться для расшифрования и сверки ЭЦП (в последнем случае вы всякий раз будете получать предупреждение, что ключ аннулирован).

Кроме аннулирования своих ключей вы можете отзываться с чужих ключей собственные сертифицирующие подписи, например, если посчитаете, что ключ скомпрометирован и вы более не можете гарантировать его целостность и принадлежность только изначальному владельцу, или если по иным причинам не захотите, чтобы другие пользователи полагались на вашу подтверждающую подпись. Аннулированная подпись никогда не берётся в расчёт при вычислении достоверности ключа.

Чтобы аннулировать свою ключевую пару или отозвать подтверждающую подпись с чужого сертификата:

1. В окне PGPkeys выделите нужный ключ или свою ЭЦП, и в строке меню нажмите *Ключи > Аннулировать*.
2. На просьбу подтвердить свои намерения ответьте *Да*, если действительно хотите это сделать.
3. Введите ключевую фразу (при отзыве подписи – ключевую фразу того ключа, которым ставили эту подпись) и нажмите *ОК*.

Ключевая пара или подтверждающая подпись будет аннулирована и отмечена иконкой 🗑️ или 🚫 соответственно. *Обязательно обновите ключ на сервере и разошлите уведомления с копиями ключа всем своим постоянным корреспондентам!*

Но может сложится и иная ситуация. Допустим, вы забудете свою ключевую фразу или потеряете закрытый ключ (например, после серьёзного системного сбоя). Без закрытого ключа и ключевой фразы вы не сможете издать

KRC, чтобы аннулировать открытый ключ и не допустить шифрование им информации, которую теперь тоже невозможно прочитать. Если вы не резервировали свой закрытый ключ, чтобы восстановить его в случае потери, описанный сценарий безнадёжен.

В качестве меры предосторожности можно уполномочить одного или нескольких доверенных человек в чрезвычайной ситуации аннулировать ваш ключ. Эти "отменители" смогут издать сертификат KRC собственными закрытыми ключами без всякого вмешательства с вашей стороны. (Эта функция поддерживается только в PGP 6.0 и выше для ключей типа DH/DSS.)

Чтобы добавить "отменителя":

1. В окне PGPkeys выделите нужный ключ, и в строке меню нажмите *Ключи > Добавить > Аннулирование*.
2. В появившемся списке ключей выделите те из них, владельцам которых хотите дать полномочия аннулирования. Нажмите *ОК*.
3. На просьбу подтвердить свои намерения ответьте *Да*, если действительно хотите это сделать.
4. Введите ключевую фразу и нажмите *ОК*.

Владельцы указанных вами ключей получат полномочия аннулирования и смогут аннулировать ваш ключ, как и любой собственный. Обязательно передайте им обновлённую копию своего ключа, а также отправьте его на сервер.

Также нужно отметить один крайне важный нюанс. Если ваш ключ был аннулирован "отменителем", то, чтобы у стороннего пользователя он выглядел таковым (🔒), и ваш открытый ключ, и открытый ключ "отменителя" должны присутствовать на его связке. Если ключа "отменителя" на связке пользователя нет, ваш аннулированный ключ будет казаться ему нормальным, и он будет продолжать шифровать им информацию. Поэтому ключ "отменителя" должен находиться в относительно широком распространении и, по меньшей мере, его копия должна храниться на общественном сервере-депозитарии.

Если уполномоченный отменитель недостаточно добросовестен и есть опасение, что он может злонамеренно аннулировать ваш ключ без всякой на то необходимости, можно поступить иначе. Небезынтересна такая схема: вы создаёте новую ключевую пару, которую добавляете к своему главному ключу в качестве доверенного отменителя. Открытый ключ этой новой пары вы отправляете на сервер, закрытый разделяете на несколько долей, каждую из которых отдаёте на хранение относительно доверенному человеку. В форс-мажорной ситуации все они по вашей просьбе реконструируют этот закрытый ключ и аннулируют им ваш собственный.

2.4. Работа с буфером обмена и активным окном

Хотя шифрование электронной почты более удобно осуществлять с помощью плагинов, далеко не все мэйл-клиенты поддерживают их, да и сами плагины отсутствуют в бесплатной freeware-версии PGP. Однако программа предоставляет ничуть не более сложный способ криптографирования текста – это работа с активным окном и с содержимым буфера обмена через PGPtray. Через PGPtray можно легко зашифровать, расшифровать или подписать любой текст, будь то электронное письмо или содержимое любого текстового файла, а использование комбинаций "горячих клавиш" сделает выполнение этих операций совершенно необременительным.

Два способа работы с текстом через PGPtray – активное окно и буфер обмена – в целом равнозначны. Но если для зашифрования текста в буфере обмена этот текст предварительно нужно туда скопировать, работа с содержимым активного окна более автоматизирована. Немного попрактикуйтесь, и вы сами почувствуете разницу и определите применимость каждого способа для решения тех или иных задач.

2.4.1. Зашифрование и подписание текста

Как правило, удобнее и проще шифровать текст с помощью функции активного окна. Если вам нужно зашифровать или подписать не всё содержимое окна, а только фрагмент находящегося там текста, достаточно его выделить и, оставив окно в фокусе, выполнить те же инструкции, что и для содержимого окна целиком.

1. Напишите своё письмо, как вы делаете это в обычных условиях. Можно использовать любой текстовый редактор, мэйл-клиент и даже веб-интерфейс почтовой службы. (Желательно отправлять важные сообщения с пустым заголовком темы, чтобы не давать потенциальному злоумышленнику даже малейшей информации о содержании письма.)
2. Составив письмо, выполните одно из следующих действий – они равноправны:
 - оставьте окно текстового редактора активным и нажмите на иконку PGPtray (🔒) > *Текущее окно* > *Зашифровать* (чтобы только зашифровать), *Подписать* (чтобы только подписать) или *Зашифровать и подписать* (чтобы одновременно подписать и зашифровать своё письмо);
 - оставьте окно текстового редактора активным и нажмите комбинацию "горячих клавиш", соответствующую требуемой операции с текстом;
 - выделите и копируйте текст в буфер обмена (Ctrl+C), нажмите на иконку PGPtray (🔒) > *Буфер обмена* > *требуемая операция с текстом* (опция *Очистить* так же позволяет очистить буфер обмена, а *Редактировать* – отредактировать его содержимое).
3. Если вы шифруете текст, а не только подписываете его, то в открывшемся окне выбора ключей получателей *Выбор ключа* (рисунок 2.18) укажите, для каких корреспондентов хотите зашифровать своё сообщение. Не удивляйтесь количеству пунктов в верхней части окна – там представлены не отдельные открытые ключи с вашей связки, а все

записи сертификатов этих ключей, которых может быть намного больше. Поэтому для каждого получателя сообщения достаточно указать всего одну запись с его ключа.

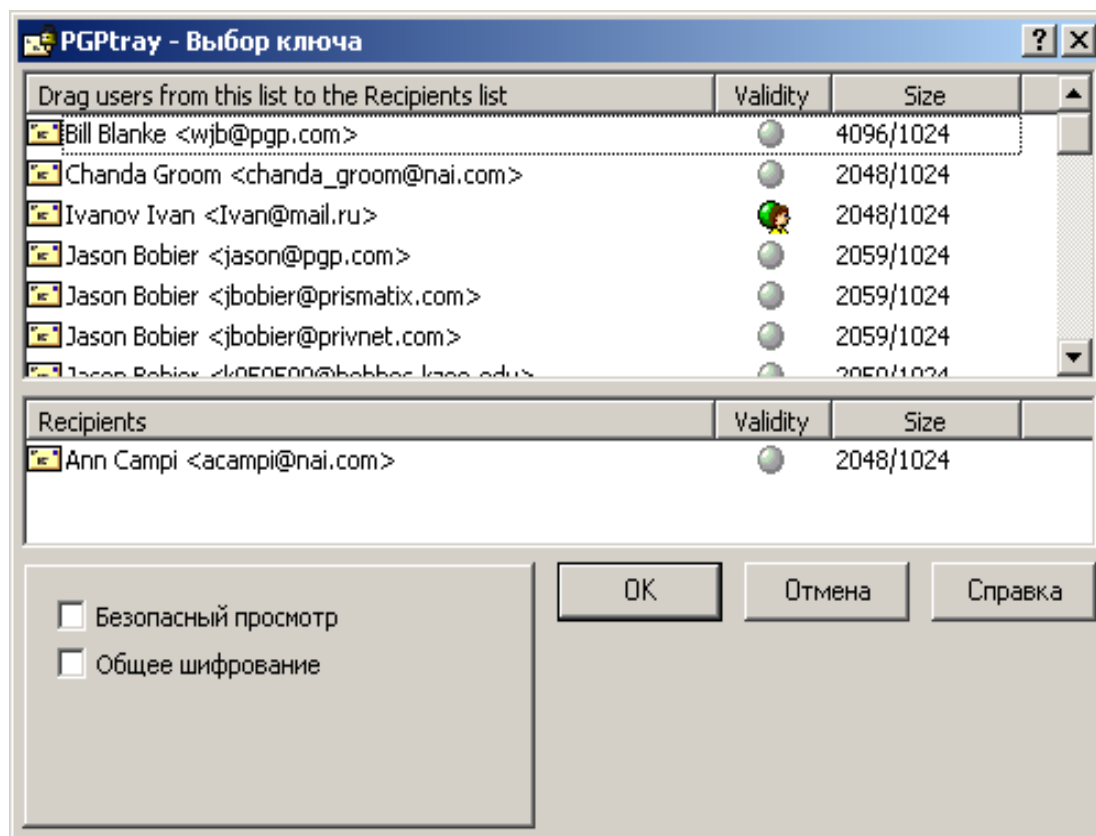


Рис. 2.18.

Перетащите в нижнюю часть окна те из них, для кого вы хотите зашифровать своё послание (можно не перетаскивать, а дважды щёлкнуть по любой записи, чтобы переместить её вниз или вверх). Если у вас выбран ключ "по умолчанию", то он изначально будет присутствовать в числе получателей сообщения. Не стоит его убирать, иначе впоследствии вы сами не сможете расшифровать отправленное письмо, чтобы, скажем, его перечитать.

Индикатор *Validity* показывает степень достоверности каждой из идентификационных записей. Крайне нежелательно шифровать сообщение теми ключами (и отправлять на те адреса), которые

отмечены как недостоверные. Проведите хотя бы самую элементарную и минимальную проверку подлинности данного ключа и соответствующей записи и подпишите её.

4. При необходимости отметьте дополнительные опции шифрования и нажмите *ОК*:

- *Безопасный просмотр* – отображать у получателя расшифрованный текст письма в специальном окне *Безопасный просмотр*, используя шрифт, предотвращающий так называемую TEMPEST-атаку (удалённый съём информации по электромагнитному излучению монитора); кроме того, в этом случае сообщение невозможно будет сохранить в виде открытого текста. Разумно использовать для сообщений крайней секретности, но не для повседневных писем (ещё и потому, что этот поставляемый с PGP TEMPEST-защитный шрифт не имеет кириллических символов, и, соответственно, письмо придётся писать латиницей). Также имейте в виду, что в PGP до 6.0 поддержка функции *Безопасный просмотр* отсутствует, и если получатель использует более раннюю версию программы, она выбранную вами опцию просто проигнорирует.
- *Общее шифрование* – симметричное шифрование паролем вместо открытого ключа. При выборе этой опции программа предложит ввести ключевую фразу, которая потребуется и для расшифрования текста. Понятно, что использовать эту функцию для пересылки электронной почты нерационально – криптография с открытым ключом для этой цели практичнее и удобнее. Однако шифрование простым паролем может пригодиться для защиты документов, хранящихся на вашем собственном диске. (С технической точки зрения программа использует введённый пароль для шифрования сеансового ключа, которым шифруется само сообщение по

алгоритму, указанному как *Назначенный алгоритм* в настройках PGP.)

5. Если вы электронно подписываете текст, а не только шифруете его, программа попросит ввести ключевую фразу вашего закрытого ключа (если у вас несколько ключевых пар, в меню *Ключи подписи* можно выбрать ключ для подписания).

Учтите, что если у вас выбран ключ "по умолчанию", и его ключевая фраза в данный момент хранится в кэше, этот запрос не появится – PGP автоматически подпишет текст вашим основным ключом. Чтобы выбрать другой ключ вам потребуется предварительно удалить ключевую фразу из оперативной памяти (*PGPtray > Буфер обмена > Очистить*).

PGP зашифрует / подпишет текст и заменит оригинал криптографированным материалом (рисунок 2.19). (Если вместо работы с активным окном вы предпочли работу с буфером обмена, вам самим придётся вставить обработанный текст в нужное приложение.)

```
-----BEGIN PGP MESSAGE-----
Version: PGPfreeware 6.5.8 for non-commercial use <http://www.pgp.com>
Comment: Так будет выглядеть зашифрованное сообщение

qANQR1DBwU4DjdJ0ointlicQCADpr4Q6iSkbMzZ0MTfTHSGffpiFv7nPMUmO90E1
olIbO8yld7i2qqCE3PvcWd8jGEvQrtOk0pW2ACslJaVfLaVr45CAE7YX/BCGismx
rq9fvZdl0RhlyeVcbqyAn5i4RRVJzKugQRyVxYaJfMrJpObqVfbj76JELfnQZTFV
bR0bQd2JLLz/6mRN/zqAZf/LLJRVpOAhjM6JQQ1jZ+PmIID9tpXlWWCJ0sHuHgze
qJdTGChTlIA1QtCgssmPQwIAOgcPI+laCeJ7b7ZR/agyAGlk2TXifnBch/8wyiLN
EbLeywjxbbkWSR1l/Ce2/Kzg7z5mhGKjr7ddrLrN+DG7VrJ7B/9Lo3fgeehFMJ7o
iteuOS7xvflSKbMUMRrPBAEb0mOmRjyMgzNFxJLY3raaTIPSwjmaLzn/4rnFYyxr
yIm8r2Vts0C/50PD1N5j59UEXWI5jbjbb4Tn/49jAISTiEhrUmAG2SQU8GMIAfr
JFyYrLoutvhdPR4DR2o/K2zjT23YooWuH2Cqw05SSnex2I84IZMt80ljfQ6NshKV
NCXvJ0DA1gUKKbs0o/7GdKfdb7D/TIAKrptBrujYNlli/OARaeA9p7rygZXXy0gj
j16vPmQ9hw8phjENM21PnxRTWS5rKcC6mcs+eRQoTsL9GtyLn4gzrhrNRLTt23pF
cdK1qP60ycGhMO7aJLo2aLVruBYu5rX1kqkgCox/r1bexHmWqBEpYqRZ8+wRhgdu
nk4RFFe4/qSVK343CeHZiedOIDBCXAi3a20arH6XU+gZQQ+1IKNSws6vw8bOnf9S
WZ3f/IuZTZDpZbU6FymkPvRv+6sSsbAugff4asw5PD8rEXBrE74EZNUMSPTzyWxO
IbkCD/s7nTbOlVQQJu0iINHxza2eU3Qs19CjfADzFYehVkpGWv/CP4KrhzFYFuZR
yTKtPqaiu5NzY1S/hDxcjYTYfnAja3RLlGc2JWNsIqJ0kX7vweNo8oApI+nLxlaO
SvZ8yoheelMcU4TK02jWRKXSSKMQLYfliHWiITU6zHbefGpvqkhKTvtSelQRNG6
nIPgyYB3/33psJgBUCq0Ih1rvYIU09gmyjsDNB/VX0EgCWKmoakduUWbREEDgXNr
Jxdj5ZDmBBnfBMCEg6Jgo40hiYroh+Z1IrhLTGNd9iKCENlCz6Tpv5FI6vilwYpi
9GZtepAkD9RSdC6Puu2esi4uGWHmGkb/dl7DKn2zZ9xMxzzjI8kViIYzsKxVzDb/M
bNFQL24pQndFzwkKdDIi2sXAE/fZradtbs7caSsx2dBU+wc7UANGoTbWp6p+r7
m0VL2ioUwPodvc09zAxB3TdLPGu9/n4qFh0fZHzSpw+Hom77aQerE1KhbkQs34KT
X67eVp0Upm0OIPWJgQW3U8VMksYxFgCuUrHGZwAV6N3ZtFUCqSAF52zdHgOYV1GJ
38U8zZtJqQ3nkpwIC6wJCX3NOSAjqZaR1TGT7Nctfx+JV6NYSBRxv5wa
=G8km
-----END PGP MESSAGE-----
```

Рис. 2.19

2.4.2. Расшифрование и сверка ЭЦП

Расшифрование текста и сверка цифровой подписи особенно легки с использованием "горячих клавиш" – достаточно пары нажатий. В остальных случаях, как и с шифрованием, можно использовать PGPtray. Если вы хотите расшифровать сообщение через буфер обмена, нужно выделить весь блок шифртекста вместе с заголовками типа “-----BEGIN PGP MESSAGE-----”, “-----END PGP SIGNATURE-----” и др. и целиком копировать его в буфер обмена.

Как правило, попытка расшифровать сообщение через активное окно прямо со страницы веб-сайта приводит к ошибке. Так случается потому, что веб-страницы обычно содержат графику и множество других нетекстовых элементов. Но чтобы не копировать шифртекст в буфер обмена можно пойти на маленькую хитрость: достаточно выделить шифртекст в окне браузера и затем нажать комбинацию клавиш для расшифрования или указать эту же команду через меню работы с активным окном в PGPtray.

Чтобы расшифровать текстовое сообщение и/или сверить его электронную подпись:

1. Откройте сообщение в своём мэйл-клиенте (если это письмо), текстовом редакторе или веб-браузере. Если оно зашифровано, а не только подписано, вы увидите лишь нечитаемый шифртекст.
2. Выполните одно из следующих действий – они равноправны:
 - оставьте окно текстового редактора активным и нажмите на иконку PGPtray (🔒) > *Текущее окно* > *Расшифровать и Проверить* (чтобы расшифровать сообщение и/или сверить ЭЦП);
 - оставьте окно текстового редактора активным и нажмите комбинацию "горячих клавиш", соответствующую операции расшифрования;
 - выделите блок шифртекста вместе с заголовками и копируйте его в буфер обмена (Ctrl+C), нажмите на иконку PGPtray (🔒) > *Буфер обмена* > *Расшифровать и Проверить*.

3. Появится окно *Введите пароль* со списком открытых ключей, которыми зашифровано сообщение. Если сообщение предназначено вам, программа попросит ввести ключевую фразу одного из ваших закрытых ключей. Если же вместо поля для ввода ключевой фразы в окошке указана ошибка "*Невозможно расшифровать данное сообщение ...*", причина этого в том, что у вас на связке нет нужного для расшифрования закрытого ключа (он мог быть удалён, либо отправитель намеренно или случайно не зашифровал письмо вашим открытым ключом).

4. Введите ключевую фразу и нажмите *ОК*.

Программа расшифрует сообщение и отобразит результат в окошке *Просмотр текста*. Если сообщение было подписано, там же будет указано состояние цифровой подписи.

Если при зашифровании отправитель отметил опцию *Безопасный просмотр*, PGP выдаст предупреждение, что письмо предназначено только для ваших глаз и его стоит читать с соблюдением максимальных мер предосторожности. Когда будете готовы открыть сообщение, нажмите *ОК*. Текст будет выведен в окне *Безопасный просмотр* с помощью специального TEMPEST-защитного шрифта. (Имейте в виду, что ни копировать его, ни сохранить в расшифрованном виде вам не удастся.)

Если полученное сообщение было подписано, программа также сообщит вам некоторые сведения о цифровой подписи (рисунок 2.20).

```
*** СТАТУС ПОДПИСИ PGP: хороший
*** Подписыватель: Ivanov Ivan <Ivan@mail.ru>
*** Подписано: 17.12.04 13:21:52
*** Проверено: 19.12.04 5:58:28
*** НАЧАЛО РАСШИФРОВАННОГО/ПРОВЕРЕННОГО СООБЩЕНИЯ PGP ***
```

Текст подписанного сообщения

```
*** КОНЕЦ РАСШИФРОВАННОГО/ПРОВЕРЕННОГО СООБЩЕНИЯ PGP ***
```

Рис. 2.20.

Это выглядит как набор заголовков, где в строке *Подписано* указана дата подписания (относительно вашего часового пояса), в *Проверено* – дата сличения подписи (т.е. текущий момент), в *Подписыватель* – имя владельца ключа, которым была поставлена подпись, а в *Статус подписи PGP* – собственно, состояние подписи:

- *Good* – информация получена вами ровно в том виде, в каком была подписана и отправлена автором.
- *Bad* – подписанная информация была каким-то образом изменена (искажена). Причиной тому могло послужить не только злонамеренное вмешательство, но и более тривиальные вещи, например, плохое качество связи, повлекшее искажение информации в процессе передачи, случайное редактирование сообщения автором уже после подписания, изменение, внесённое почтовой программой отправителя или вашей. В любом случае, к подобного рода сообщениям следует относиться с большой осторожностью; желательно также в кратчайшие сроки выяснить причину происшедшего.
- *Unknown* – это говорит о том, что на вашей связке ключей отсутствует тот, которым информация была подписана, и, следовательно, программа не может сверить подпись. В таких ситуациях PGP пытается самостоятельно связаться с сервером-депозитарием, чтобы найти соответствующий открытый ключ (если в настройках программы во вкладке *Серверы* включена опция *Удостоверение*).
- *Invalid* – так PGP уведомит вас о том, что ключ автора сообщения есть на вашей связке, но не признан подлинным, и, соответственно, программа не может оценить целостность подписанной информации. Вам нужно проверить достоверность ключа и заверить его.

При сличении подписи обязательно проверяйте, чтобы дата/время, указанные в строке *Проверено*, совпадали с текущими показаниями системных

часов! Если вы не будете этого делать либо будете делать недостаточно тщательно, злоумышленник сможет одурачить вас, подсунув сформированное определённым образом составное сообщение, при расшифровании выглядящее так, словно было подписано одним из ваших доверенных корреспондентов. Но поскольку мошенник не может доподлинно знать с точностью до секунд, в какой момент времени вы откроете письмо, указанная им в строке *Проверено* дата будет иметь расхождение с реальной датой расшифрования сообщения (т.е. с текущим моментом).

Учтите, что метка времени, стоящая в строке *Подписано*, указывает время системного таймера отправителя. Отправителю эту метку крайне легко сфальсифицировать – достаточно перед подписанием перевести системные часы.

2.5. Защита файлов

PGP позволяет шифровать не только текст, но и файлы для их безопасного хранения на диске или для пересылки в качестве вложений к электронным письмам. Зашифрованный файл можно без опасений размещать и в Интернете, поскольку никто кроме владельца соответствующего закрытого ключа не сможет узнать его содержание.

При шифровании папки она не будет зашифрована целиком; напротив, PGP индивидуально зашифрует находящиеся в ней файлы. Чтобы зашифровать папку с сохранением структуры каталогов, стоит предварительно упаковать её в архив (например, с помощью WinZip) и уже затем зашифровать сам архивный файл.

Плагин PGP в некоторых мэйл-клиентах не способен автоматически зашифровывать и расшифровывать вложения электронной почты. Поэтому чтобы безопасно переслать файл или извлечь его из полученного письма вам придётся вручную выполнить с ним описанные ниже действия.

Зашифрование и подписание файлов можно производить двумя способами: через контекстное меню в Проводнике Windows или с помощью утилиты PGPtools, которую можно вызвать через PGPtray. Расшифровывать же файлы проще всего в Проводнике, просто дважды щёлкнув на имя зашифрованного файла; вам понадобится только ввести нужную ключевую фразу, и в том же каталоге появится расшифрованная копия этого файла.

Чтобы выполнить ту или иную операцию над файлом с помощью PGPtools, можно либо нажать на соответствующую кнопку и указать этот файл в меню *Обзор*, либо перетащить этот файл из Проводника на нужную кнопку инструмента. Кроме того, в меню *Обзор*, вызываемом с помощью этих кнопок, присутствует опция *Буфер обмена*, позволяющая выполнить данную операцию над содержимым буфера обмена. Назначение кнопок инструмента следующее (рисунок 2.4, слева направо):

- *PGPkeys* – открыть менеджер ключей;
- *Encrypt* – зашифровать файл;
- *Sign* – подписать файл;
- *Encrypt and Sign* – подписать и зашифровать файл;
- *Decrypt / Verify* – расшифровать файл и / или сверить цифровую подпись;
- *Wipe* – стереть файл с диска (уничтожить без возможности восстановления);
- *Freespace Wipe* – очистить диск от фрагментов прежде удалённых файлов.

Нажав в верхнем левом углу окошка и выбрав *Stay On Top*, вы заставите инструмент находиться поверх остальных окон.

Если вы зашифровываете файл, а не только подписываете его, то откроется окно выбора ключей получателей *Выбор ключа* (рисунок 2.21), где нужно указать, для каких корреспондентов вы хотите зашифровать этот файл. Не удивляйтесь количеству пунктов в верхней части окна – там представлены не отдельные открытые ключи с вашей связки, а все записи сертификатов этих

ключей, которых может быть намного больше. Поэтому для каждого получателя файла достаточно указать всего одну запись с его ключа.

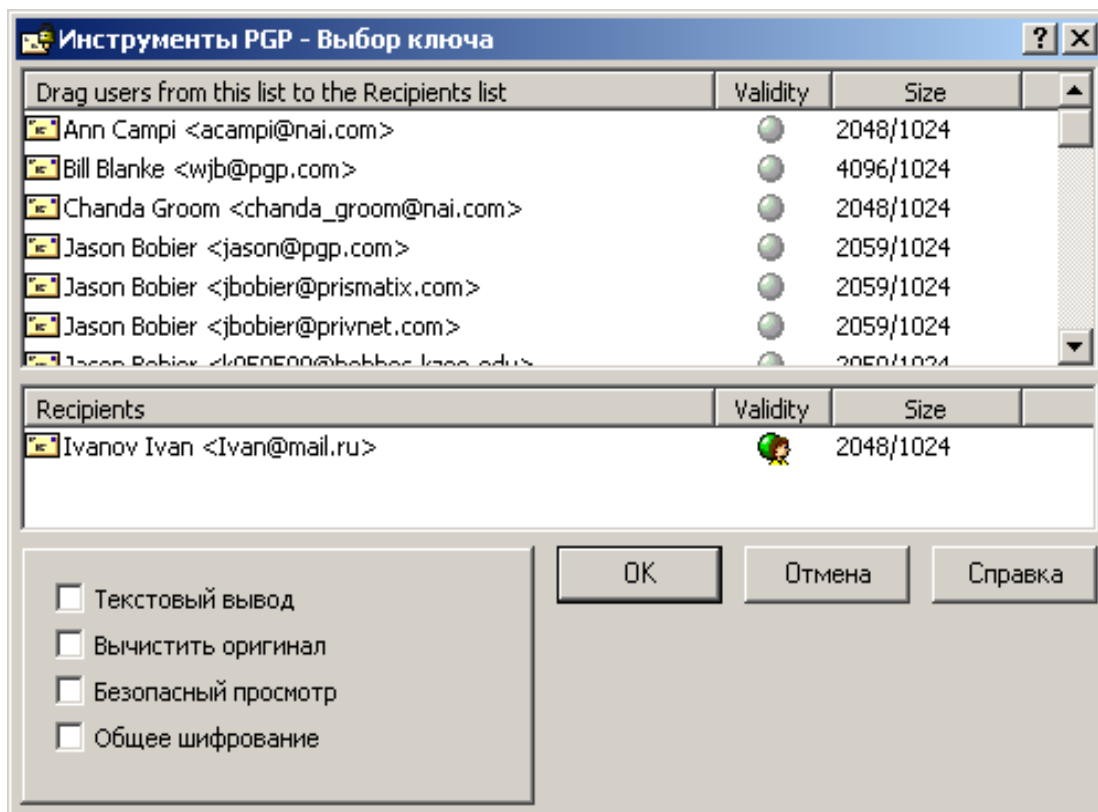


Рис. 2.21.

Перетащите в нижнюю часть окна те из них, для кого вы хотите зашифровать файл (можно не перетаскивать, а дважды щёлкнуть по любой записи, чтобы переместить её вниз или вверх). Если у вас выбран ключ "по умолчанию", то он изначально будет присутствовать в числе получателей. Не стоит его убирать, иначе впоследствии вы сами не сможете расшифровать отправленный файл, и ни в коем случае не убирайте его, если шифруете файл для сохранения у себя на диске (только если ни собираетесь заменить его другим своим ключом).

Индикатор *Validity* показывает степень достоверности каждой из идентификационных записей. Крайне нежелательно шифровать файл теми ключами (и отправлять на те адреса), которые отмечены как недостоверные. Проведите хотя бы самую элементарную и минимальную проверку подлинности данного ключа и соответствующей записи и подпишите её.

При необходимости отметьте дополнительные опции шифрования и нажмите *ОК*:

- *Текстовый вывод* – сохранить шифртекст не в двоичном, а в ASCII-формате (в текстовом виде). Некоторые старые мэйл-клиенты не позволяют пересылать двоичный код, и эта опция может оказаться полезной. Кроме того, таким образом можно отправлять файл не вложением, а прямо в теле письма, если открыть зашифрованный файл любым текстовым редактором и скопировать содержимое в письмо. Учтите, однако, что использование этой опции увеличит объём файла примерно на 30% (в сравнении с шифрованием с выключенной опцией, а не с исходным файлом).
- *Вычистить оригинал* – уничтожить исходный файл с открытым текстом после зашифрования, перезаписав его на диске случайными данными. Так, файл останется только в виде шифртекста, а оригинал будет более недоступен.
- *Безопасный просмотр*.
- *Общее шифрование* – симметричное шифрование паролем вместо открытого ключа. При выборе этой опции программа предложит ввести ключевую фразу, которая потребуется и для расшифрования файла. Понятно, что использовать эту функцию для пересылки файла по электронной почте нерационально – криптография с открытым ключом для этой цели практичнее и удобнее. Однако шифрование простым паролем может пригодиться для защиты файлов, сохраняемых на вашем собственном диске. (С технической точки зрения программа использует введённый пароль для шифрования сеансового ключа, которым шифруется файл по алгоритму, указанному как *Назначенный алгоритм* в настройках PGP.)

Если вы электронно подписываете файл, а не только шифруете его, программа попросит ввести ключевую фразу вашего закрытого ключа (если у




у вас несколько ключевых пар, в меню *Ключи подписи* можно выбрать ключ для подписания). Учтите, что если у вас выбран ключ "по умолчанию", и его ключевая фраза в данный момент хранится в кэше, этот запрос не появится – PGP автоматически подпишет файл вашим основным ключом. Чтобы выбрать другой ключ вам потребуется предварительно удалить ключевую фразу из оперативной памяти (*PGPTray > Буфер обмена > Очистить*).

Если вы только подписываете файл, окно *Введите пароль* будет содержать несколько дополнительных опций:

- *Нарушенная подпись* – изготовить "съёмную" цифровую подпись. Если опция включена (а она включена по умолчанию), цифровая подпись будет сохранена в виде отдельного крохотного файла, имеющего такое же имя, что и у подписанного файла, но с расширением.sig. Такой файл-подпись можно передавать и публиковать отдельно от подписанного, дабы не усложнять использование подписанного файла людям, не пользующимся PGP. Если опцию выключить, файл будет сохранён, как при обычном шифровании, и использовать его без сверки ЭЦП будет невозможно.
- *Текстовый вывод* – сохранить "съёмную" ЭЦП или подписанный файл (в зависимости от предыдущей настройки) не в двоичном, а в ASCII-формате (в текстовом виде). Некоторые старые мэйл-клиенты не позволяют пересылать двоичный код, и эта опция может оказаться полезной. Кроме того, таким образом можно отправлять файл не вложением, а прямо в теле письма, если открыть зашифрованный файл любым текстовым редактором и скопировать содержимое в письмо. Учтите, однако, что использование этой опции увеличит объём подписанного файла примерно на 30% (в сравнении с шифрованием с выключенной опцией, а не с исходным файлом); для "съёмной" ЭЦП это несущественно.

PGP зашифрует / подпишет файл и сохранит эту копию в том же каталоге, что и исходный файл (если при зашифровании была отмечена опция *Вычислить оригинал*, PGP уничтожит исходную копию с открытым текстом). В зависимости от характера содержимого зашифрованного файла он будет представлен одним из трех значков (таблица 2.5).

Таблица 2.5

	Файл, зашифрованный с опцией <i>Текстовый вывод</i> (шифртекст в ASCII-формате, asc-файл).
	Обычный зашифрованный файл (шифртекст в двоичном формате, pgp-файл).
	"Съёмная" цифровая подпись (sig-файл).

Чтобы расшифровать файл и/или сверить ЭЦП, достаточно дважды щёлкнуть в Проводнике на имя зашифрованного файла или на sig-файл. Если файл был зашифрован, появится окно *Введите пароль* со списком открытых ключей получателей. Если файл предназначен вам, программа попросит ввести ключевую фразу одного из ваших закрытых ключей. Если же вместо поля для ввода ключевой фразы в окошке указана ошибка *"Невозможно расшифровать данное сообщение ..."*, причина этого в том, что у вас на связке нет нужного для расшифрования закрытого ключа (он мог быть удалён, либо отправитель намеренно или случайно не зашифровал файл вашим открытым ключом). Введите ключевую фразу и нажмите *ОК*. Программа расшифрует исходный файл и сохранит его в одном каталоге с шифртекстом.

Если файл был только подписан, вы сразу увидите окно отчёта PGPlog, содержащее некоторые сведения о цифровой подписи. Так, в столбце *Name* указано имя подписанного файла, в *Signer* – имя владельца ключа, которым была

поставлена подпись, в *Validity* – степень достоверности ключа подписания, а в *Signed* – состояние ключа подписания (дезактивирован, аннулирован и т.п.) и самой ЭЦП: если подпись корректна, и файл был получен ровно в том виде, в как был подписан отправителем, здесь будет указана дата подписания (относительно вашего часового пояса), в иных случаях будет отмечено одно из следующих значений:

- *Bad signature* – подписанный файл был каким-то образом изменён (искажён). Причиной тому могло послужить не только злонамеренное вмешательство, но и более тривиальные вещи, например, плохое качество связи, повлекшее искажение информации в процессе передачи, случайное редактирование файла автором уже после подписания, изменение, внесённое почтовой программой отправителя или вашей. В любом случае, к подобного рода файлам следует относиться с большой осторожностью; желательно также в кратчайшие сроки выяснить причину происшедшего.
- *Unknown signing key* – это говорит о том, что на вашей связке ключей отсутствует тот, которым файл был подписан, и, следовательно, программа не может сверить подпись. В таких ситуациях PGP пытается самостоятельно связаться с сервером-депозитарием, чтобы найти соответствующий открытый ключ (если в настройках программы во вкладке *Серверы* включена опция *Удостоверение*).
- *Invalid key* – так PGP уведомит вас о том, что ключ отправителя файла есть на вашей связке, но не признан подлинным, и, соответственно, программа не может оценить целостность подписанной информации. Вам нужно проверить достоверность ключа и заверить его.

Учтите, что метка времени, стоящая в строке *Signed*, указывает время системного таймера отправителя. Отправителю эту метку крайне легко сфальсифицировать – достаточно перед подписанием перевести системные часы.

2.6. Уничтожение данных



Создавая и удаляя важные документы обычными средствами операционной системы, вы оставляете информацию, содержащуюся в удалённых файлах, лежать в освободившемся пространстве жёсткого диска. Когда вы удаляете файл, помещая его в Корзину, он по сути просто перемещается из одного каталога в другой. Но и очистив Корзину, вы не сотрёте файл с диска окончательно, пока ОС не перезапишет высвобожденные сектора (это может случиться очень нескоро). Более того, многие программы и почти все текстовые процессоры создают в ходе работы множество резервных копий редактируемых документов. Хотя эти копии удаляются программой по завершении работы, содержащаяся в них информация по-прежнему остаётся записанной на диске. В общем смысле, ценные файлы никогда не исчезают полностью, и при наличии должных инструментов могут быть восстановлены в первоначальный вид.

Если вам нужно безвозвратно стереть файл с лица жёсткого диска, используйте утилиту PGP Wipe. Она удаляет файл, многократно перезаписывая сектора диска, в которых он находился, случайными данными ("мусором") так, что его не удастся восстановить даже самыми совершенными коммерческими средствами.

Для очистки всего диска от остатков прежде удалённых файлов предназначена утилита PGP FreeSpace Wipe. Во-первых, она перезаписывает все свободные сектора диска, уничтожая фрагменты невостребованной информации, во-вторых, очищает частично занятые сектора диска в "хвостах" хранящихся на диске файлов (так называемый slack space).


Эти файловые системы сохраняют в своей специальной внутренней области резервные копии всех записей, вносимых ОС в файловую систему – ведут журнал изменений. Такой журнал представляет собой последовательное описание всех изменений, произведённых на диске, и служит цели надёжного

восстановления ФС и содержимого диска после системных сбоев, однако, и усложняет задачу надёжного уничтожения данных. Стирание файла с помощью PGP Wipe не способно удалить все журнальные записи, вероятно сделанные файловой системой.


Обе утилиты можно вызвать из PGPtools с помощью соответствующих кнопок в правой части инструмента:  для PGP Wipe и  для Freespace Wipe. Кроме того, функция PGP Wipe доступна из контекстного меню в Проводнике Windows по нажатию правой кнопки на имени файла или папки, а также через интерфейс шифрования файлов.

Чтобы надёжно стереть файл с жёсткого диска:

1. Выделите в Проводнике файлы и/или папки, подлежащие уничтожению.
2. Нажмите правой кнопкой мыши на выделенные объекты, выберите в контекстном меню пункт *PGP > Wipe*.
3. В появившемся окне с перечнем удаляемых файлов / папок нажмите *Да*, чтобы подтвердить свой выбор, или *Нет*, чтобы отказаться от уничтожения этих файлов. Не забывайте, это последняя возможность передумать.

Ту же операцию можно проделать с помощью PGPtools: для этого нажмите кнопку *Wipe* () , укажите файлы, подлежащие уничтожению (или перетащите файлы на эту кнопку из Проводника), и подтвердите свой выбор, нажав *Да*.

Чтобы очистить свободное пространство диска от остатков удалённых файлов с помощью утилиты PGP Freespace Wipe:

1. В окне PGPtools нажмите кнопку *Freespace Wipe* (). Появится окно мастера с приветственным сообщением (рисунок 2.22).

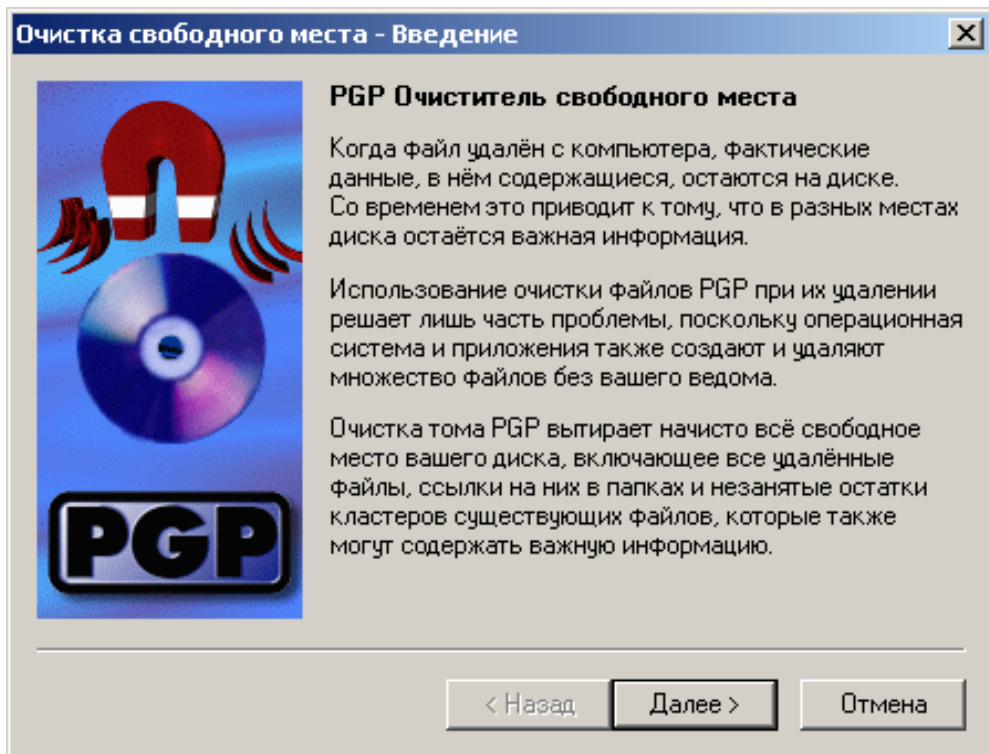


Рис. 2.22.

2. Нажмите *Далее*.

3. Укажите параметры данного сеанса очистки (рисунок 2.23). В поле *Очистить диск* выберите диск, подлежащий очистке, и введите число проходов очистки (количество перезаписей свободного пространства диска).

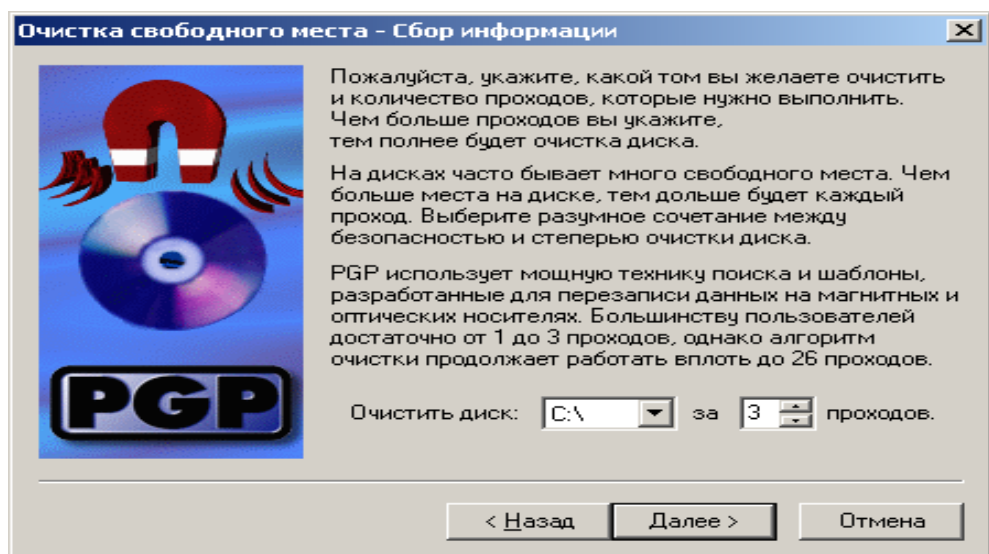


Рис. 2.23.

4. Чтобы продолжить, нажмите *Далее*.

На следующей странице мастера (рисунок 2.24) будут отображены технические сведения о выбранном для очистки разделе диска и график выполнения операции.

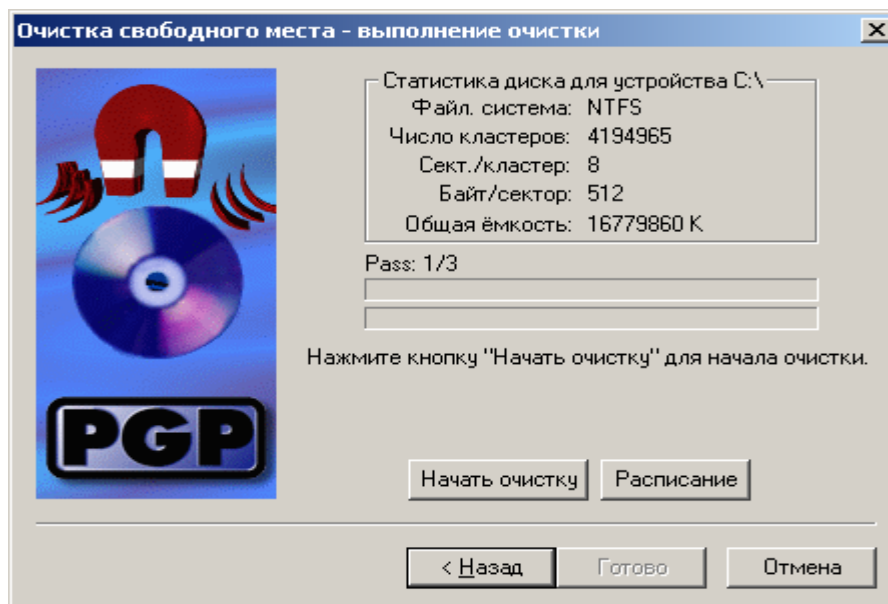


Рис. 2.24.

5. Нажмите кнопку *Начать очистку*, чтобы приступить к очистке свободного пространства. Ход очистки можно прервать в любой момент, нажав кнопку *Отмена*. Однако это оставит на диске не перезаписанные фрагменты файлов.

По окончании процесса очистки в нижней части окна появится сообщение *Поздравляем! Диск был очищен*.

6. Для завершения работы мастера нажмите *Готово* (рисунок 2.25).

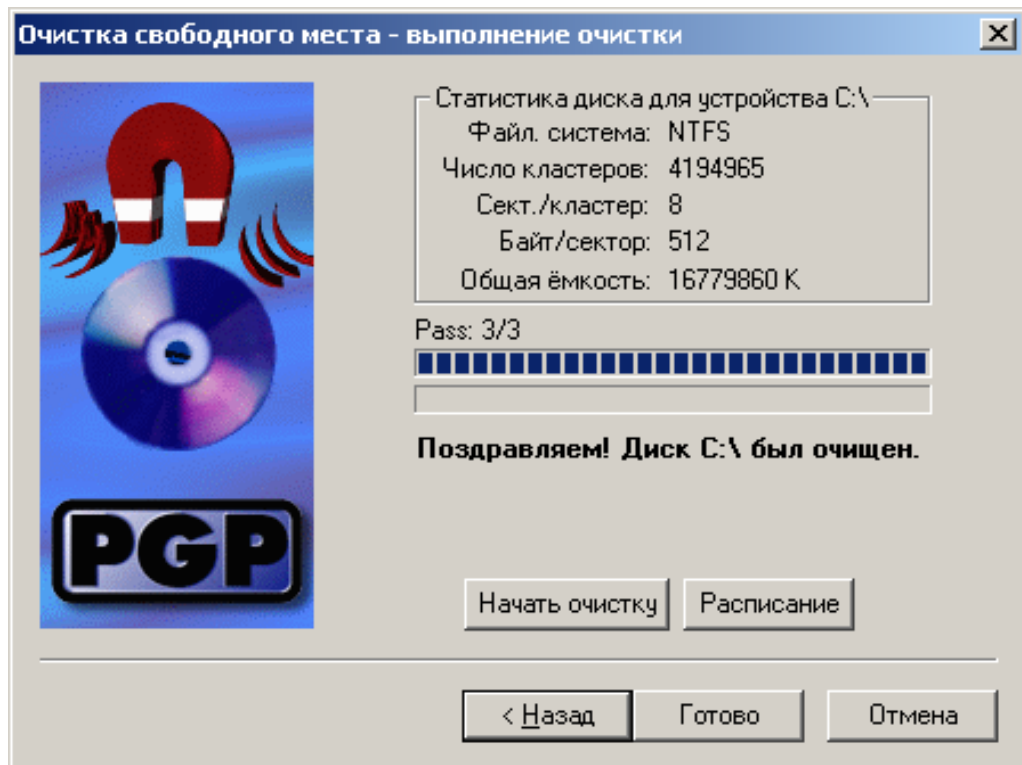


Рис. 2.25.

С помощью утилит PGP Wipe и Freespace Wipe и Планировщика Windows можно создать ряд заданий для регулярной очистки свободного пространства тех или иных дисков или уничтожения содержимого определённых папок.

Чтобы запланировать регулярную очистку свободного пространства диска с помощью PGP Freespace Wipe:

1. В окне PGPtools нажмите кнопку *Freespace Wipe* (🗑️).
2. В окне мастера очистки нажмите *Далее*.
3. В поле *Очистить диск* выберите диск, подлежащий регулярной очистке, введите число проходов очистки
4. Чтобы продолжить, нажмите *Далее*.
5. На странице *Очистка свободного места – выполнение очистки* нажмите кнопку *Расписание*.

6. Если хотите запланировать регулярную очистку свободного пространства выбранного диска с указанными настройками, то в ответ на предупреждение нажмите *ОК*.

Работая в Windows NT вам потребуется ввести свой логин и пароль. Сделайте это в появившемся окошке.

7. В открывшемся окне Планировщика заданий Windows (рисунок 2.26) укажите периодичность выполнения операции (ежедневно, еженедельно, при простое и т.д.) и, если нужно, время начала выполнения очистки.

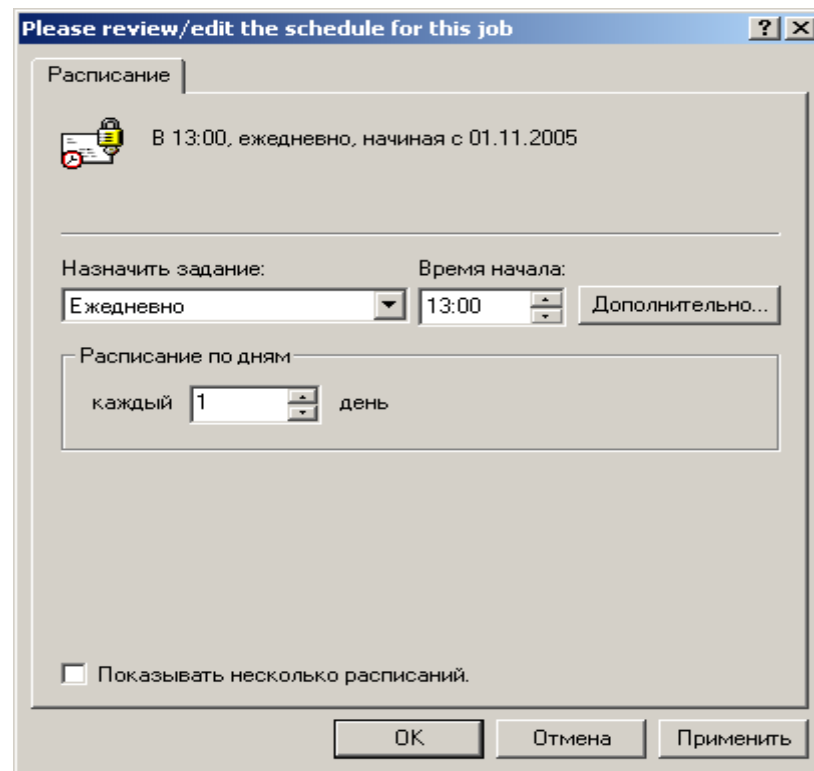


Рис. 2.26.

8. В разделе *Дополнительных* настроек можете указать некоторые расширенные параметры выполнения задания, в частности, дату начала, дату окончания выполнения задания, график повторов и пр.

9. Закончив планирование задания нажмите *ОК*.

Если в дальнейшем вам потребуется изменить порядок выполнения задания или отменить его, откройте Планировщик, находящийся в системном трее по соседству с PGPtray.

3. Задание на лабораторную работу

1. Изучите вкладки окна “Настройки PGP”.
2. Создайте с помощью менеджера PGPkeys ключи шифрования двух типов: RSA и DH/DSS.
3. Сохраните полученные ключи (открытые и секретные) в отдельный файл.
4. Назначьте ключ DH/DSS используемым по умолчанию.
5. Дезактивируйте ключевую пару RSA.
6. Активируйте ключевую пару RSA.
7. Аннулируйте ключевую пару RSA.
8. Изучите свойства ключевой пары DH/DSS:
 - 8.1.измените пароль;
 - 8.2.добавьте подключ;
 - 8.3.аннулируйте созданный подключ;
 - 8.4.добавьте фото-удостоверение.
9. Обменяйтесь с другим студентом открытыми ключами DH/DSS.
10. Импортируйте/вставьте полученный ключ на связку.
11. Установите подлинность полученного ключа с помощью его отпечатка.
12. Установите степень доверия к владельцу полученного ключа на максимальный уровень.
13. Зашифруйте произвольное сообщение/файл и обменяйтесь полученным результатом с другим студентом. Расшифруйте полученное сообщение.

14. Подпишите произвольное сообщение/файл и обменяйтесь полученным результатом с другим студентом. Проверьте достоверность источников полученного сообщения.
15. Подпишите и зашифруйте произвольное сообщение/файл и обменяйтесь полученным результатом с другим студентом. Расшифруйте полученное сообщение и проверьте достоверность его источников.
16. Уничтожьте все ненужные файлы, используя утилиту PGP Wipe.
17. Сделайте отчет по проделанной работе. Отчет должен содержать
 - 17.1. цель работы;
 - 17.2. описание выполненных действий по каждому пункту задания;
 - 17.3. открытые и секретные ключи, полученные в ходе выполнения работы;
 - 17.4. открытый ключ, полученный от другого студента;
 - 17.5. результаты шифрования/расшифрования, подписи/верификации сообщений/файлов;
 - 17.6. выводы по проделанной работе.

4. Контрольные вопросы

1. Что такое асимметричная криптографическая система? В чем ее преимущества перед симметричной системой? В чем недостатки?
2. Какие проблемы безопасности позволяет решить криптографическая защита информации? В чем они заключаются?
3. Что такое цифровая подпись? В чем ее отличие от рукописной подписи?
4. Какие задачи позволяет решить цифровая подпись?
5. Какие функции входят в состав PGP?
6. Какие типы ключей используются в PGP?
7. В чем различие между аннулированием и удалением ключа?

8. Что такое “отпечаток ключа” и для чего он используется?
9. Что такое “имплицитное доверие”?
10. Какую информацию может содержать сертификат открытого ключа PGP?
11. Каким образом в PGP производится надёжно удаление файлов с жёсткого диска?

4. Рекомендуемая литература

1. <http://www.pgpru.com/>
2. Романец Ю.В., Тимофеев П.А., Шаньгин В.Ф. Защита информации в компьютерных системах и сетях. – М.: Радио и связь, 2001. – 376 с.
3. Алферов А.П., Зубов А.Ю., Кузьмин А.С., Черемушкин А.В. Основы криптографии – М.: Гелиос АРВ, 2001. – 480 с., ил.
4. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. – М.: ТРИУМФ, 2002 – 816 с.