Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Кафедра информатики

В. А. Ганжа, В. В. Сидорик, О. И. Чичко

Компьютерные сети. Информационная безопасность и сохранение информации

Рекомендовано УМО по образованию в области информатики и радиоэлектроники в качестве учебно-методического пособия для специальности 1-40 04 01 «Информатика и технологии программирования»

Минск БГУИР 2014

УДК 004.7(076)
ББК 32.973.202я73
Г19

Рецензенты:

кафедра информационных технологий в образовании учреждения образования «Белорусский государственный педагогический университет имени Максима Танка» (протокол №8 от 23.05.2013 г.);

главный научный сотрудник государственного научного учреждения «Объединенный институт проблем информатики Национальной академии наук Беларуси», доктор технических наук, доцент С. Ф. Липницкий

Ганжа, В. А.

Г19 Компьютерные сети. Информационная безопасность и сохранение информации : учеб.-метод. пособие / В. А. Ганжа, В. В. Сидорик, О. И. Чичко. – Минск : БГУИР, 2014. – 128 с. : ил.

ISBN 978-985-543-031-6.

Рассматриваются проблемы информационной безопасности. Приводится ряд мероприятий по усилению информационной защиты. Разбираются некоторые аспекты использования пакета PGP, пакета стеганографии, вычисления хэш-функции для практической работы со студентами.

УДК 004.7(076)
ББК 32.973.202я73

ISBN 978-985-543-031-6

Содержание

Введение

При подготовке инженеров-программистов учебным планом предусмотрено изучение курса «Компьютерные сети». Этот предмет объединяет в себе много других не менее сложных предметов, кроме того, компьютерная сеть как информационная система включает в себя персонал, людей. В данной работе рассматриваются вопросы безопасности информации в компьютерных сетях, в том числе и роль фактора персонала. Для понимания и овладения этим предметом недостаточно только технических знаний, поскольку в проблемах безопасности информации и в различных её нарушениях в малой степени «повинна» аппаратура, но в значительно большей степени – люди, персонал, их взаимоотношения, их отношение к аппаратуре, оборудованию, к материальным и информационным ценностям.

Если в обработке информации участвует только аппаратура, как правило, никаких коллизий с ней не происходит, только в случае сбоя в аппаратуре некоторые биты могут потеряться или перепутаться. Для этого случая существуют различные методы контроля, например, дополнительные разряды для контроля чётности. Другой причиной для беспокойства могут являться шумы, попавшие в информационный поток. Однако с шумами также можно бороться, следуя каноническим предписаниям классика теории информации К. Шеннона:

- увеличить мощность передатчика;
- уменьшить скорость передачи информации;
- изменить метод модуляции;
- применить помехоустойчивое кодирование информации.

Перечисленные проблемы лежат в плоскости синтаксической и семантической адекватности информации.

В данном пособии будет рассматриваться несколько другой круг проблем – преобразование и обработка информации в информационных системах.

В классическом определении «информационная система» (ИС) рассматривается как комплекс двух компонентов: аппаратуры, осуществляющей обработку информации, и персонала. Компьютерные сети тоже подпадают под это определение, так как это огромная, рассредоточенная ИС – комплекс аналоговой и цифровой аппаратуры плюс многочисленный, многоязычный персонал. Вот почему в этом курсе наряду с техническими аспектами рассматриваются нормативно-правовые аспекты, законодательные, между-

народные стандарты, поскольку аппаратуру всегда обслуживает персонал, постольку всегда будут присутствовать риски «сбоя» персонала и «человеческий» фактор.

Проблемы из синтаксической адекватности информации переходят в плоскость прагматической адекватности, то есть в случае «сбоя» мы оцениваем ущерб не в байтах и битах, а пытаемся оценить его в денежном, материальном эквиваленте.

Под информационной безопасностью обычно понимают защищённость информации от случайных или преднамеренных воздействий естественного или искусственного характера, которые могут нанести ущерб субъектам информационных отношений: владельцам информации, пользователям информации и оборудованию. Защита информации – комплекс мероприятий, направленных на обеспечение информационной безопасности. Поэтому трактовка проблем, связанных с информационной безопасностью, для разных категорий субъектов может существенно различаться. Для примера достаточно сопоставить организации со строгим режимом работы и учебные институты. В первом случае «пусть лучше все сгорит и сломается, чем враг узнает хоть один секретный бит», во втором – «да нет у нас никаких секретов, лишь бы учебный процесс не останавливался».

Важность этой проблемы объясняется двумя основными причинами: ценностью накопленных информационных ресурсов и критической зависимостью от информационных технологий. Разрушение важной информации, кража конфиденциальных данных, перерыв в работе вследствие отказа – всё это выливается в крупные материальные потери, наносит ущерб репутации организации. Проблемы с системами управления или медицинскими системами угрожают здоровью и жизни людей. Современные информационные системы сложны и, значит, опасны уже сами по себе, даже без учёта активности злоумышленников. Постоянно обнаруживаются новые уязвимые места в программном обеспечении. Приходится принимать во внимание чрезвычайно широкий спектр аппаратного и программного обеспечения, многочисленные связи между компонентами.

Информационная безопасность не сводится исключительно к защите от несанкционированного доступа к информации, это принципиально более широкое понятие. Субъект информационных отношений может пострадать, нести убытки, получить моральный ущерб не только от несанкционированного доступа, но и от поломки системы, вызвавшей перерыв в работе. Более того, для мно-

гих организаций защита от несанкционированного доступа к информации стоит по важности отнюдь не на первом месте, например, в вузах, в электронных средствах массовой информации.

Правильный подход к проблемам информационной безопасности начинается с выявления субъектов информационных отношений и интересов этих субъектов, связанных с использованием ИС. Угрозы информационной безопасности – оборотная сторона использования информационных технологий.

Цель мероприятий в области информационной безопасности – защитить интересы субъектов информационных отношений.

Принято выделять основные составляющие [3] информационной безопасности (ИБ) – обеспечение **доступности**, **целостности** и **конфиденциальности** информационных ресурсов:

- **доступность** – возможность за приемлемое время получить требуемую информационную услугу;
- **целостность** – актуальность и непротиворечивость информации, её защищённость от разрушения и несанкционированного изменения;
- **конфиденциальность** – защита от несанкционированного доступа к информации.

Превалирующая роль доступности проявляется в системах управления производством, транспортом. Весьма неприятные последствия – и материальные, и моральные – может иметь длительная недоступность информационных услуг, которыми пользуется большое количество людей: продажа железнодорожных и авиабилетов, банковские услуги; web-сайты организаций, резервирование мест в гостиницах.

Целостность оказывается важнейшим аспектом ИБ в тех случаях, когда информация служит «руководством к действию». Рецептура лекарств, предписанные медицинские процедуры, набор и характеристики комплектующих изделий, ход технологического процесса – всё это примеры информации, нарушение целостности которой может оказаться в буквальном смысле смертельным. Средства контроля целостности применяются при анализе потока финансовых сообщений с целью выявления кражи, переупорядочения или дублирования отдельных сообщений. Неприятно и искажение официальной информации, если это текст закона или страница web-сервера государственной структуры.

Конфиденциальность – самый понятный аспект информационной безопасности. Практическая реализация мер по обеспечению

конфиденциальности современных информационных систем наталкивается на серьёзные трудности. Во-первых, сведения о технических каналах утечки информации являются закрытыми, так что большинство пользователей лишено возможности составить представление о потенциальных рисках. Во-вторых, на пути пользовательской криптографии как основного средства обеспечения конфиденциальности стоят многочисленные законодательные препятствия и технические проблемы.

Для всех субъектов информационных отношений, реально использующих ИС, на первом месте стоит доступность. Практически не уступает ей по важности целостность. Какой смысл в информационной услуге, если она содержит искажённые сведения? Наконец, конфиденциальные моменты есть также у многих организаций (даже в упоминавшихся выше учебных институтах стараются не разглашать сведения о зарплате сотрудников) и отдельных пользователей (например пароли).

Полную защиту информации осуществить невозможно, всегда останутся неучтённые бреши, непредвиденные факторы и обстоятельства. Можно лишь стремиться к такому идеалу. К тому же полная защита информационной системы в масштабах предприятия требует огромных капиталовложений, поэтому идут на компромиссы в зависимости от решаемых задач. Если стоимость информационных ценностей невелика, то создание дорогой системы защиты – обременительная роскошь. Если же речь идёт о защите важных информационных объектов, то и вложение средств здесь оправдано. Система защиты информации эффективна, если задействованы и участвуют все компоненты информационной безопасности:

– законодательный (стандарты, нормативно-правовые документы);
– административный;
– процедурный;
– программно-технический.

Проблема ИБ не только техническая; без законодательной базы, без постоянного внимания руководства организации и выделения необходимых ресурсов, без мер управления персоналом и физической защиты решить её невозможно. Комплексность также усложняет проблематику ИБ, требуется взаимодействие специалистов из разных областей.

Сегодня Интернет представляет собой эффективную, но вместе с тем и непредсказуемую среду, полную разнообразных угроз и

опасностей. Большая группа угроз связана с несовершенством протоколов, в частности протоколов стека TCP/IP [7, 8]. Эти протоколы разрабатывались в то время, когда проблема обеспечения информационной безопасности ещё не стояла на повестке дня. Сообщество пользователей Интернета представляло собой ограниченный круг заинтересованных в эффективной работе Сети специалистов, и уж, конечно, никто не посягал на её работоспособность. Создаваемые в такой «тепличной» атмосфере протоколы не содержали механизмов, позволяющих противостоять возможным (тогда только теоретически) атакам злоумышленников.

Особенно хочется коснуться проблем безопасности в компьютерных сетях и эволюции, которая произошла с концепцией защиты информации в сетях. Изначально, в конце 60-х гг., при зарождении и возникновении компьютерных сетей предполагалось, что к ним будет ограниченный доступ высококультурного и высокоинтеллектуального персонала, который в принципе не может и не будет замышлять ничего плохого и злонамеренного в отношении оборудования, программного обеспечения и обрабатываемой информации. Шли годы, компьютерные сети росли и развивались, к ним получили доступ и ими стали пользоваться огромные массы людей «с улицы», часто не очень обременённые соблюдением каких-либо запретов и правил. И к чему это привело?

Оказалось, что более беззащитную и уязвимую структуру, чем компьютерные сети, трудно придумать. Рассмотрим подробнее. Инженеры при разработке компьютерных сетей больше всего думали о быстрой и надёжной передаче информации, трафика и меньше всего были озабочены проблемами безопасности. Это привело к тому, что были разработаны прекрасные технологии доступа к разделяемой среде Ethernet и Token Ring. Но именно на канальном уровне модели OSI и начинаются основные проблемы безопасности.

Как работает технология Ethernet? В любом учебнике по компьютерным сетям, например [8], мы читаем: «Конечные узлы для обмена данными используют **единственную** разделяемую среду[1)], применяя метод случайного доступа». Что это означает? Это означает, что узел, передающий данные, формирует кадр и отправляет его в разделяемую среду **всем** узлам. Этот кадр получают **все** узлы

[1)] Под разделяемой средой обычно понимают общую среду, по которой передаются данные в сети и к которой подключены все узлы сегмента сети. Разделяемой средой может быть медный сетевой кабель (витая пара, коаксиал), это может быть общий оптоволоконный кабель, это может быть беспроводный радиоэфир.

и проверяют MAC-адрес. По правилам кадр остаётся лишь у узла с соответствующим MAC-адресом. Если MAC-адрес кадра не соответствует MAC-адресу узла, то по этим правилам компьютер должен отбросить такой кадр. Но потенциальный нарушитель не будет соблюдать эти правила. Злоумышленник всегда может настроить свой узел с соответствующим анализатором пакетов на прослушивание и сбор всех пакетов сегмента сети.

Похожая ситуация в плане безопасности на сетевом и транспортном уровне модели OSI, где и IP, и TCP-протоколы создают пакеты с заголовками, у которых обязаны на определённых местах расставить IP-адрес, номер порта, размер пакета. И эти «места» жёстко регламентированы стандартами, поскольку их несоблюдение и нарушение приведёт к тому, что ваши данные сеть не сможет правильно идентифицировать и доставить адресату. Здесь возникает вопрос: «Как компьютерная сеть вообще ещё работает? Шифровать мои данные! Немедленно шифровать все данные!».

Да, криптография играет важную роль в системе информационной безопасности, но в компьютерных сетях спасает слабо и ненадолго, и вот почему. Допустим, читатель зашифровал свои данные и посылает их своему корреспонденту на другой конец планеты в Антарктиду, используя компьтерную сеть. Компьютерная сеть, являясь сетью с коммутацией пакетов, начнёт расчленять зашифрованные цельные данные и на каждом уровне модели OSI добавлять свою служебную информацию. Сначала это будут IP-пакеты на сетевом уровне, добравшись до канального уровня, они будут уже называться кадрами, а технология Ethernet добавит туда свои служебные данные. И спустившись, наконец, до физического уровня данные обрастают значительной служебной информацией. Такой процесс называется «Инкапсуляция пакетов». Конечно, **сердцевина кадра зашифрована**, но вся информационная оболочка, возникшая в процессе инкапсуляции, открыта, не зашифрована и доступна любопытствующему пользователю и, конечно, взломщику.

Как видно из вышеприведенных фактов, компьютерная сеть очень ненадёжная вещь в плане информационной безопасности. Конечно, никто сейчас не рискнёт менять идеологию и переделывать эту архисложную, но функционирующую ИС. Вот почему мы вынуждены пользоваться различными «заплатками», которые лишь частично решают проблему безопасности компьютерных сетей. Вот почему ещё раз возникает вопрос: «Почему компьютерная сеть ещё до сих пор работает?».

В пособии приведён ряд необходимых действий и мероприятий, которые помогут читателю (лишь частично) оградить и сохранить свои данные, поскольку полную 100-процентную защиту выстроить невозможно. Выходя в сеть, пользуясь её услугами и сервисами, никто не застрахован от атаки и покушения на свои данные. Будьте бдительны!

1 Теоретическая часть

1.1 Законодательный уровень

Основным для обеспечения информационной безопасности является законодательный уровень. Большинство людей не совершают противоправных действий не потому, что это технически невозможно, а потому, что это осуждается обществом или наказывается законом, потому что так поступать не принято. На этом уровне выделяют две группы мер:

– создание и поддержание в обществе негативного отношения к нарушениям и нарушителям информационной безопасности;
– разрушение романтического ореола, существующего вокруг слова «хакер», и осознание того, что хакер – преступник, с которым вы также можете столкнуться;
– повышение общей культуры и образованности общества в области информационной безопасности, помогающее в разработке и распространении средств обеспечения информационной безопасности.

На законодательном уровне должен работать механизм, согласующий процесс разработки законов с реалиями и прогрессом информационных технологий. Законы не могут опережать жизнь, но разрыв между ними и практикой не должен быть значительным, иначе этот законодательный «вакуум» будет заполнен криминалом.

В Республике Беларусь существуют законодательные акты, регламентирующие вопросы информации, информатизации и защиты информации. Это прежде всего [16]:

– ЗАКОН РЕСПУБЛИКИ БЕЛАРУСЬ «Об информации, информатизации и защите информации» от 10 ноября 2008 г. №455-З;
– ПОСТАНОВЛЕНИЕ СОВЕТА МИНИСТРОВ РЕСПУБЛИКИ БЕЛАРУСЬ «О некоторых вопросах защиты информации» от 26 мая 2009 г. №675;
– УКАЗ ПРЕЗИДЕНТА РЕСПУБЛИКИ БЕЛАРУСЬ «О мерах по совершенствованию использования национального сегмента сети Интернет» от 1 февраля 2010 г. №60;
– ПОСТАНОВЛЕНИЕ СОВЕТА МИНИСТРОВ РЕСПУБЛИКИ БЕЛАРУСЬ «О некоторых вопросах совершенствования использования национального сегмента глобальной компьютерной сети Интернет» от 29 апреля 2010 г. №644;

– ПОСТАНОВЛЕНИЕ СОВЕТА МИНИСТРОВ РЕСПУБЛИКИ БЕЛАРУСЬ «О Стратегии развития информационного общества в Республике Беларусь на период до 2015 года и плане первоочередных мер по реализации Стратегии развития информационного общества в Республике Беларусь на 2010 год» от 9 августа 2010 г. №1174;
– УКАЗ ПРЕЗИДЕНТА РЕСПУБЛИКИ БЕЛАРУСЬ «О некоторых вопросах в сфере государственных секретов» от 25 февраля 2011 г. №68.

Эти законодательные акты можно найти по ссылке http://www.pravo.by/.

1.2 Нормативные документы и стандарты

Все государства в вопросах информационной безопасности должны понимать друг друга, говорить «на одном языке» стандартов и нормативных технических документов. Эти документы рассматривают проблемы информационной безопасности с очень близких позиций под несколько разным ракурсом. Это оценочные стандарты, классифицирующие информационные системы и средства защиты по требованиям безопасности, и технические спецификации, предписывающие различные пути реализации средств защиты.

1.2.1 Оценочный стандарт Министерства обороны США «Критерии оценки доверенных компьютерных систем»

Исторически первым оценочным стандартом, получившим широкое распространение во многих странах, стал стандарт Министерства обороны США «Критерии оценки доверенных компьютерных систем», называемый по цвету обложки «Оранжевой книгой», впервые опубликованный в августе 1983 г. Этот документ описывает не безопасные, а доверенные системы, то есть системы, которым можно оказать определенную степень доверия.

«Оранжевая книга» определяет понятие безопасной системы, которая «управляет с помощью соответствующих средств доступом к информации, так что только должным образом авторизованные лица или процессы, действующие от их имени, получают право читать, записывать, создавать и удалять информацию». Поскольку безопасных систем не существует, оценивается лишь степень доверия, которую можно оказать той или иной системе.

Доверенная система. В «Оранжевой книге» доверенная система определяется как «система, использующая достаточные аппарат-

ные и программные средства, чтобы обеспечить одновременную обработку информации разной степени секретности группой пользователей без нарушения прав доступа».

Степень доверия оценивается по **активному аспекту защиты** – политике безопасности и **пассивному аспекту защиты** – уровню гарантированности. Политика безопасности – правила и нормы поведения, определяющие, как организация обрабатывает и защищает информацию. Чем выше степень доверия системе, тем строже и многообразнее политика безопасности. Политика безопасности – активный аспект защиты, включающий в себя анализ возможных угроз и выбор мер противодействия.

Уровень гарантированности – мера доверия, которая может быть оказана архитектуре и реализации информационной системы. Доверие безопасности основано на анализе результатов тестирования и проверки общего замысла и реализации системы. Уровень гарантированности показывает, насколько корректны механизмы, отвечающие за реализацию политики безопасности. Это – пассивный аспект защиты.

Механизмы безопасности. Согласно «Оранжевой книге», политика безопасности должна обязательно включать в себя следующие элементы:

– произвольное управление доступом;
– безопасность повторного использования объектов;
– метки безопасности;
– принудительное управление доступом.

Классы безопасности. «Критерии» Министерства обороны США допускают ранжирование информационных систем по степени доверия безопасности. Определяется четыре уровня доверия – D, C, B и A. Уровень D предназначен для систем, признанных неудовлетворительными. По мере перехода от уровня C к A к системам предъявляются все более жёсткие требования. Уровни C и B подразделяются на классы (C1, C2, B1, B2, B3) с постепенным возрастанием степени доверия.
Всего имеется шесть классов безопасности – C1, C2, B1, B2, B3, A1. Чтобы в результате процедуры сертификации систему можно было отнести к некоторому классу, её политика безопасности и уровень гарантированности должны удовлетворять заданным требованиям.

В своё время публикация «Оранжевой книги» стала эпохальным событием в области безопасности информации. Появился обще-

признанный понятийный базис, без которого обсуждение проблем информационной безопасности было бы затруднительным. Конечно, на момент публикации (1983 г.) компьютерные сети были ещё в зачаточном состоянии и поэтому в «Оранжевой книге» вопросы безопасности распределённых систем не рассматриваются.

1.2.2 Информационная безопасность распределённых систем. Рекомендации Международного союза телекоммуникаций (ITU) X.800

Документ X.800 является технической спецификацией, предписывающей различные способы реализации информационной безопасности и защиты компьютерных сетей. Документ описывает следующие сетевые функции безопасности, сервисы безопасности:

– аутентификация – проверка подлинности партнёров по общению;
– управление доступом – обеспечение защиты от несанкционированного использования ресурсов, доступных по сети;
– конфиденциальность данных – обеспечение защиты от несанкционированного получения информации;
– целостность данных – защита данных от несанкционированного изменения;
– неотказуемость – подтверждение подлинности данных и невозможность отказа от совершённых действий.

Документ X.800 широко использует стандарт семиуровневой модели открытых систем OSI, на которых реализуются перечисленные выше сервисы безопасности. Поддержку всех сервисов можно реализовать на уровне приложений.

1.2.3 Стандарт ISO/IEC 15408 «Критерии оценки безопасности информационных технологий»

«Критерии оценки безопасности информационных технологий» (изданы в 1999 г.) является оценочным стандартом. Этот международный стандарт – итог многолетней работы специалистов нескольких стран, он охватывает документы национального и межнационального масштаба. По историческим причинам данный стандарт часто называют «Общими критериями».

«Общие критерии» (ОК) на самом деле являются метастандартом, определяющим инструменты оценки безопасности ИС и порядок их использования. В отличие от «Оранжевой книги» ОК не содержат предопределенных «классов безопасности». Такие классы можно строить, исходя из требований безопасности, существую-

щих для конкретной организации и/или конкретной информационной системы.

Как и «Оранжевая книга», «Общие критерии» содержат два основных вида требований безопасности:

– функциональные, соответствующие **активному аспекту защиты**, предъявляемые к функциям безопасности и реализующим их механизмам;
– требования доверия, соответствующие **пассивному аспекту защиты**, предъявляемые к технологии, процессу разработки и эксплуатации.

Требования безопасности предъявляются, а их выполнение проверяется для определенного объекта оценки – аппаратно-программного продукта или информационной системы. Безопасность в «Общих критериях» рассматривается в привязке к жизненному циклу объекта оценки. Выделяются следующие этапы:

– определение назначения, условий применения, целей и требований безопасности;
– проектирование и разработка;
– испытания, оценка и сертификация;
– внедрение и эксплуатация.

В ОК объект оценки рассматривается в контексте среды безопасности, которая характеризуется определенными условиями и угрозами (рисунок 1).

Собственник определяет множество **информационных ценностей**, которые должны быть защищены от различного рода атак. Атаки осуществляются противниками или оппонентами, использующими различные уязвимости в защищаемых ценностях. Основными нарушениями безопасности являются раскрытие информационных ценностей (потеря конфиденциальности), их неавторизованная модификация (потеря целостности) или неавторизованная потеря доступа к этим ценностям (потеря доступности).

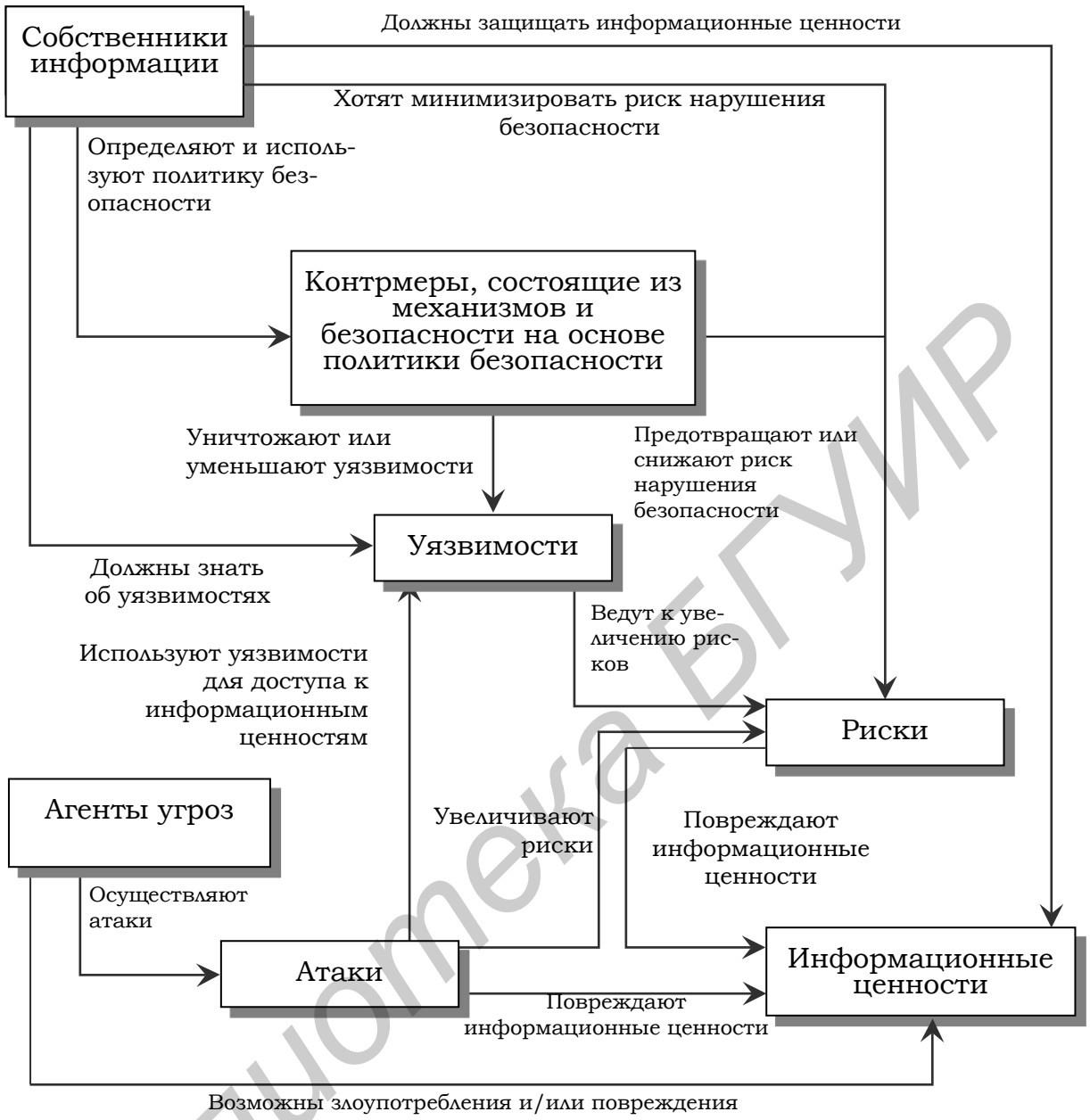


Рисунок 1 – Связи концепций безопасности
по стандарту ISO/IEC 15408

Собственники информационных ценностей анализируют уязвимости защищаемых ресурсов и возможные атаки, которые могут иметь место в конкретном окружении. В результате такого анализа определяются риски для данного набора информационных ценностей. Этот анализ определяет выбор контрмер, который задается политикой безопасности и обеспечивается с помощью механизмов и сервисов безопасности. Следует учитывать, что отдельные уязвимости могут сохраниться и после применения механизмов и сервисов безопасности. Политика безопасности определяет согласованную совокупность механизмов и сервисов безопас-

ности, адекватную защищаемым ценностям и окружению, в котором они используются.

В «Общих критериях» используются следующие определения.

Уязвимость – слабое место в системе, с использованием которого может быть осуществлена атака.

Риск – вероятность того, что конкретная атака будет осуществлена с использованием конкретной уязвимости. В конечном счете каждая организация должна принять решение о допустимом для нее уровне риска. Это решение должно найти отражение в политике безопасности, принятой в организации.

Политика безопасности – правила, директивы и практические навыки, которые определяют то, как информационные ценности обрабатываются, защищаются и распространяются в организации и между информационными системами; набор критериев для предоставления сервисов безопасности.

Атака – любое действие, нарушающее безопасность информационной системы. Более формально можно сказать, что атака – это действие или последовательность связанных между собой действий, использующих уязвимости данной информационной системы и приводящих к нарушению политики безопасности.

Механизм безопасности – программное и/или аппаратное средство, которое определяет и/или предотвращает атаку.

Сервис безопасности – сервис, который обеспечивает задаваемую политикой безопасность систем и/или передаваемых данных, либо определяет осуществление атаки. Сервис использует один или более механизмов безопасности.

1.3 Административный уровень

К административному уровню информационной безопасности относятся действия общего характера, предпринимаемые руководством организации. Главная цель мер административного уровня [3] – сформировать программу работ в области информационной безопасности и обеспечить её выполнение, выделяя необходимые ресурсы и контролируя состояние дел. Основой программы является политика безопасности, отражающая подход организации к защите своих информационных активов. Руководство каждой организации должно осознать необходимость поддержания режима безопасности и выделения на эти цели значительных ресурсов.

Политика безопасности строится на основе анализа рисков, которые признаются реальными для информационной системы орга-

низации. Когда риски проанализированы и стратегия защиты определена, составляется программа обеспечения информационной безопасности. Под эту программу выделяются ресурсы, назначаются ответственные, определяется порядок контроля выполнения программы.

1.3.1 Политика безопасности

Политика безопасности – совокупность документированных решений, принимаемых руководством организации и направленных на защиту информации. С практической точки зрения политика безопасности рассматривается на трёх уровнях. К **верхнему уровню** относятся решения, затрагивающие организацию в целом. Они носят общий характер и исходят от руководства организации. Примерный список подобных решений может включать в себя следующие элементы:

– решение сформировать комплексную программу обеспечения информационной безопасности, назначение ответственных за её осуществление;
– формулировка целей организации в области информационной безопасности, определение общих направлений в достижении этих целей;
– обеспечение базы для соблюдения законов и правил;
– формулировка административных решений по вопросам реализации программы безопасности.

Для политики верхнего уровня цели организации в области информационной безопасности формулируются в терминах целостности, доступности и конфиденциальности. Если организация отвечает за поддержание критически важных баз данных, на первом плане может стоять уменьшение числа потерь, повреждений или искажений данных. Для организации, занимающейся продажей компьютерной техники, вероятно, важна актуальность информации о предоставляемых услугах и ценах и ее доступность максимальному числу потенциальных покупателей. Руководство режимного предприятия в первую очередь заботится о защите от несанкционированного доступа, то есть о конфиденциальности.

На верхний уровень выносится управление защитными ресурсами и координация использования этих ресурсов, выделение специального персонала для защиты критически важных систем и взаимодействие с другими организациями, обеспечивающими или контролирующими режим безопасности. Политика верхнего уровня должна чётко очерчивать сферу своего влияния. Возможно, это

будут все компьютерные системы организации (или даже больше, если политика регламентирует некоторые аспекты использования сотрудниками своих домашних компьютеров). Возможна, однако, и такая ситуация, когда в сферу влияния включаются лишь наиболее важные системы.

В политике определяются обязанности должностных лиц по выработке программы безопасности и претворению её в жизнь. Политика верхнего уровня имеет дело с тремя аспектами законопослушности и исполнительской дисциплины. Во-первых, организация должна соблюдать существующие законы. Во-вторых, следует контролировать действия лиц, ответственных за выработку программы безопасности. Наконец, необходимо обеспечить определённую степень исполнительности персонала, а для этого нужно выработать систему поощрений и наказаний.

На верхний уровень следует выносить минимум вопросов. Это целесообразно в случае экономии средств или когда иначе поступить просто невозможно.

Британский стандарт BS 7799:1995 рекомендует включать в документ, характеризующий политику безопасности организации, следующие разделы:

- вводный, подтверждающий озабоченность высшего руководства проблемами информационной безопасности;
- организационный, содержащий описание подразделений, комиссий, групп, отвечающих за работы в области информационной безопасности;
- классификационный, описывающий имеющиеся в организации материальные и информационные ресурсы и необходимый уровень их защиты;
- штатный, характеризующий меры безопасности, применяемые к персоналу (описание должностей с точки зрения информационной безопасности, организация обучения и переподготовки персонала, порядок реагирования на нарушения режима безопасности);
- раздел, освещающий вопросы физической защиты;
- управляющий раздел, описывающий подход к управлению компьютерами и компьютерными сетями;
- раздел, описывающий правила разграничения доступа к производственной информации;
- раздел, характеризующий порядок разработки и сопровождения систем;

– раздел, описывающий меры, направленные на обеспечение непрерывной работы организации;
– юридический раздел, подтверждающий соответствие политики безопасности действующему законодательству.

К **среднему уровню** можно отнести вопросы, касающиеся отдельных аспектов информационной безопасности, но важные для различных эксплуатируемых организацией систем. Примеры таких вопросов – отношение к передовым (но, возможно, недостаточно проверенным) технологиям, доступ в Internet (как совместить свободу доступа к информации с защитой от внешних угроз?), использование домашних компьютеров, применение пользователями неофициального программного обеспечения.

Политика среднего уровня должна для каждого аспекта освещать следующие темы.

Описание аспекта. Например, если рассмотреть применение пользователями неофициального программного обеспечения, последнее можно определить как ПО, которое не было одобрено и/или закуплено на уровне организации.

Область применения. Следует определить, где, когда, как, по отношению к кому и чему применяется данная политика безопасности. Например, касается ли политика, связанная с использованием неофициального программного обеспечения, организаций-субподрядчиков? Затрагивает ли она сотрудников, пользующихся портативными и домашними компьютерами и вынужденных переносить информацию на компьютеры своих рабочих мест?

Позиция организации по данному аспекту. Продолжая пример с неофициальным программным обеспечением, можно представить себе позиции полного запрета, выработки процедуры приемки подобного ПО. Позиция может быть сформулирована и в гораздо более общем виде, как набор целей, которые преследует организация в данном аспекте. Вообще стиль документов, определяющих политику безопасности, в разных организациях может сильно отличаться.

Роли и обязанности. В «политический» документ необходимо включить информацию о должностных лицах, ответственных за реализацию политики безопасности. Например, если для использования неофициального программного обеспечения сотрудникам требуется разрешение руководства, должно быть известно, у кого и как его можно получить. Если неофициальное программное

обеспечение использовать нельзя, следует знать, кто следит за выполнением данного правила.

Законопослушность. Политика должна содержать общее описание запрещенных действий и наказаний за них.

Точки контакта. Должно быть известно, куда следует обращаться за разъяснениями, помощью и дополнительной информацией. Обычно «точкой контакта» служит определённое должностное лицо, а не конкретный человек, занимающий в данный момент данный пост.

Политика безопасности **нижнего уровня** относится к конкретным информационным сервисам. Она включает в себя два аспекта – цели и правила их достижения, поэтому её порой трудно отделить от вопросов реализации. В отличие от двух верхних уровней рассматриваемая политика должна быть определена более подробно. Есть много вещей, специфичных для отдельных видов услуг, которые нельзя единым образом регламентировать в рамках всей организации. В то же время эти вещи настолько важны для обеспечения режима безопасности, что относящиеся к ним решения должны приниматься на управленческом, а не техническом уровне. Приведем несколько примеров вопросов, на которые следует дать ответ в политике безопасности нижнего уровня:

1 Кто имеет право доступа к объектам, поддерживаемым сервисом?
2 При каких условиях можно читать и модифицировать данные?
3 Как организован удалённый доступ к сервису?

При формулировке целей политики нижнего уровня можно исходить из соображений целостности, доступности и конфиденциальности, но нельзя на этом останавливаться. Её цели должны быть более конкретными. Например, если речь идёт о системе расчёта заработной платы, можно поставить цель, чтобы только сотрудникам отдела кадров и бухгалтерии позволялось вводить и модифицировать информацию. В более общем случае цели должны связывать между собой объекты сервиса и действия с ними.

Из целей выводятся правила безопасности, описывающие, кто, что и при каких условиях может делать. Чем подробнее правила, чем более формально они изложены, тем проще поддержать их выполнение программно-техническими средствами. С другой стороны, слишком жёсткие правила могут мешать работе пользователей, вероятно, их придется часто пересматривать. Руководству пред-

стоит найти разумный компромисс, когда за приемлемую цену будет обеспечен приемлемый уровень безопасности, а сотрудники не окажутся чрезмерно связаны. Обычно наиболее формально задаются права доступа к объектам ввиду особой важности данного вопроса.

1.3.2 Программа безопасности

После того, как сформулирована политика безопасности, можно приступать к составлению программы её реализации и собственно к реализации.

Чтобы понять и реализовать какую-либо программу, её нужно структурировать по уровням, обычно в соответствии со структурой организации. В простейшем и самом распространенном случае достаточно двух уровней – верхнего, или центрального, который охватывает всю организацию, и нижнего, или служебного, который относится к отдельным услугам или группам однородных сервисов.

Программу верхнего уровня возглавляет лицо, отвечающее за информационную безопасность организации. Главные цели программы следующие:

- управление рисками (оценка рисков, выбор эффективных средств защиты);
- координация деятельности в области информационной безопасности, пополнение и распределение ресурсов;
- стратегическое планирование;
- контроль деятельности в области информационной безопасности.

В рамках программы верхнего уровня принимаются стратегические решения по обеспечению безопасности, оцениваются технологические новинки. Информационные технологии развиваются очень быстро, поэтому необходимо иметь чёткую политику контроля и внедрения новых средств.

Контроль деятельности в области безопасности имеет двустороннюю направленность. Во-первых, необходимо гарантировать, что действия организации не противоречат законам. При этом следует поддерживать контакты с внешними контролирующими организациями. Во-вторых, нужно постоянно отслеживать состояние безопасности внутри организации, реагировать на случаи нарушений и дорабатывать меры защиты с учётом изменения обстановки.

Следует подчеркнуть, что программа верхнего уровня должна занимать строго определённое место в деятельности организации, она должна официально приниматься и поддерживаться руководством, а также иметь определённый штат и бюджет.

Цель программы нижнего уровня – обеспечить надёжную и экономичную защиту конкретного сервиса или группы однородных сервисов. На этом уровне решается, какие следует использовать механизмы защиты; закупаются и устанавливаются технические средства; выполняется повседневное администрирование; отслеживается состояние слабых мест. Обычно за программу нижнего уровня отвечают администраторы сервисов.

1.3.3 Синхронизация программы безопасности с жизненным циклом систем

Если синхронизировать программу безопасности нижнего уровня с жизненным циклом защищаемого сервиса, можно добиться большего эффекта с меньшими затратами. Программисты знают, что добавить новую возможность к уже готовой системе на порядок сложнее, чем изначально спроектировать и реализовать ее. То же справедливо и для информационной безопасности.

В жизненном цикле информационного сервиса можно выделить следующие этапы.

Инициация. На данном этапе выявляется необходимость в приобретении нового сервиса, документируется его предполагаемое назначение.

Закупка. На данном этапе составляются спецификации, прорабатываются варианты приобретения, выполняется собственно закупка.

Установка. Сервис устанавливается, конфигурируется, тестируется и вводится в эксплуатацию.

Эксплуатация. На данном этапе сервис не только работает и администрируется, но и подвергается модификациям.

Выведение из эксплуатации. Происходит переход на новый сервис.

Рассмотрим действия, выполняемые на каждом из этапов, более подробно.

На этапе инициации оформляется понимание того, что необходимо приобрести новый или значительно модернизировать существующий сервис; определяется, какими характеристиками и ка-

кой функциональностью он должен обладать; оцениваются финансовые и иные ограничения.

С точки зрения безопасности важнейшим действием здесь является оценка критичности как самого сервиса, так и информации, которая с его помощью будет обрабатываться. Требуется сформулировать ответы на следующие вопросы:

1 Какого рода информация предназначается для обслуживания новым сервисом?
2 Каковы возможные последствия нарушения конфиденциальности, целостности и доступности этой информации?
3 Каковы угрозы, по отношению к которым сервис и информация будут наиболее уязвимы?
4 Есть ли какие-либо особенности нового сервиса (например, территориальная распределенность компонентов), требующие принятия специальных процедурных мер?
5 Каковы характеристики персонала, имеющие отношение к безопасности (квалификация, благонадежность)?
6 Каковы законодательные положения и внутренние правила, которым должен соответствовать новый сервис?

Результаты оценки критичности являются отправной точкой в составлении спецификаций. Кроме того, они определяют ту меру внимания, которую служба безопасности организации должна уделять новому сервису на последующих этапах его жизненного цикла.

Этап закупки – один из самых сложных. Нужно окончательно сформулировать требования к защитным средствам нового сервиса, к компании, которая может претендовать на роль поставщика, и к квалификации, которой должен обладать персонал, использующий или обслуживающий закупаемый продукт. Все эти сведения оформляются в виде спецификации, куда входят не только аппаратура и программы, но и документация, обслуживание, обучение персонала. Разумеется, особое внимание должно уделяться вопросам совместимости нового сервиса с существующей конфигурацией. Нередко средства безопасности являются необязательными компонентами коммерческих продуктов, и нужно проследить, чтобы соответствующие пункты не выпали из спецификации.

Когда продукт закуплен, его необходимо установить. Установка является очень ответственным делом. Во-первых, новый продукт следует сконфигурировать. Как правило, коммерческие продукты поставляются с отключенными средствами безопасности; их необходимо включить и должным образом настроить. Для большой ор-

ганизации, где много пользователей и данных, начальная настройка может стать весьма трудоёмким и ответственным делом.

Во-вторых, новый сервис нуждается в процедурных регуляторах. Следует позаботиться о чистоте и охране помещения, о документах, регламентирующих использование сервиса, о подготовке планов на случай экстренных ситуаций, об организации обучения пользователей. После принятия перечисленных мер необходимо провести тестирование. Его полнота и комплексность могут служить гарантией безопасности эксплуатации в штатном режиме.

Период эксплуатации – самый длительный и сложный. С психологической точки зрения наибольшую опасность в это время представляют незначительные изменения в конфигурации сервиса, в поведении пользователей и администраторов. Если безопасность не поддерживать, она ослабевает. Пользователи не столь ревностно выполняют должностные инструкции, администраторы менее тщательно анализируют регистрационную информацию. То один, то другой пользователь получает дополнительные привилегии. Кажется, что в сущности ничего не изменилось, на самом же деле от былой безопасности не осталось и следа.

Для борьбы с эффектом медленных изменений приходится прибегать к периодическим проверкам безопасности сервиса. Разумеется, после значительных модификаций подобные проверки являются обязательными.

При выведении из эксплуатации затрагиваются аппаратно-программные компоненты сервиса и обрабатываемые им данные. Аппаратура продается, утилизируется или выбрасывается. Только в специфических случаях необходимо заботиться о физическом разрушении аппаратных компонентов, хранящих конфиденциальную информацию. Программы, вероятно, просто стираются, если иное не предусмотрено лицензионным соглашением.

При выведении данных из эксплуатации их обычно переносят на другую систему, архивируют, выбрасывают или уничтожают. Если архивирование производится с намерением впоследствии прочитать данные в другом месте, следует позаботиться об аппаратно-программной совместимости средств чтения и записи. Информационные технологии развиваются очень быстро, и через несколько лет устройств, способных прочитать старый носитель, может просто не оказаться. Если данные архивируются в зашифрованном виде, необходимо сохранить ключ и средства расшифровки. При

архивировании и хранении архивной информации нельзя забывать о поддержании конфиденциальности данных.

1.4 Процедурный уровень. Основные классы мер процедурного уровня

На процедурном уровне рассматриваются меры безопасности, которые ориентированы на людей, а не на технические средства. Именно люди формируют режим информационной безопасности, и они же оказываются главной угрозой, поэтому «человеческий фактор» заслуживает особого внимания [9].

На процедурном уровне можно выделить следующие классы мер:

– управление персоналом;
– физическая защита;
– поддержание работоспособности;
– реагирование на нарушения режима безопасности;
– планирование восстановительных работ.

1.4.1 Управление персоналом

Управление персоналом начинается с приёма нового сотрудника на работу и даже раньше – с составления должностной инструкции. Уже на данном этапе желательно подключить к работе специалиста по информационной безопасности для определения компьютерных привилегий, ассоциируемых с должностью. Существует два общих принципа, которые следует иметь в виду:

– разделение обязанностей;
– минимизация привилегий.

Принцип разделения обязанностей предписывает так распределять роли и ответственность, чтобы один человек не мог нарушить критически важный для организации процесс. Например, нежелательна ситуация, когда крупные платежи от имени организации выполняет один человек. Надежнее поручить одному сотруднику оформление заявок на подобные платежи, а другому – заверять эти заявки. Другой пример – процедурные ограничения действий суперпользователя. Можно искусственно «расщепить» пароль суперпользователя, сообщив первую его часть одному сотруднику, а вторую – другому. Тогда критически важные действия по администрированию ИС они смогут выполнить только вдвоём, что снижает вероятность ошибок и злоупотреблений.

Принцип минимизации привилегий предписывает выделять пользователям только те права доступа, которые необходимы им для выполнения служебных обязанностей. Назначение этого принципа

очевидно – уменьшить ущерб от случайных или умышленных некорректных действий.

Предварительное составление описания должности позволяет оценить её критичность и спланировать процедуру проверки и отбора кандидатов. Чем ответственнее должность, тем тщательнее нужно проверять кандидатов: навести о них справки, быть может, побеседовать с бывшими сослуживцами. Подобная процедура может быть длительной и дорогой, поэтому нет смысла дополнительно усложнять её. В то же время неразумно и совсем отказываться от предварительной проверки.

Когда кандидат определён, он, вероятно, должен пройти обучение; по крайней мере его следует подробно ознакомить со служебными обязанностями, а также с нормами и процедурами информационной безопасности. Желательно, чтобы меры безопасности были им усвоены до вступления в должность и до создания его учётной записи в системе с входным именем, паролем и привилегиями.

С момента заведения учётной записи начинается его администрирование, а также протоколирование и анализ действий пользователя. Постепенно изменяется окружение, в котором работает пользователь, его служебные обязанности. Всё это требует соответствующего изменения привилегий. Техническую сложность представляют временные перемещения пользователя, выполнение им обязанностей взамен сотрудника, ушедшего в отпуск, и иные обстоятельства, когда полномочия нужно сначала предоставить, а через некоторое время взять обратно. В такие периоды профиль активности пользователя резко меняется, что создает трудности при выявлении подозрительных ситуаций. Определенную аккуратность следует соблюдать и при выдаче новых постоянных полномочий, не забывая ликвидировать старые права доступа.

Ликвидация учётной записи пользователя, особенно в случае конфликта между сотрудником и организацией, должна производиться максимально оперативно. Возможно и физическое ограничение доступа к рабочему месту. Разумеется, если сотрудник увольняется, у него нужно принять всё его компьютерное «хозяйство» и, в частности, криптографические ключи, если использовались средства шифрования.

К управлению сотрудниками примыкает администрирование лиц, работающих по контракту. В соответствии с принципом минимизации привилегий им нужно выделить ровно столько прав, сколько необходимо, и изъять эти права сразу по окончании контракта. Проблема, однако, состоит в том, что на начальном этапе внедре-

ния «внешние» сотрудники будут администрировать «местных», а не наоборот. Здесь на первый план выходит квалификация персонала организации, его способность быстро обучаться, а также оперативное проведение учебных курсов. Важны и принципы выбора деловых партнёров.

Иногда внешние организации принимают на обслуживание и администрирование ответственные компоненты компьютерной системы, например, сетевое оборудование. Нередко администрирование выполняется в удалённом режиме. Вообще говоря, это создаёт в системе дополнительные уязвимые места, которые необходимо компенсировать усиленным контролем средств удалённого доступа или обучением собственных сотрудников.

Проблема обучения – одна из основных с точки зрения информационной безопасности. Если сотрудник не знаком с политикой безопасности своей организации, он не может стремиться к достижению сформулированных в ней целей. Не зная мер безопасности, он не сможет их соблюдать. Напротив, если сотрудник знает, что его действия протоколируются, он, возможно, воздержится от нарушений.

1.4.2 Физическая защита

Безопасность информационной системы зависит от окружения, в котором она функционирует. Необходимо принять меры для защиты зданий и прилегающей территории, поддерживающей инфраструктуры, вычислительной техники, носителей данных. Основной принцип физической защиты, соблюдение которого следует постоянно контролировать, формулируется как «непрерывность защиты в пространстве и времени».

Мы кратко рассмотрим следующие направления физической защиты:

– физическое управление доступом;
– противопожарные меры;
– защита поддерживающей инфраструктуры;
– защита от перехвата данных;
– защита мобильных систем.

Меры физического управления доступом позволяют контролировать и при необходимости ограничивать вход и выход сотрудников и посетителей. Контролироваться может всё здание организации, а также отдельные помещения. Нужно сделать так, чтобы посетители по возможности не имели непосредственного доступа к компьютерам или в крайнем случае позаботиться о том, чтобы от

окон и дверей не просматривались экраны мониторов и принтеры. Необходимо, чтобы посетителей по внешнему виду можно было отличить от сотрудников. Если отличие состоит в том, что посетителям выдаются идентификационные карточки, а сотрудники ходят «без опознавательных знаков», злоумышленнику достаточно снять карточку, чтобы его считали «своим». Очевидно, соответствующие карточки нужно выдавать всем.

Средства физического управления доступом известны давно. Это охрана, двери с замками, перегородки, телекамеры, датчики движения и многое другое. Для выбора оптимального средства целесообразно провести анализ рисков.

Необходимо установка противопожарной сигнализации и автоматических средств пожаротушения. Обратим также внимание на то, что защитные меры могут создавать новые слабые места. Если на работу взят новый охранник, это, вероятно, улучшает физическое управление доступом. Если же он по ночам нарушает рабочий режим, то ввиду повышенной пожароопасности подобная мера защиты может только навредить.

Отдельную проблему составляют аварии водопровода. Они происходят нечасто, но могут нанести огромный ущерб. При размещении компьютеров необходимо принять во внимание расположение водопроводных и канализационных труб и постараться держаться от них подальше. Сотрудники должны знать, куда следует обращаться при обнаружении протечек.

Перехват данных может осуществляться самыми разными способами. Злоумышленник может подсматривать за экраном монитора, читать пакеты, передаваемые по сети, производить анализ побочных электромагнитных излучений и наводок. Необходимо повсеместно использовать криптографию, стараться максимально расширить контролируемую территорию, разместившись поодаль, пытаться держать под контролем линии связи (например, заключать их в надувную оболочку с обнаружением прокалывания), но самое разумное, вероятно, – постараться осознать, что для коммерческих систем обеспечение конфиденциальности является все-таки не главной задачей.

Мобильные и портативные компьютеры – заманчивый объект кражи. Их часто оставляют без присмотра, в автомобиле или на работе, и похитить такой компьютер совсем несложно. Необходимо шифровать данные на жёстких дисках таких компьютеров.

Вообще говоря, при выборе средств физической защиты следует производить анализ рисков. Так, принимая решение о закупке источника бесперебойного питания, необходимо учесть качество электропитания в здании, занимаемом организацией, характер и длительность сбоев электропитания, стоимость доступных источников и возможные потери от аварий. В то же время во многих случаях решения очевидны. Меры противопожарной безопасности обязательны для всех организаций. Стоимость реализации многих мер (например, установка обычного замка на дверь серверной комнаты) либо мала, либо хоть и заметна, но всё же явно меньше, чем возможный ущерб. В частности, имеет смысл регулярно копировать большие базы данных.

1.4.3 Поддержание работоспособности

Существует ряд рутинных мероприятий, направленных на поддержание работоспособности информационных систем. Именно здесь таится наибольшая опасность. Нечаянные ошибки системных администраторов и пользователей грозят повреждением аппаратуры, разрушением программ и данных; в лучшем случае они создают бреши в защите, которые делают возможной реализацию угроз.

Недооценка факторов безопасности в повседневной работе – ахиллесова пята многих организаций. Дорогие средства безопасности теряют смысл, если они плохо документированы, конфликтуют с другим программным обеспечением, а пароль системного администратора не менялся с момента установки.

Можно выделить следующие направления повседневной деятельности:

– поддержка пользователей;
– поддержка программного обеспечения;
– конфигурационное управление;
– резервное копирование;
– управление носителями;
– документирование;
– регламентные работы.

Поддержка пользователей подразумевает прежде всего консультирование и оказание помощи при решении разного рода проблем. Иногда в организациях создают для этой цели специальный «справочный стол», но чаще от пользователей «отбивается» системный администратор. Очень важно в потоке вопросов уметь выявлять проблемы, связанные с информационной безопасностью. Так,

многие трудности пользователей, работающих на персональных компьютерах, могут быть следствием заражения вирусами. Целесообразно фиксировать вопросы пользователей, чтобы выявлять их типичные ошибки и выпускать памятки с рекомендациями для распространенных ситуаций.

Поддержка программного обеспечения – одно из важнейших средств обеспечения целостности информации. Прежде всего необходимо следить за тем, какое программное обеспечение установлено на компьютерах. Если пользователи будут устанавливать программы по своему усмотрению, это может привести к заражению вирусами, а также появлению утилит, действующих в обход защитных средств. Вполне вероятно также, что «самодеятельность» пользователей постепенно приведет к хаосу на их компьютерах, а исправлять ситуацию придётся системному администратору.

Второй аспект поддержки программного обеспечения – контроль за отсутствием неавторизованного изменения программ и прав доступа к ним. Сюда же можно отнести поддержку эталонных копий программных систем. Обычно контроль достигается комбинированием средств физического и логического управления доступом, а также использованием утилит проверки и обеспечения целостности.

Резервное копирование необходимо для восстановления программ и данных после аварий. И здесь целесообразно автоматизировать работу, как минимум сформировав компьютерное расписание создания полных и инкрементальных копий, а как максимум – воспользовавшись соответствующими программными продуктами. Нужно также наладить размещение копий в безопасном месте, защищенном от несанкционированного доступа, пожаров, протечек, то есть от всего, что может привести к краже или повреждению носителей. Целесообразно иметь несколько экземпляров резервных копий и часть из них хранить вне территории организации, защищаясь таким образом от крупных аварий и аналогичных инцидентов.

Документирование – неотъемлемая часть информационной безопасности. В виде документов оформляется почти всё – от политики безопасности до журнала учёта носителей. К хранению одних документов применимы требования обеспечения конфиденциальности, других, таких как план восстановления после аварий, – требования целостности и доступности.

Регламентные работы – очень серьёзная угроза безопасности. Сотрудник, осуществляющий регламентные работы, получает исключительный доступ к системе, и на практике очень трудно контролировать, какие именно действия он совершает. Здесь на первый план выходит степень доверия к тем, кто выполняет работу.

1.4.4 Реагирование на нарушения режима безопасности

Программа безопасности, принятая организацией, должна предусматривать набор оперативных мероприятий, направленных на обнаружение и нейтрализацию нарушений режима информационной безопасности. Важно, чтобы в подобных случаях последовательность действий была спланирована заранее, поскольку меры нужно принимать срочные и скоординированные.

Реакция на нарушения режима безопасности преследует три главные цели:

– локализация инцидента и уменьшение наносимого вреда;
– выявление нарушителя;
– предупреждение повторных нарушений.

В организации должен быть человек, доступный 24 часа в сутки, который отвечает за реакцию на нарушения. Все должны знать координаты этого человека и обращаться к нему при первых признаках опасности. В общем, как при пожаре, нужно знать, куда звонить и что делать до приезда пожарной команды.

Нередко требование локализации инцидента и уменьшения наносимого вреда вступает в конфликт с желанием выявить нарушителя. В политике безопасности организации приоритеты должны быть расставлены заранее. Поскольку, как показывает практика, выявить злоумышленника очень сложно, на наш взгляд, в первую очередь следует заботиться о минимизации ущерба.

Чтобы найти нарушителя, нужно заранее выяснить контактные координаты поставщика сетевых услуг и договориться с ним о самой возможности и порядке выполнения соответствующих действий. Чтобы предотвратить повторные нарушения, необходимо анализировать каждый инцидент, выявлять причины, накапливать статистику. Каковы источники вредоносного ПО? Какие пользователи имеют обыкновение выбирать слабые пароли? На подобные вопросы и должны дать ответ результаты анализа.

1.4.5 Планирование восстановительных работ

Ни одна организация не застрахована от серьёзных аварий, вызванных естественными причинами, действиями злоумышленни-

ка, халатностью или некомпетентностью. В то же время у каждой организации есть функции, которые руководство считает критически важными, они должны выполняться несмотря ни на что. Планирование восстановительных работ позволяет подготовиться к авариям, уменьшить ущерб от них и сохранить способность к функционированию хотя бы в минимальном объёме.

Отметим, что меры информационной безопасности можно разделить на три группы, в зависимости от того, направлены ли они на предупреждение, обнаружение или ликвидацию последствий атак. Большинство мер носит предупредительный характер. Оперативный анализ регистрационной информации и некоторые аспекты реагирования на нарушения (так называемый активный аудит) служат для обнаружения и отражения атак. Планирование восстановительных работ, очевидно, можно отнести к последней из трех перечисленных групп.

Процесс планирования восстановительных работ можно разделить на следующие этапы:

– выявление критически важных функций организации, установление приоритетов;
– идентификация ресурсов, необходимых для выполнения критически важных функций;
– определение перечня возможных аварий;
– разработка стратегии восстановительных работ;
– подготовка к реализации выбранной стратегии;
– проверка стратегии.

Планируя восстановительные работы, следует отдавать себе отчёт в том, что полностью сохранить функционирование организации не всегда возможно. Необходимо выявить критически важные функции, без которых организация теряет своё лицо, и даже среди критичных функций расставить приоритеты, чтобы как можно быстрее и с минимальными затратами возобновить работу после аварии.

Идентифицируя ресурсы, необходимые для выполнения критически важных функций, следует помнить, что многие из них имеют некомпьютерный характер. На данном этапе желательно подключать к работе специалистов разного профиля, способных в совокупности охватить все аспекты проблемы. Критичные ресурсы обычно относятся к одной из следующих категорий:

– персонал;
– информационная инфраструктура;

– физическая инфраструктура.

Составляя списки ответственных специалистов, следует учитывать, что некоторые из них могут непосредственно пострадать от аварии (например от пожара), кто-то может находиться в состоянии стресса, часть сотрудников может лишиться возможности попасть на работу (например в случае массовых беспорядков). Желательно иметь некоторый резерв специалистов или заранее определить каналы, по которым можно на время привлечь дополнительный персонал.

При определении перечня возможных аварий нужно попытаться разработать их сценарии. Как будут развиваться события? Каковы могут оказаться масштабы бедствия? Что произойдёт с критичными ресурсами? Например, смогут ли сотрудники попасть на работу? Будут ли выведены из строя компьютеры? Возможны ли случаи саботажа? Будет ли работать связь? Пострадает ли здание организации? Можно ли будет найти и прочитать необходимые бумаги?

Стратегия восстановительных работ должна базироваться на наличных ресурсах и быть не слишком накладной для организации. При разработке стратегии целесообразно провести анализ рисков, которым подвергаются критичные функции, и попытаться выбрать наиболее экономичное решение. Стратегия должна предусматривать не только работу по временной схеме, но и возвращение к нормальному функционированию.

Подготовка к реализации выбранной стратегии состоит в выработке плана действий в экстренных ситуациях и по их окончании, а также в обеспечении некоторой избыточности критичных ресурсов. Последнее возможно и без большого расхода средств, если заключить с одной или несколькими организациями соглашения о взаимной поддержке в случае аварий – те, кто не пострадал, предоставляют часть своих ресурсов во временное пользование менее удачливым партнерам.

Избыточность обеспечивается также мерами резервного копирования, хранением копий в нескольких местах, представлением информации в разных видах (на бумаге и в файлах). Имеет смысл заключить соглашение с поставщиками информационных услуг о первоочередном обслуживании в критических ситуациях.

1.5 Программно-технический уровень

1.5.1 Основные понятия программно-технического уровня информационной безопасности

Программно-технические меры, то есть меры, направленные на контроль оборудования, программ и данных, образуют последний и один из самых важных рубежей информационной безопасности. Как показывает статистика, ущерб наносят в основном действия легальных пользователей, по отношению к которым процедурные регуляторы малоэффективны. Главные «враги» – некомпетентность и неаккуратность при выполнении служебных обязанностей, и только программно-технические меры способны им противостоять.

С одной стороны, быстрое развитие информационных технологий предоставляет «обороняющимся» новые возможности, но, с другой стороны, затрудняет обеспечение надёжной защиты, если опираться исключительно на меры программно-технического уровня. Причин тому несколько:

– повышение быстродействия микросхем, развитие архитектур с высокой степенью параллелизма позволяет взламывать ранее казавшиеся неприступными барьеры;
– развитие сетей и сетевых технологий, увеличение числа связей между информационными системами, рост пропускной способности каналов расширяют круг злоумышленников, имеющих техническую возможность организовывать атаки;
– появление новых информационных сервисов ведёт и к образованию новых уязвимых мест;
– конкуренция среди производителей программного обеспечения заставляет сокращать сроки разработки, что приводит к снижению качества тестирования и выпуску продуктов с дефектами защиты;
– навязываемая потребителям парадигма постоянного наращивания мощности аппаратного и программного обеспечения не позволяет долго оставаться в рамках надёжных, апробированных конфигураций.

Для программно-технического уровня основным является понятие сервиса безопасности, в который входят:

– идентификация и аутентификация;
– управление доступом;
– протоколирование и аудит;
– шифрование;

– контроль целостности;
– экранирование;
– анализ защищенности;
– обеспечение отказоустойчивости;
– обеспечение безопасного восстановления;
– туннелирование;
– управление.

1.5.2 Особенности современных информационных систем, существенные с точки зрения безопасности

Информационная система современной организации является весьма сложным образованием, которое пользуется многочисленными внешними сервисами и в свою очередь предоставляет собственные сервисы вовне.

Следует учитывать ещё по крайней мере два момента. Во-первых, для каждого сервиса основные грани ИБ (доступность, целостность, конфиденциальность) трактуются по-своему. Целостность с точки зрения системы управления базами данных и с точки зрения почтового сервера – вещи принципиально разные. Бессмысленно говорить о безопасности локальной или иной сети вообще, если сеть включает в себя разнородные компоненты. Следует анализировать защищенность сервисов, функционирующих в сети. Для разных сервисов и защиту строят по-разному.

Во-вторых, основная угроза информационной безопасности организаций по-прежнему исходит не от внешних злоумышленников, а от собственных сотрудников.

В силу изложенных причин далее будут рассматриваться распределённые, разнородные, многосервисные, эволюционирующие системы. Соответственно, нас будут интересовать решения, ориентированные на подобные конфигурации.

1.5.3 Архитектурная безопасность

Сервисы безопасности, какими бы мощными они ни были, сами по себе не могут гарантировать надежность программно-технического уровня защиты. Только проверенная архитектура способна сделать эффективным объединение сервисов, обеспечить управляемость информационной системы, ее способность развиваться и противостоять новым угрозам при сохранении таких свойств, как высокая производительность, простота и удобство использования.

С практической точки зрения наиболее важными являются следующие принципы архитектурной безопасности:

– непрерывность защиты в пространстве и времени, невозможность миновать защитные средства;
– следование признанным стандартам, использование апробированных решений;
– иерархическая организация ИС с небольшим числом сущностей на каждом уровне;
– усиление самого слабого звена;
– невозможность перехода в небезопасное состояние;
– минимизация привилегий;
– разделение обязанностей;
– эшелонированность обороны;
– разнообразие защитных средств;
– простота и управляемость информационной системы.

Кратко смысл перечисленных принципов заключается в следующем.

Если **непрерывность** нарушена, тогда у злоумышленника или недовольного пользователя появится возможность миновать защитные средства, он, разумеется, так и сделает.

Следование **признанным стандартам** и использование апробированных решений повышает надежность ИС и уменьшает вероятность попадания в тупиковую ситуацию, когда обеспечение безопасности потребует непомерно больших затрат и принципиальных модификаций.

Иерархическая организация ИС с небольшим числом сущностей на каждом уровне необходима по технологическим соображениям. При нарушении данного принципа система станет неуправляемой и, следовательно, обеспечить ее безопасность будет невозможно.

Надёжность любой обороны определяется самым **слабым звеном**. Злоумышленник не будет бороться против силы, он предпочтёт легкую победу над слабостью. Часто самым слабым звеном оказывается не компьютер или программа, а человек, и тогда проблема обеспечения информационной безопасности приобретает нетехнический характер.

Принцип невозможности **перехода в небезопасное состояние** означает, что при любых обстоятельствах, в том числе нештатных, защитное средство либо полностью выполняет свои функции, либо полностью блокирует доступ.

Применительно к программно-техническому уровню принцип **минимизации привилегий** предписывает выделять пользователям и администраторам только те права доступа, которые необходимы им для выполнения служебных обязанностей. Этот принцип позволяет уменьшить ущерб от случайных или умышленных некорректных действий пользователей и администраторов.

Принцип **разделения обязанностей** предполагает такое распределение ролей и ответственности, чтобы один человек не мог нарушить критически важный для организации процесс или создать брешь в защите по заказу злоумышленников. В частности, соблюдение данного принципа особенно важно, чтобы предотвратить злонамеренные или неквалифицированные действия системного администратора.

Принцип **эшелонированности обороны** предписывает не полагаться на один защитный рубеж, каким бы надёжным он ни казался. За средствами физической защиты должны следовать программно-технические средства, за идентификацией и аутентификацией – управление доступом и как последний рубеж – протоколирование и аудит. Эшелонированная оборона способна по крайней мере задержать злоумышленника, а благодаря наличию такого рубежа, как протоколирование и аудит, его действия не останутся незамеченными.

Принцип **разнообразия защитных** средств предполагает создание различных по своему характеру оборонительных рубежей, чтобы от потенциального злоумышленника требовалось овладение разнообразными и по возможности несовместимыми между собой навыками.

Очень важен **принцип простоты и управляемости** информационной системы в целом и защитных средств в особенности. Только для простого защитного средства можно формально или неформально доказать его корректность. Только в простой и управляемой системе можно проверить согласованность конфигурации различных компонентов и осуществлять централизованное администрирование.

1.6 Основные понятия криптографии

1.6.1 Основные понятия

Обсудим принципы, на которых основаны современные методы традиционного шифрования. Рассмотрим общую схему симметричной, или традиционной, криптографии (рисунок 2).

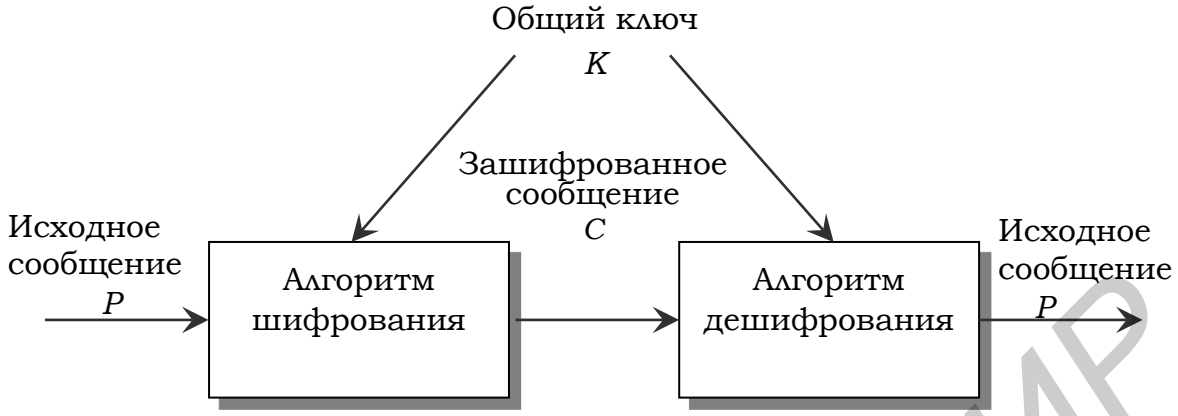


Рисунок 2 – Общая схема симметричного шифрования

В процессе шифрования используется определённый алгоритм шифрования, по которому обрабатываются входные незашифрованные данные [1]. Математический алгоритм обычно является открытым и описание его доступно в открытой печати, например, в математических журналах. Однако в процессе шифрования используется ключ, который является закрытым, им владеют только отправитель и получатель. Результатом работы алгоритма является зашифрованное сообщение. Ключ является значением, не зависящим от шифруемого сообщения. Изменение ключа должно приводить к изменению зашифрованного сообщения.

Зашифрованное сообщение передается получателю. Получатель преобразует зашифрованное сообщение в исходное незашифрованное сообщение с помощью алгоритма дешифрования и того же самого ключа, который использовался при шифровании, или ключа, легко получаемого из ключа шифрования.

Незашифрованное сообщение будем обозначать *P* или *M*, от слов *plaintext* и *message*. Зашифрованное сообщение будем обозначать *C*, от слова *ciphertext*.

Безопасность, обеспечиваемая традиционной криптографией, зависит от нескольких факторов. Во-первых, криптографический алгоритм должен быть достаточно сильным, чтобы передаваемое зашифрованное сообщение невозможно было расшифровать без ключа, используя только различные статистические закономерности зашифрованного сообщения или какие-либо другие способы его анализа.

Во-вторых, безопасность передаваемого сообщения должна зависеть от секретности ключа, но не от секретности алгоритма. Алго-

ритм должен быть проанализирован специалистами, чтобы исключить наличие слабых мест, при которых плохо скрыта взаимосвязь между незашифрованным и зашифрованным сообщениями. К тому же при выполнении этого условия производители могут создавать дешевые аппаратные чипы и свободно распространяемые программы, реализующие данный алгоритм шифрования.

В-третьих, алгоритм должен быть таким, чтобы нельзя было узнать ключ, даже зная достаточно много пар (зашифрованное сообщение, незашифрованное сообщение), полученных при шифровании с использованием данного ключа.

Описание стойкости алгоритма шифрования при статистическом анализе потока зашифрованных сообщений использует понятия диффузии и конфузии, введенные Клодом Шенноном.

Суть диффузии заключается в рассеянии статистических особенностей незашифрованного текста в широком диапазоне статистических характеристик зашифрованного текста. Это достигается тем, что значение каждого элемента незашифрованного текста влияет на значения многих элементов зашифрованного текста или, что то же самое, любой элемент зашифрованного текста зависит от многих элементов незашифрованного текста.

Конфузия усложняет статистическую взаимосвязь между зашифрованным текстом и ключом с целью противостояния попыткам определить ключ. Это достигается использованием сложных подстановочных алгоритмов – простые линейные подстановочные функции увеличивают беспорядок лишь в незначительной степени.

Если X – исходное сообщение и K – криптографический ключ, то зашифрованный передаваемый текст Y можно записать в виде

$$Y = EK(X), \quad (1)$$

где E – функция, осуществляющая алгоритм шифрования.

Получатель, используя тот же ключ, расшифровывает сообщение

$$X = DK(Y), \quad (2)$$

где D – функция, осуществляющая алгоритм дешифрования.

Противник, не имея доступа к K и X, пытается узнать X, K.

Алгоритмы симметричного шифрования различаются способом, которым обрабатывается исходный текст. Возможно шифрование блоками или шифрование потоком. Поточными называются шиф-

ры, в которых поток цифровых данных шифруется последовательно бит за битом. Блочными называются шифры, в которых логической единицей шифрования является некоторый блок открытого текста, после преобразований которого получается блок шифрованного текста такой же длины.

Блочные шифры изучены гораздо лучше. Считается, что они обладают более широкой областью применения, чем поточные. Блок текста рассматривается как неотрицательное целое число. Длина блока всегда выбирается равной степени двойки. Обычно используют блоки размером 64 бита. В большинстве блочных алгоритмов симметричного шифрования используются следующие типы операций:

– табличная подстановка, при которой группа бит отображается в другую группу бит, это так называемые *S*-box;
– перемещение, с помощью которого биты сообщения переупорядочиваются;
– операция сложения по модулю 2, обозначаемая XOR или $\oplus$;
– операция сложения по модулю 2^{32} или по модулю 2^{16};
– циклический сдвиг на некоторое число бит.

Эти операции циклически повторяются в алгоритме, образуя так называемые раунды. Входом каждого раунда является выход предыдущего раунда и ключ, который получен по определенному алгоритму из ключа шифрования *K*. Ключ раунда называется подключом.

На рисунке 3 показана структура шифра Файстеля. На вход алгоритма шифрования подаётся блок открытого текста и подключ K_1. Блок открытого текста разделяется на две равные половинки L_0 и R_0, которые последовательно проходят n раундов обработки, а затем объединяются снова для получения блока шифрованного текста соответствующей длины. Для раунда i в качестве входных данных берутся L_{i-1} R_{i-1}, полученные на выходе предыдущего раунда и подключ K_i, вычисляемый по общему ключу *K*. Все ключи K_i отличаются как от общего ключа *K*, так и друг от друга.

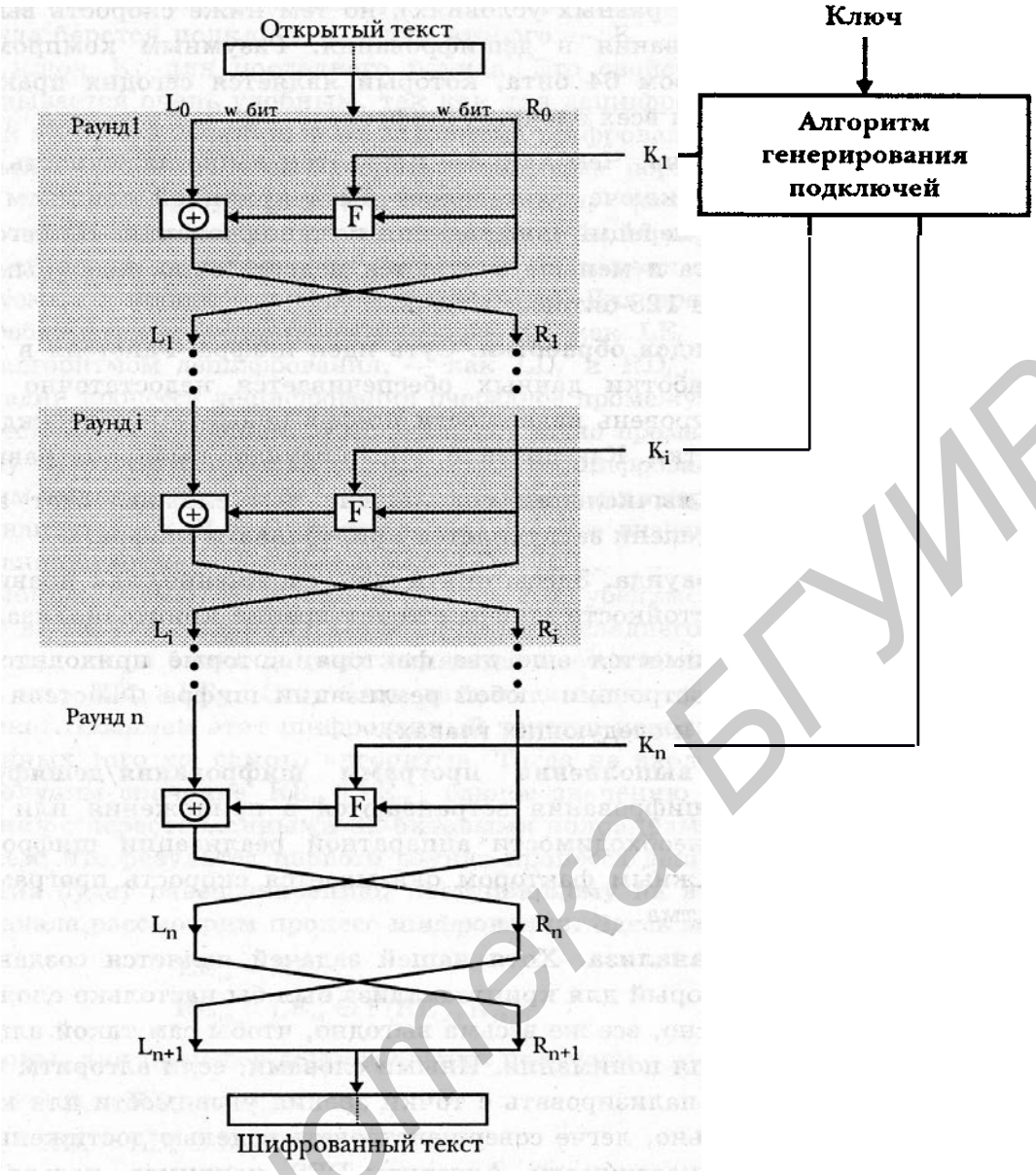


Рисунок 3 – Классическая схема Файстеля

Все раунды обработки проходят по одной и той же схеме: для левой половины выполняется операция подстановки. Она заключается в применении к правой половине блока данных некоторой функции раунда F и последующем сложении по модулю два ($\oplus$)[2)] полученного результата с левой половиной блока данных. Для всех раундов функция F имеет одну и ту же структуру, но зависит от параметра – подключа раунда K_i. После подстановки выполняется перестановка местами обеих половин данных.

Процесс дешифрования шифра Файстеля не отличается от процесса шифрования. Применяется тот же алгоритм, но на вход схемы подаётся шифрованный текст, а подключи K_i используются

[2)] операция XOR.

в обратной последовательности. Это свойство данной схемы шифрования очень удобно, так как для дешифрования не требуется вводить второй алгоритм, отличный от алгоритма шифрования.

В работе [11] показывается, что алгоритм с обратным порядком ключей приводит к нужному результату и расшифровывает текст. Далее в этой же работе доказывается удивительный результат – в схеме Файстеля вовсе не обязательно, чтобы функция F имела обратную.

1.6.2 Криптоанализ

Процесс воссоздания значений X, K, Y из выражений (1) и (2), называется криптоанализом. Используемая стратегия зависит от схемы шифрования. Одной из возможных атак на алгоритм шифрования является лобовая атака, то есть простой перебор всех возможных ключей. Если множество ключей достаточно большое, то подобрать ключ маловероятно. При длине ключа n бит количество возможных ключей равно $2n$. Таким образом, чем длиннее ключ, тем более стойким считается алгоритм для лобовой атаки.

Существуют различные типы атак [4, 5], основанные на том, что противнику известно определенное количество пар «незашифрованное сообщение – зашифрованное сообщение». При анализе зашифрованного текста противник часто применяет статистические методы анализа текста. При этом он может иметь общее представление о типе текста, например, английский или русский текст, исполняемый файл, исходный текст на языке программирования. Во многих случаях криптоаналитик имеет достаточно много информации об исходном тексте. Криптоаналитик может иметь возможность перехвата одного или нескольких незашифрованных сообщений вместе с их зашифрованным видом. Или криптоаналитик может знать основной формат или основные характеристики сообщения. Говорят, что криптографическая схема абсолютно стойкая, если зашифрованное сообщение не содержит информации достаточной для однозначного восстановления соответствующего открытого текста, какой бы большой по объёму шифрованный текст не имелся у противника. Однако среди алгоритмов шифрования нет абсолютно стойких. И поэтому говорят, что криптографическая схема **вычислительно безопасна**, если:

– стоимость взлома шифра превышает стоимость расшифрованной информации;
– время, которое требуется для того чтобы взломать шифр, превышает время, в течение которого информация актуальна.

1.6.3 Алгоритмы традиционного симметричного шифрования

Рассмотрим несколько наиболее известных и популярных алгоритмов симметричного шифрования. Требования, предъявляемые к алгоритмам симметричного шифрования, обычно сводятся к следующим:

– иметь размер блока 64 или 128 бит;
– иметь масштабируемый ключ до 256 бит;
– использовать простые операции, эффективные на микропроцессорах, то есть сложение, сложение по модулю 2, табличные подстановки, умножение по модулю;
– алгоритм должен работать на 8-битном процессоре с минимальными требованиями к памяти;
– алгоритм должен использовать заранее вычисленные подключи;
– всегда должна быть возможность шифрования данных без каких-либо предварительных вычислений;
– алгоритм должен состоять из переменного числа итераций;
– алгоритм не должен иметь слабых ключей, если это невозможно, то количество слабых ключей должно быть минимальным;
– алгоритм должен использовать простую для понимания структуру, это уменьшает закрытость алгоритма.

При построении большинства алгоритмов симметричного шифрования используются блочные алгоритмы, основанные на использовании сети Файстеля. Они удовлетворяют всем требованиям к алгоритмам симметричного шифрования, а с другой стороны, достаточно просты и компактны.

Одним из самых распространённых и наиболее известных алгоритмов симметричного шифрования является DES (Data Encryption Standard – стандарт шифрования данных). Алгоритм был разработан в 1977 г., в 1980 г. был принят NIST (National Institute of Standards and Technolody, США) в качестве стандарта.

DES является классической сетью Файстеля с двумя ветвями. Данные шифруются 64-битными блоками, используя 56-битный ключ. Алгоритм преобразует за несколько раундов 64-битный вход в 64-битный выход. Длина ключа равна 56 битам.

При такой длине ключа существует 256 возможных комбинаций. На сегодня такая длина ключа недостаточна, поскольку допускает успешное применение лобовых атак. Альтернативой DES можно

считать тройной DES, IDEA, а также алгоритм Rijndael, принятый в качестве нового стандарта на алгоритмы симметричного шифрования.

Ещё одним интересным алгоритмом симметричного шифрования является IDEA (International Data Encryption Algorithm). Это блочный симметричный алгоритм шифрования, разработанный в Швейцарском федеральном институте технологий в 1990 г.

IDEA является одним из нескольких симметричных криптографических алгоритмов, которыми первоначально предполагалось заменить DES. IDEA является блочным алгоритмом, который использует 128-битовый ключ для шифрования данных блоками по 64 бита. Целью разработки IDEA было создание относительно стойкого криптографического алгоритма с достаточно простой реализацией.

Алгоритм Blowfish является сетью Файстеля, у которой количество итераций равно 16. Длина блока равна 64 битам, ключ может иметь любую длину в пределах 448 бит. Перед началом шифрования выполняется сложная фаза инициализации, но само шифрование данных выполняется достаточно быстро.

Алгоритм предназначен в основном для приложений, в которых ключ меняется нечасто, во время фазы инициализации происходит аутентификация сторон и согласование общих параметров и секретов. Классическим примером подобных приложений является сетевое взаимодействие. При реализации на 32-битных микропроцессорах с большим кэшем данных Blowfish значительно быстрее DES.

Алгоритм ГОСТ 28147 является российским стандартом для алгоритмов симметричного шифрования. ГОСТ 28147 разработан в 1989 г., является блочным алгоритмом шифрования, длина блока равна 64 битам, длина ключа равна 256 битам, количество раундов равно 32. Алгоритм представляет собой классическую сеть Файстеля.

В рамках американского института стандартов (NIST) была сделана попытка разработать продвинутый стандарт шифрования (Advanced Encryption Standard – AES). Основная цель состояла в создании федерального стандарта, который бы описывал алгоритм шифрования, используемый для защиты информации как в государственном, так и в частном секторе.

В 1998 г. NIST объявил несколько кандидатов на алгоритм AES. Эти алгоритмы были разработаны промышленными и академиче-

скими кругами двенадцати стран. В 1999 г. были представлены выбранные NIST пять финалистов. Ими стали MARS, RC6, Rijndael, Serpent и Twofish.

1.6.4 Шифрование с открытым ключом. Основные требования к алгоритмам асимметричного шифрования

Создание алгоритмов асимметричного шифрования является величайшим достижением в истории криптографии. Алгоритмы шифрования с открытым ключом разрабатывались для того, чтобы решить две наиболее трудные задачи, возникшие при использовании симметричного шифрования.

Первой задачей является распределение ключа. При симметричном шифровании требуется, чтобы обе стороны уже имели общий ключ, который каким-то образом должен быть им заранее передан. Диффи, один из основоположников шифрования с открытым ключом, заметил, что это требование отрицает всю суть криптографии, а именно – возможность поддерживать всеобщую секретность при коммуникациях.

Второй задачей является необходимость создания таких механизмов, при использовании которых невозможно было бы подменить кого-либо из участников, то есть нужна цифровая подпись. При использовании коммуникаций для решения широкого круга задач, например в коммерческих и частных целях, электронные сообщения и документы должны иметь эквивалент подписи, содержащейся в бумажных документах. Необходимо создать метод, при использовании которого все участники будут убеждены, что электронное сообщение было послано конкретным участником. Это более сильное требование, чем аутентификация.

Диффи и Хеллман предложили способ решения обеих задач, который радикально отличается от всех предыдущих подходов к шифрованию. Рассмотрим общие черты алгоритмов шифрования с открытым ключом и требования к этим алгоритмам. Определим требования, которым должен соответствовать алгоритм, использующий один ключ для шифрования, другой ключ – для дешифрования.

В некоторых алгоритмах асимметричного шифрования, например RSA, имеется интересная особенность: каждый из двух ключей может использоваться как для шифрования, так и для дешифрования.

При описании симметричного шифрования и шифрования с открытым ключом будем использовать следующую терминологию.

Ключ, используемый в симметричном шифровании, будем называть секретным ключом. Два ключа, используемые при шифровании с открытым ключом, будем называть открытым ключом и закрытым ключом. Закрытый ключ держится в секрете, но называть его будем закрытым ключом, а не секретным, чтобы избежать путаницы с ключом, используемым в симметричном шифровании. Закрытый ключ пользователя *A* будем обозначать KR_a, открытый ключ – KU_a.

Предполагается, что все участники имеют доступ к открытым ключам друг друга, а закрытые ключи создаются локально каждым участником и распределяться не должны.

В любое время участник может изменить свой закрытый ключ и опубликовать составляющий пару открытый ключ, заменив им старый открытый ключ.

Диффи и Хеллман описывают требования, которым должен удовлетворять алгоритм шифрования с открытым ключом:

1) для стороны *B* процесс генерирования пары ключей (открытый ключ KU_b, закрытый ключ KR_b) не должен вызывать вычислительных трудностей;

2) для отправителя *A* вычислительно легко, имея открытый ключ получателя *B* (KU_b) и незашифрованное сообщение *M*, создать для *B* соответствующее зашифрованное сообщение

$$C = EKU_b(M); \tag{3}$$

3) для получателя *B* не должно вызывать вычислительных трудностей дешифрирование сообщения, если использовать свой закрытый ключ KU_b:

$$M = D_{KRb}(C) = D_{KRb}(E_{KUb}(M)); \tag{4}$$

4) для противника должно быть вычислительно невозможно восстановить личный ключ KR_b из имеющегося открытого ключа KU_b:

$$KR_b = F(KU_b), \tag{5}$$

функция *F* не вычисляется;

5) для противника должно быть вычислительно невозможно восстановить оригинальное сообщение *M*, зная открытый ключ KU_b и зашифрованное сообщение *C*:

$$M = F_{KUb}(C), \tag{6}$$

функция F_{KUb} не вычисляется.

Можно добавить ещё одно, шестое требование, выполняющееся не для всех алгоритмов с открытым ключом;

6) шифрующие и дешифрующие функции могут применяться в любом порядке:

$$M = E_{KUb}(D_{KRb}(M)). \quad (7)$$

Перечисленные требования достаточно сложны для выполнения, что подтверждается тем, что за несколько десятилетий, прошедших со времени открытия метода криптографии с открытым ключом, только один такой алгоритм получил широкое признание. Эти требования сводятся к необходимости нахождения односторонней функции с секретом. Односторонней функцией называется такая функция, у которой каждый аргумент имеет единственное обратное значение, при этом вычислить саму функцию легко, а вычислить обратную функцию трудно:

$Y = f(X)$ – легко,
$X = f^{-1}(Y)$ – трудно,

или другими словами – равенство (3) вычисляется легко, а обратное равенство

$$M = E^{-1}{}_{KU}(C) \quad (8)$$

вычисляется трудно.

Термин «вычислительно легко» означает, что проблема может быть решена за полиномиальное (линейное) время от длины вводимого значения. Так, если длина вводимого значения n бит, то время, требуемое для вычисления функции пропорционально n^a, где a – фиксированная константа. Термин «вычислительно трудно» означает более сложное понятие. В общем случае можно сказать, что функция является практически невычислимой, если усилия по её вычислению возрастают быстрее, чем n^a (полиномиального времени от величины входа). Например, если длина входа n бит, а время вычисления функции пропорционально 2^n, то такая функция считается практически невычислимой.

Вернёмся к определению односторонней функции с секретом, которую, подобно односторонней функции, легко вычислить в одном направлении и трудно вычислить в обратном направлении до тех пор, пока недоступна некоторая дополнительная информация. При наличии этой дополнительной информации инверсию можно вычислить за полиномиальное время. Таким образом, односторон-

няя функция с секретом принадлежит семейству односторонних функций f_k таких, что

$Y = f_k(X)$ – легко, если k и X известны;

$X = f_k^{-1}(Y)$ – легко, если k и Y известны;

$X = f_k^{-1}(Y)$ – трудно, если Y известно, но k неизвестно.

Видно, что разработка конкретного алгоритма с открытым ключом зависит от открытия соответствующей односторонней функции с секретом.

1.6.5 Основные способы использования алгоритмов с открытым ключом

Основными способами использования алгоритмов с открытым ключом являются шифрование/дешифрование, создание и проверка подписи и обмен ключа (рисунок 4).

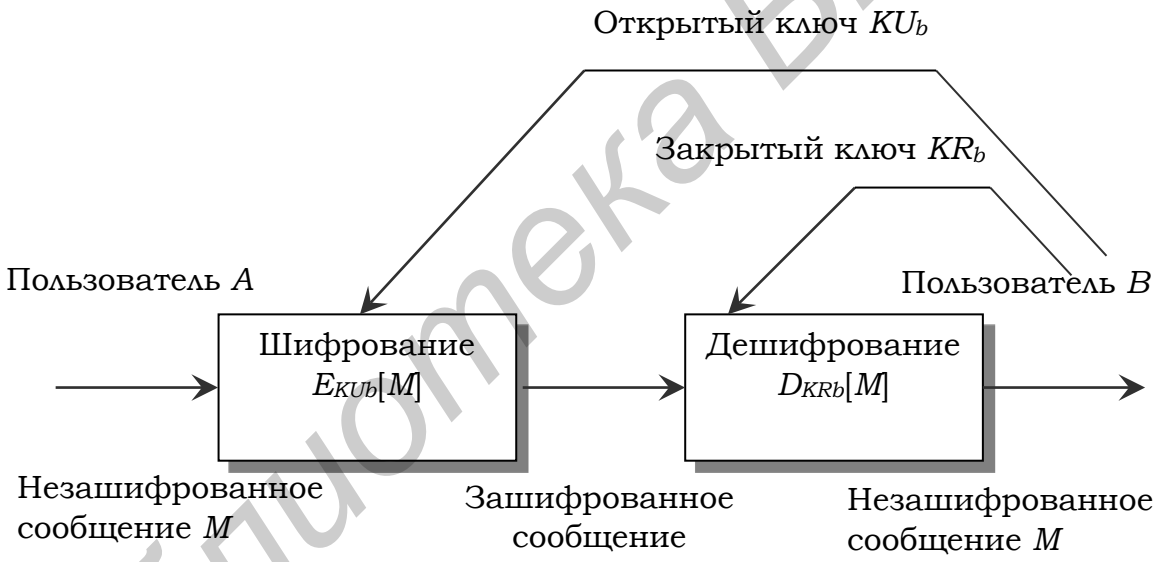


Рисунок 4 – Криптосистема с открытым ключом: защита

Шифрование с открытым ключом состоит из следующих шагов:

1 Пользователь B создает пару ключей KU_b и KR_b, используемых для шифрования и дешифрования передаваемых сообщений.

2 Пользователь B делает доступным некоторым надёжным способом свой ключ шифрования, то есть открытый ключ KU_b. Составляющий пару закрытый ключ KR_b держится в секрете.

3 Если A хочет послать сообщение B, он шифрует сообщение, используя открытый ключ B – KU_b.

4 Когда *B* получает сообщение, он дешифрует его, используя свой закрытый ключ KR_b. Никто другой не сможет дешифровать сообщение, так как этот закрытый ключ знает только *B*.

Если пользователь (конечная система) надежно хранит свой закрытый ключ, то никто не сможет подсмотреть передаваемые сообщения (рисунок 5).

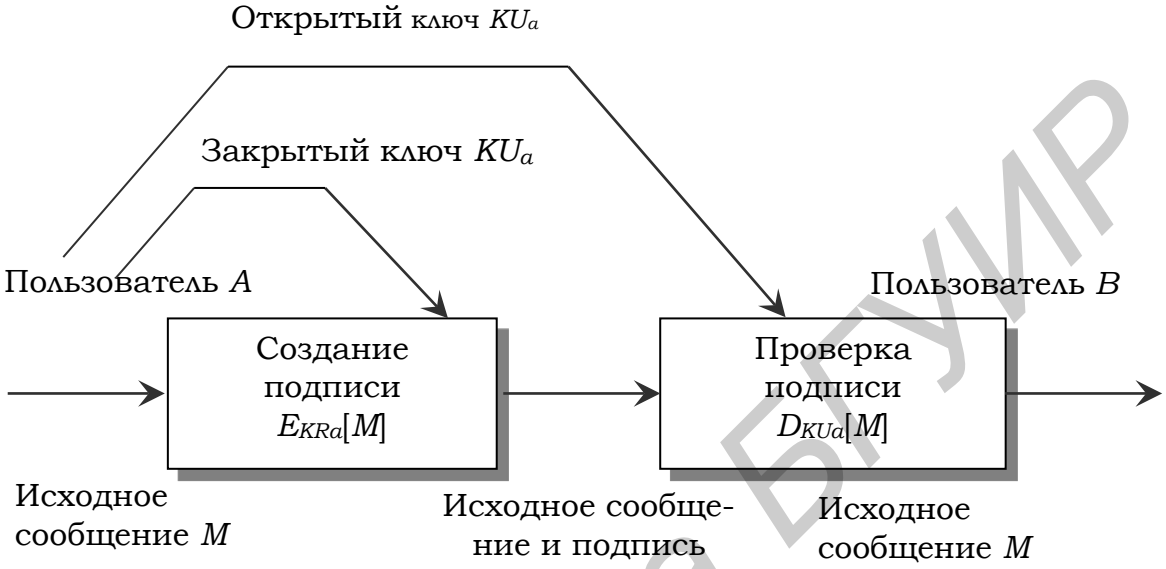


Рисунок 5 – Криптосистема с открытым ключом: аутентификация

Создание и проверка подписи состоит из следующих шагов:

1 Пользователь *A* создает пару ключей KR_a и KU_a, используемых для создания и проверки подписи передаваемых сообщений.

2 Пользователь *A* делает доступным некоторым надёжным способом свой ключ проверки, то есть открытый ключ KU_a. Составляющий пару закрытый ключ KR_a держится в секрете.

3 Если *A* хочет послать подписанное сообщение *B*, он создает подпись $E_{KR}a(M)$ для этого сообщения, используя свой закрытый ключ KR_a.

4 Когда *B* получает подписанное сообщение, он проверяет подпись $D_{KUa}(M)$, используя открытый ключ *A* – KU_a. Никто другой не может подписать сообщение, так как этот закрытый ключ знает только *A*.

До тех пор, пока пользователь или прикладная система надёжно хранит свой закрытый ключ, их подписи достоверны.

Кроме того, невозможно изменить сообщение, не имея доступа к закрытому ключу *A*, обеспечивая тем самым аутентификацию и целостность данных.

В этой схеме все сообщение подписывается, причем для подтверждения целостности сообщения требуется много памяти. Каждое сообщение должно храниться в незашифрованном виде для использования в практических целях. Кроме того, копия сообщения также должна храниться в зашифрованном виде, чтобы можно было проверить в случае необходимости подпись. Более эффективным способом является шифрование небольшого блока бит, который является функцией от сообщения. Такой блок, называемый аутентификатором, должен обладать свойством невозможности изменения сообщения без изменения аутентификатора. Если аутентификатор зашифрован закрытым ключом отправителя, он является цифровой подписью, с помощью которой можно проверить исходное сообщение. Далее эта технология будет рассматриваться в деталях.

Важно подчеркнуть, что описанный процесс создания подписи не обеспечивает конфиденциальность. Это означает, что сообщение, посланное таким способом, невозможно изменить, но можно подсмотреть. Это очевидно в том случае, если подпись основана на аутентификаторе, так как само сообщение передается в явном виде. Но даже если осуществляется шифрование всего сообщения, конфиденциальность не обеспечивается, так как любой может расшифровать сообщение, используя открытый ключ отправителя.

Обмен ключей означает, что две стороны взаимодействуют для обмена ключом сессии, который в дальнейшем можно использовать в алгоритме симметричного шифрования.

Некоторые алгоритмы можно задействовать тремя способами, в то время как другие могут использоваться одним или двумя способами.

Перечислим наиболее популярные алгоритмы с открытым ключом и возможные способы их применения (таблица 1).

Таблица 1 – Наиболее популярные алгоритмы с открытым ключом

Алгоритм	Шифрование/дешифрование	Цифровая подпись	Обмен ключей
RSA	Да, непригоден для больших блоков	Да	Да
DSS	Нет	Да	Нет
Диффи-Хеллман	Нет	Нет	Да

Алгоритм RSA. Диффи и Хеллман определили новый подход к шифрованию, что вызвало к жизни разработку алгоритмов шифрования, удовлетворяющих требованиям систем с открытым ключом. Одним из первых результатов был алгоритм, разработанный в 1977 г. Роном Ривестом, Ади Шамиром и Леном Адлеманом и опубликованный в 1978 г. С тех пор алгоритм Rivest-Shamir-Adleman (RSA) широко применяется практически во всех приложениях, использующих криптографию с открытым ключом.

Алгоритм основан на использовании того факта, что задача факторизации является трудной, то есть легко перемножить два числа, в то время как не существует полиномиального алгоритма нахождения простых сомножителей большого числа.

Алгоритм RSA представляет собой блочный алгоритм шифрования, где зашифрованные и незашифрованные данные являются целыми между 0 и (n – 1) для некоторого n.

1.6.6 Аутентификация сообщений

Рассмотрим основные понятия, относящиеся к обеспечению целостности и аутентификации сообщений с помощью MAC-кода и хэш-функций. Аутентификация сообщений представляет собой процедуру проверки того, что полученное сообщение пришло от указанного источника и не было изменено в пути следования. Цифровая подпись является средством аутентификации и включает меры противодействия возможности оспорить источником или адресатом факт отправки и получения сообщения.

Функции, которые могут служить для создания аутентификатора, делятся на три типа:

– шифрованное сообщение, когда в качестве аутентификатора используется сам шифрованный текст всего сообщения;
– код аутентичности сообщения (Message Authentication Code – MAC), когда в качестве аутентификатора выступает значе-

ние фиксированной длины, создаваемое некоторой открытой функцией сообщения и секретным ключом;
– функция хэширования, когда в качестве аутентификатора используется значение фиксированной длины, создаваемое некоторой открытой функцией от сообщения произвольной длины.

Хэш-функции. Хэш-функцией называется односторонняя функция, предназначенная для получения сообщения или некоторого блока данных постоянной длины при обработке входного сообщения переменной длины.

Хэш-код создается функцией H:

$$h = H(M), \qquad (9)$$

где M является сообщением произвольной длины и h является фиксированным значением функции (хэш-кодом).

Рассмотрим требования, которым должна соответствовать хэш-функция для того, чтобы она могла использоваться в качестве аутентификатора сообщения. Начнём с простого примера хэш-функции и проанализируем несколько подходов к построению хэш-функции.

Хэш-функция H, которая используется для аутентификации сообщений, должна обладать следующими свойствами:

1 Аргументом хэш-функции H может быть блок данных любой длины.
2 Значение, принимаемое хэш-функцией H, всегда фиксированной длины.
3 Значение функции $H(M)$ относительно легко (за полиномиальное время) вычисляется для любого аргумента M.
4 Для любого полученного значения хэш-кода h вычислительно невозможно найти X, такое, чтобы $H(X) = h$. Такое свойство называется односторонностью.
5 Для любого данного X вычислительно невозможно найти $Y \neq X$, чтобы $H(X) = H(Y)$.
6 Должно быть практически невозможно найти произвольную пару значений ($Y \neq X$), такую, что $H(X) = H(Y)$.

Первые три свойства требуют, чтобы хэш-функция создавала хэш-код для любого сообщения.

Четвертое свойство определяет требование односторонности хэш-функции: легко создать хэш-код по данному сообщению, но невозможно восстановить сообщение по данному хэш-коду. Это

свойство важно, если аутентификация с использованием хэш-функции включает секретное значение. Само секретное значение может не посылаться, тем не менее, если хэш-функция не является односторонней, противник может легко раскрыть секретное значение следующим образом. При перехвате передачи атакующий получает сообщение M и хэш-код $C = H(S_{AB} \,||\, M)$. Если атакующий может инвертировать хэш-функцию, то, следовательно, он может получить $S_{AB} \,||\, M = H^{-1}(C)$. Так как атакующий теперь знает и M, и $S_{AB} \,||\, M$, получить S_{AB} совсем просто.

Пятое свойство гарантирует, что невозможно найти другое сообщение, чьё значение хэш-функции совпадало бы со значением хэш-функции данного сообщения. Это предотвращает подделку аутентификатора при использовании зашифрованного хэш-кода.

Хэш-функция MD5. Одним из распространённых алгоритмов получения хэш-функции от сообщения произвольной длины является алгоритм MD5, разработанный Роном Ривестом.

Алгоритм, обрабатывая сообщение произвольной длины, создаёт фиксированную хэш-функцию длиной 128 бит.

Алгоритм MD5 разработан на основе более раннего алгоритма MD4 того же автора – Рона Ривеста. Первоначально данный алгоритм MD4 был опубликован в октябре 1990 г.

Хэш-функция SHA-1. На основе алгоритма MD4 в настоящее время национальным институтом стандартов и технологии (NIST) разработан ещё один алгоритм получения хэш-функции – безопасный хэш-алгоритм (Secure Hash Algorithm – SHA-1). Алгоритм SHA-1 может обработать сообщение максимальной длины 2^{64} бит и создаёт хэш-функцию сообщения длиной 160 бит.

Алгоритм ГОСТ 3411 является российским стандартом для хэш-функций. Его структура довольно сильно отличается от структуры алгоритмов SHA-1,2 или MD5, в основе которых лежит алгоритм MD4. Длина хэш-кода, создаваемого российским алгоритмом ГОСТ 3411, равна 256 битам.

Коды аутентификации сообщений MAC и требования к ним. Напомним, что обеспечение целостности сообщения – невозможность изменения сообщения так, чтобы получатель этого не обнаружил. Под аутентификацией понимается подтверждение того, что информация получена от законного источника, и получателем является тот, кто нужно. Один из способов обеспечения целостности – вычисление MAC (Message Authentication Code). В данном

случае под MAC понимается некоторый аутентификатор, являющийся определённым способом вычисленным блоком данных, с помощью которого можно проверить целостность сообщения. В некоторой степени симметричное шифрование всего сообщения может выполнять функцию аутентификации этого сообщения. Но в таком случае сообщение должно содержать достаточную избыточность, которая позволяла бы проверить, что сообщение не было изменено. Избыточность может быть в виде определённым образом отформатированного сообщения, текста на конкретном языке. Если сообщение допускает произвольную последовательность бит (например, зашифрован ключ сессии), то симметричное шифрование всего сообщения не может обеспечивать его целостность, так как при дешифровании в любом случае получится последовательность битов, правильность которой проверить нельзя. Поэтому гораздо чаще используется криптографически созданный небольшой блок данных фиксированного размера – аутентификатор, так называемая криптографическая контрольная сумма, с помощью которого проверяется целостность сообщения. Этот блок данных и есть MAC. Он может создаваться с помощью секретного ключа, который используют отправитель и получатель. MAC вычисляется в тот момент, когда известно, что сообщение корректно. После этого MAC присоединяется к сообщению и передаётся вместе с ним получателю. Получатель вычисляет MAC, используя тот же самый секретный ключ, и сравнивает вычисленное значение с полученным. Если эти значения совпадают, то с большой долей вероятности можно считать, что при пересылке изменения сообщения не произошло:

$$\text{MAC} = C_K(M). \qquad (10)$$

Хочется заметить, что MAC-код не обеспечивает цифровую подпись, так как и отправитель, и получатель используют один и тот же общий ключ.

1.6.7 Цифровые подписи и протоколы аутентификации

Требования к цифровой подписи. Аутентификация защищает двух участников, которые обмениваются сообщениями, от воздействия некоторой третьей стороны, но не обеспечивает защиту участников друг от друга, тогда как и между ними тоже могут возникать определённые формы споров.

Например, предположим, что адресат *A* посылает адресату *B* аутентифицированное сообщение, и аутентификация осуществля-

ется на основе общего ключа. Рассмотрим возможные недоразумения, которые могут при этом возникнуть:

1 Адресат *B* может подделать сообщение и утверждать, что оно пришло от адресата *A*. Адресату *B* достаточно просто создать сообщение и присоединить аутентификационный код, используя совместный ключ, который имеется у адресата *A* и адресата *B*.
2 Адресат *A* может отрицать, что он посылал сообщение адресату *B*. Так как адресат *B* может подделать сообщение, у него нет способа доказать, что адресат *A* действительно посылал его.

В ситуации, когда обе стороны не доверяют друг другу, необходимо нечто большее, чем аутентификация. Возможным решением подобной проблемы является использование цифровой подписи. Цифровая подпись должна обладать следующими свойствами:

– должна быть возможность проверить автора, дату и время создания подписи;
– должна быть возможность установить достоверность содержимого сообщения (аутентифицировать) на время создания подписи;
– подпись должна быть проверяема третьей стороной в случае возникновения спора.

Таким образом, функция цифровой подписи включает, в частности, функцию аутентификации. На основании этих свойств можно сформулировать следующие требования к цифровой подписи.

1 Подпись должна быть двоичным кодом, который зависит от подписываемого сообщения.
2 Подпись должна использовать некоторую уникальную информацию отправителя для предотвращения подделки (фальсификации) или отказа (отрицания авторства).
3 Создавать цифровую подпись должно быть относительно легко.
4 Цифровую подпись должно быть относительно просто распознать и проверить.
5 Должно быть вычислительно невозможно подделать цифровую подпись как созданием нового сообщения для существующей цифровой подписи, так и созданием ложной цифровой подписи для некоторого сообщения.
6 Цифровые подписи должны быть компактны для удобного хранения в запоминающем устройстве.

Хэш-функции, описанные в подразделе 6.1 и зашифрованные закрытым ключом отправителя, удовлетворяют перечисленным требованиям.

Существует несколько подходов к использованию функции цифровой подписи. Все они могут быть разделены на две категории: прямые и арбитражные.

Прямая и арбитражная цифровые подписи. При использовании **прямой** цифровой подписи взаимодействуют только сами участники, то есть отправитель и получатель. Предполагается, что получатель знает открытый ключ отправителя. Цифровая подпись может быть создана шифрованием всего сообщения или его хэш-кода закрытым ключом отправителя.

Конфиденциальность может быть обеспечена дальнейшим шифрованием всего сообщения вместе с подписью открытым ключом получателя (асимметричное шифрование) или совместным секретным ключом (симметричное шифрование). Заметим, что обычно функция подписи выполняется первой, и только после этого выполняется функция конфиденциальности. В случае возникновения спора некая третья сторона должна просмотреть сообщение и его подпись. Если функция подписи выполняется над зашифрованным сообщением, то для разрешения споров придется хранить сообщение как в незашифрованном виде (для практического использования), так и в зашифрованном (для проверки подписи). Если цифровая подпись выполняется над незашифрованным сообщением, получатель может хранить только сообщение в незашифрованном виде и соответствующую подпись к нему.

Все прямые схемы, рассматриваемые далее, имеют общее слабое место. Действенность схемы зависит от безопасности закрытого ключа отправителя. Если отправитель впоследствии не захочет признать факт отправки сообщения, он может утверждать, что закрытый ключ был потерян или украден, и в результате кто-то подделал его подпись. Можно применить административное управление, обеспечивающее безопасность закрытых ключей, для того чтобы хоть в какой-то степени ослабить эти угрозы. Один из возможных способов состоит в требовании в каждую подпись сообщения включать отметку времени (дату и время) и сообщать о скомпрометированных ключах в специальный центр.

Другая угроза состоит в том, что закрытый ключ может быть действительно украден у отправителя X в момент времени T. Нарушитель может затем послать сообщение, подписанное подписью X и помеченное временной меткой, которая меньше или равна T.

Проблемы, связанные с прямой цифровой подписью, могут быть частично решены с помощью арбитра. Существуют различные схемы с применением **арбитражной** подписи. В общем виде арбитражная подпись выполняется следующим образом. Каждое подписанное сообщение от отправителя *X* к получателю *Y* первым делом поступает к арбитру *A*, который проверяет подпись для данного сообщения. После этого сообщение датируется и посылается к *Y* с указанием того, что оно было проверено арбитром. Присутствие *A* решает проблему схем прямой цифровой подписи, при которых *X* может отказаться от сообщения.

В таких схемах арбитр играет исключительно важную роль, и все участники должны ему доверять. Рассмотрим несколько примеров схем арбитражной цифровой подписи.

Симметричное шифрование, при котором арбитр может видеть сообщение

Обозначим:

X – отправитель;
Y – получатель;
A – арбитр;
M – открытое сообщение;
E – алгоритм шифрования;
K_{X_A} – совместный ключ *X* и *A;*
K_{Y_A} – совместный ключ *Y* и *A;*
K_{X_Y} – совместный ключ *X* и *Y;*
H – функция хэширования;
|| – операция конкатенации;
T – метка времени.

В этих обозначениях последовательность действий участников цифровой подписи можно описать следующими формулами:

$$X \to A: \quad M \,||\, E_{K_{X_A}}(ID_X \,||\, H(M)). \qquad (11)$$

Предполагается, что отправитель *X* и арбитр *A* используют общий секретный ключ K_{X_A} и что *A* и *Y* используют общий секретный ключ K_{Y_A}. *X* создаёт сообщение *M* и вычисляет его хэш-значение *H*(*M*). Затем *X* передаёт сообщение с добавленной к нему подписью арбитру *A*. Подпись складывается из идентификатора *X* и хэш-

значения, всё это зашифровано с использованием ключа K_{X_A}. Арбитр *A* дешифрует подпись и проверяет хэш-значение:

$$A \rightarrow Y: \quad E_{K_{Y_A}}(ID_X \,||\, M \,||\, E_{K_{X_A}}(ID_X \,||\, H(M)) \,||\, T). \qquad (12)$$

Затем *A* передает сообщение к *Y*, шифруя его K_{Y_A}. Сообщение включает ID_X, первоначальное сообщение от *X*, подпись и отметку времени. Получатель *Y* может дешифровать его для получения сообщения и подписи. Отметка времени информирует *Y* о том, что данное сообщение не устарело и не является повтором. *Y* может сохранить *M* и подпись к нему. В случае спора *Y*, который утверждает, что получил сообщение *M* от *X*, посылает следующее сообщение к арбитру *A*:

$$E_{K_{Y_A}}(ID_X \,||\, M \,||\, E_{K_{X_A}}(ID_X \,||\, H(M)) \,||\, T). \qquad (13)$$

Арбитр использует K_{Y_A} для получения ID_X, *M* и подписи, а затем, используя K_{X_A}, может дешифровать подпись и проверить хэш-код. По этой схеме *Y* не может прямо проверить подпись *X*; подпись используется исключительно для разрешения споров. *Y* считает сообщение от *X* аутентифицированным, потому что оно прошло через *A*. В данном сценарии обе стороны должны иметь высокую степень доверия к *A*:

– *X* должен доверять *A* в том, что тот не разгласит K_{Xa} и не будет создавать фальшивые подписи вида $E_{K_{X_A}}(ID_X \,||\, H(M))$;

– *Y* должен верить, что *A* будет посылать $E_{K_{Y_A}}(ID_X \,||\, M \,||\, E_{K_{X_A}}(ID_X \,||\, H(M)) \,||\, T)$ только в том случае, если хэш-значение является корректным и подпись была создана *X*;

– обе стороны должны быть уверены, что *A* будет честно разрешать спорные вопросы.

Симметричное шифрование, при котором арбитр не видит сообщение

Если арбитр не является такой доверенной стороной, то *X* должен добиться того, чтобы никто не мог подделать его подпись, а *Y* должен добиться того, чтобы *X* не мог отвергнуть свою подпись.

Предыдущий сценарий также предполагает, что *A* имеет возможность читать сообщения от *X* к *Y* и что возможно любое подсматривание. Рассмотрим сценарий, который, как и прежде, использу-

ет арбитраж, но при этом ещё гарантируется конфиденциальность. В таком случае также предполагается, что X и Y используют общий секретный ключ K_{XY}:

$$X \to A: \quad ID_X \,||\, E_{K_{X_Y}}(M) \,||\, E_{K_{X_A}}(ID_X \,||\, H(E_{K_{X_Y}}(M))). \qquad (14)$$

X передает A свой идентификатор, сообщение, зашифрованное K_{XY}, и подпись. Подпись состоит из идентификатора и хэш-значения зашифрованного сообщения, которые зашифрованы с использованием ключа K_{X_A}. A дешифрует подпись и проверяет хэш-значение. В данном случае A работает только с зашифрованной версией сообщения, что предотвращает его чтение.

Следующий шаг в этой схеме – A после проверки передаёт адресату Y:

$$A \to Y: \quad E_{K_{Y_A}}(ID_X \,||\, E_{K_{X_Y}}(M) \,||\, E_{K_{X_A}}(ID_X \,||\, H(E_{K_{X_Y}}(M))) \,||\, T). \qquad (15)$$

Арбитр A передает Y всё, что он получил от X плюс отметку времени, шифруя всё с использованием ключа K_{Y_A}.

Хотя арбитр и не может прочитать сообщение, он в состоянии предотвратить подделку любого из участников – X или Y. Остаётся проблема, как и в первом сценарии, что арбитр может сговориться с отправителем, отрицающим подписанное сообщение, или с получателем, для подделки подписи отправителя.

Шифрование открытым ключом, при котором арбитр не видит сообщение

Все упомянутые сложности могут быть решены с помощью открытого ключа по следующей схеме:

$$X \to A: \quad ID_X \,||\, E_{K_{R_X}}(ID_X \,||\, (E_{K_{U_Y}}(E_{K_{R_X}}(M))). \qquad (16)$$

В этом случае X осуществляет двойное шифрование сообщения M, сначала своим закрытым ключом K_{R_X}, а затем открытым ключом Y – K_{U_Y}. Получается подписанная секретная версия сообщения. Теперь это подписанное сообщение вместе с идентификатором X шифруется K_{R_X} и вместе с ID_X посылается A. Внутреннее, дважды зашифрованное сообщение недоступно арбитру (и всем, исключая Y). Однако A может дешифровать внешнюю шифрацию, чтобы

убедиться, что сообщение пришло от *X* (так как только *X* имеет K_{R_X}). Проверка даёт гарантию, что пара «закрытый/открытый ключ» законна, и тем самым верифицирует сообщение:

$$A \to Y: \quad E_{K_{R_A}}(ID_X \,||\, (E_{K_{U_Y}}(E_{K_{R_X}}(M)) \,||\, T). \qquad (17)$$

Затем *A* передаёт сообщение *Y*, шифруя его K_{R_A}. Сообщение включает ID_X, дважды зашифрованное сообщение и отметку времени.

Эта схема имеет ряд преимуществ по сравнению с предыдущими двумя схемами. Во-первых, в совместном распоряжении сторон до начала обмена данными нет никакой информации, что предотвращает возможность сговора с целью обмана. Во-вторых, некорректные данные не могут быть посланы, даже если K_{R_X} скомпрометирован, при условии, что не скомпрометирован K_{R_A}. В заключение, содержимое сообщения от *X* к *Y* неизвестно ни *A*, ни кому бы то ни было ещё.

Стандарты цифровой подписи. Национальный институт стандартов и технологии США (NIST) разработал федеральный стандарт цифровой подписи DSS. Для создания цифровой подписи используется алгоритм DSA (Digital Signature Algorithm). В качестве хэш-алгоритма стандарт предусматривает использование алгоритма SHA-1 (Secure Hash Algorithm). DSS первоначально был предложен в 1991 г. и пересмотрен в 1993 г.

DSS использует алгоритм, который разрабатывался для использования только в качестве цифровой подписи. В отличие от RSA его нельзя использовать для шифрования или обмена ключами. Тем не менее это технология открытого ключа.

В российском стандарте ГОСТ 3410, принятом в 1994 г., используется алгоритм, аналогичный алгоритму, реализованному в стандарте DSS. Оба алгоритма относятся к семейству алгоритмов ElGamal.

В стандарте ГОСТ 3410 используется хэш-функция ГОСТ 3411, которая создаёт хэш-код длиной 256 бит. Это во многом обуславливает требования к выбираемым простым числам *p* и *q*.

Еще раз обратим внимание на отличия DSS и ГОСТ 3410:

– используются разные хэш-функции (в ГОСТ 3410 применяется российский стандарт на хэш-функции ГОСТ 3411, в DSS

используется SHA-1), они имеют разную длину хэш-кода. Отсюда и разные требования на длину простого числа *q*: в ГОСТ 3410 длина q должна быть от 254 до 256 бит, а в DSS длина *q* должна быть от 159 до 160 бит;

– по-разному вычисляется компонента *s* подписи.

Подписи, созданные с использованием стандартов ГОСТ 3410 или DSS, называются **рандомизированными**, так как для одного и того же сообщения с использованием одного и того же закрытого ключа каждый раз будут создаваться разные подписи (*r*, *s*), поскольку каждый раз будет использоваться новое значение *k*. Подписи, созданные с применением алгоритма RSA, называются **детерминированными**, так как для одного и того же сообщения с использованием одного и того же закрытого ключа каждый раз будет создаваться одна и та же подпись.

1.7 Цифровые сертификаты

Одна из главных проблем асимметричных криптосистем состоит в том, что пользователи должны постоянно следить, зашифровывают ли они сообщения истинными ключами своих корреспондентов. В среде свободного обмена открытыми ключами через общественные серверы-депозитарии атаки по принципу «человек посередине» представляют серьёзную потенциальную угрозу. В этом виде атак злоумышленник подставляет пользователю собственный ключ, но с именем предполагаемого адресата; данные зашифровываются подставным ключом, перехватываются его владельцем-злоумышленником, попадая в итоге в чужие руки.

В среде криптосистем с открытым ключом критически важно, чтобы вы были абсолютно уверены, что открытый ключ, которым собираетесь что-то зашифровать – не искусная имитация, а истинная собственность вашего корреспондента. Можно шифровать только теми ключами, которые были переданы вам их владельцами из рук в руки на дискетах. Но предположим, что нужно связаться с человеком, живущим на другом краю света, с которым вы даже незнакомы; как вы можете быть уверены, что получили его подлинный ключ?

Цифровые сертификаты ключей упрощают задачу определения принадлежности открытых ключей предполагаемым владельцам. Дополнительный материал по обсуждаемой теме можно найти на сайте – http://www.pgpru.com/, где помимо материалов по цифровым сертификатам, модераторами поддерживается форум, на

котором обсуждаются проблемы шифрования при использовании электронной почты.

Цифровой сертификат состоит из трёх компонентов:

- открытого ключа, к которому он приложен;
- данных, или записей, сертификата (сведения о личности пользователя);
- одной или нескольких ЭЦП (электронных цифровых подписей), связывающих ключ с сертификатом.

Цель ЭЦП на сертификате – указать, что сведения сертификата были заверены доверенным третьим лицом или организацией. В то же время цифровая подпись не подтверждает достоверность сертификата; она является поручительством того, что подписанная запись сертификата связана с данным открытым ключом.

Цифровой сертификат – это открытый ключ с прикреплёнными к нему идентификатором и отметкой подтверждения от доверенного лица.

1.7.1 Распространение сертификатов

Цифровые сертификаты применяются, когда нужно обменяться с кем-нибудь ключами. Небольшим группам людей, нуждающимся в защищённой связи, не составит труда просто передать друг другу дискеты или отправить электронные письма, содержащие копии их ключей. Таким образом происходит ручное распространение открытых ключей, и оно эффективно только до определённого этапа. Дальнейшее – за пределами возможностей данного метода, и тогда возникает необходимость развёртывания системы, которая бы обеспечивала достаточную надёжность и безопасность, предоставляла возможности хранения и обмена ключами, так что коллеги, бизнес-партнёры или незнакомцы смогли бы отправлять друг другу зашифрованные сообщения, если в том возникнет необходимость.

Такая система может реализоваться в форме простого хранилища-депозитария, называемого сервером сертификатов или сервером-депозитарием открытых ключей, либо иметь более сложную и комплексную структуру, предполагающую дополнительные возможности администрирования ключей и называемую инфраструктурой открытых ключей (Public Key Infrastructure, PKI).

Сервер-депозитарий – это сетевая база данных, позволяющая пользователям оставлять и извлекать из неё цифровые сертификаты. Сервер ключей также может иметь некоторые функции адми-

нистрирования, помогающие организации поддерживать свою политику безопасности.

Обеспечение безопасного обмена информацией в современных электронных системах реализуется разными способами. Наиболее широкое распространение получили системы на основе открытых ключей PGP и PKI. Подтверждение личности или сообщения – основное предназначение описываемых систем – реализуется с помощью связки цифровых кодов (или сертификатов), ЭЦП, открытого и закрытого ключей. Это общее для обеих систем. Главной особенностью PKI в отличие от PGP является наличие ЦС (центра сертификации Certification Authority, CA) и ЦР (центра регистрации Registration Authority, RA). Благодаря им возможно подтверждение подлинности личности сторонними уполномоченными организациями.

Наличие ЦС и ЦР обусловило то, что в системе PKI доминирующим направлением подтверждения подлинности является вертикальная (иерархическая) составляющая, когда сертификат подтверждается кем-то, имеющим более высокий статус. В системе PGP основной является горизонтальная составляющая или, другими словами, схема «прямого доверия». Хотя и в той, и в другой системе возможно как вертикальное, так и горизонтальное подтверждение подлинности. Образно можно представить, что в PKI доверие распространяется в виде дерева, а в PGP – в виде сети.

PKI, как и простой сервер-депозитарий, имеет базу данных для хранения сертификатов, но и предоставляет сервисы и протоколы по управлению открытыми ключами. В них входят возможности выпуска (издания), отзыва (аннулирования) и системы доверия сертификатов. Главной же особенностью PKI является наличие компонентов, известных как ЦС и ЦР.

ЦС издаёт цифровые сертификаты и подписывает их своим закрытым ключом. Из-за важности своей роли ЦС является главным компонентом инфраструктуры PKI. Используя открытый ключ ЦС, любой пользователь, желающий проверить подлинность конкретного сертификата, сверяет подпись ЦС и, следовательно, удостоверяется в целостности содержащейся в сертификате информации и, что более важно, во взаимосвязности сведений сертификата и открытого ключа.

ЦР – это информационная система выдающая сертификаты своим зарегистрированным пользователям. Наличие ЦР для ЦС необязательно, но оно обеспечивает разделение функций, которое иногда необходимо.

1.7.2 Форматы сертификатов

Цифровой сертификат может быть представлен несколькими форматами:

– сертификаты OpenPGP (чаще называемые просто ключами PGP);
– сертификаты X.509.

Эти форматы различаются из-за различия в стандартах. Различия в стандартах привели к разным свойствам перечисленных форматов. Ниже приводится краткая выжимка соответствующих свойств этих форматов.

В основу **PGP** положен стандарт OpenPGP, который содержит:
– сведения о владельце сертификата;
– открытый ключ владельца сертификата;
– ЭЦП владельца сертификата;
– период действия сертификата;
– предпочтительный алгоритм шифрования.

Сертификаты **X.509** используют идеологию PKI (инфракструктуры открытых ключей) и в таком сертификате содержится следующая информация:
– открытый ключ владельца сертификата;
– серийный номер сертификата;
– уникальное имя владельца;
– период действия сертификата;
– уникальное имя издателя;
– ЭЦП издателя и идентификатор алгоритма подписи.

Существует множество версий формата X.509, но между ними всеми и форматом сертификатов PGP имеется ряд фундаментальных различий:
– сертификат PGP создаётся только лично, а сертификат X.509 вы должны запросить и получить от ЦС;
– сертификаты X.509 содержат только одно имя владельца сертификата;
– сертификаты X.509 содержат только одну ЭЦП, подтверждающую подлинность сертификата.

Обе эти технологии используются для следующих целей:
– обеспечение механизма строгой аутентификации;
– организация защищённого обмена электронной почтой;
– организация виртуальных частных сетей (VPN) для защищённых соединений удалённых пользователей [6];

– организация защищённых сайтов (доступ через Интернет), систем разграничения доступа к сайтам и приложениям.

1.7.3 Подлинность, доверие и их проверка

Любой пользователь в среде криптосистем с открытым ключом рискует рано или поздно принять по неосторожности поддельный ключ за настоящий. *Достоверность* или *подлинность* ключа состоит в том, что конкретный открытый ключ принадлежит именно тому владельцу, чья идентификационная информация указана в сертификате ключа.

Согласно системе PKI (см. пункт 1.7.1) некоторые компании уполномочивают ЦС на проверку подлинности сертификатов. В организации, использующей PKI с сертификатами X.509, задача ЦР состоит в приёме запросов на сертификаты, а задача ЦС – в выдаче сертификатов конечным пользователям.

В организации, использующей сертификаты PGP без PKI, задача ЦС состоит в проверке достоверности всех PGP-сертификатов и подписании подлинных. Основная цель ЦС – собственной подписью «связать» открытый ключ с идентификационной информацией, содержащейся в сертификате, чем заверить третьих лиц, что были приняты определённые меры по установлению связи между ключом и идентификационными сведениями.

ЦС в организации – краеугольный камень системы подлинности и доверия; в некоторых организациях, как, например, в тех, которые используют PKI, ни один сертификат не считается подлинным, пока не будет подписан доверенным ЦС.

Определение подлинности сертификата – некоторая механическая процедура, которую можно свести к следующим вариантам:

– прямая, физическая передача копии открытого ключа, вручённая вам на носителе – флэшке, дискете, компакт-диске;
– сверка отпечатка (fingerprints) сертификата, то есть проверка значения хэш-функции сертификата пользователя, которое показано как одно из его свойств;
– прямой коммуникационный контакт с владельцем ключа, например, по телефону с целью сверки вашего отпечатка с отпечатком телефонного корреспондента, если вам знаком голос корреспондента;
– довериться мнению третьей стороны, уже установившей подлинность сомнительного сертификата.

ЦС, например, ответственен за детальную проверку принадлежности открытого ключа предполагаемому владельцу перед выдачей ему сертификата. Любой пользователь, доверяющий ЦС, будет автоматически расценивать подлинными все сертификаты, подписанные ЦС.

В большинстве случаев пользователи полностью полагаются на ЦС в проверке подлинности сертификатов, то есть они убеждены, что ЦС провёл всю процедуру проверки, и уверены в его поручительствах за подлинность заверенных им сертификатов.

Модели отношений доверия. В относительно закрытых системах, таких, как небольшие организации и фирмы, можно без труда отследить путь сертификата в ЦС. Однако пользователям зачастую приходится связываться с людьми за пределами их корпоративной среды, включая и таких, с которыми они прежде никогда не встречались. Установление линии доверия с теми, кто не был явно удостоверен ЦС, становится непростой задачей.

Организации следуют одной из нескольких моделей отношений доверия, которые диктуют пользователям их действия по определению подлинности сертификатов. Существуют три различные модели:

– прямое доверие;
– иерархическое доверие;
– сеть доверия (Web of Trust).

Прямое доверие – простейшая из моделей отношений доверия. В этой схеме пользователь убеждён, что ключ подлинный, поскольку точно знает, от кого получил этот ключ. Все криптосистемы в той или иной мере используют эту форму доверия.

В PGP пользователь, заверяющий ключи самостоятельно, не прибегая к помощи доверенных поручителей, использует схему прямого доверия.

В **иерархической системе** доверия существует ряд корневых сертификатов, от которых распространяется доверие. Эти сертификаты могут либо сами заверять сертификаты конечных пользователей, либо они могут уполномочивать другие сертификаты, которые будут заверять сертификаты пользователей по некоторой цепи. Это как бы большое «дерево» доверия.

Сеть доверия объединяет обе предыдущие модели, также привнося принцип – чем больше информации, тем лучше. Таким образом, сеть доверия – это *накопительная модель доверия*. Серти-

фикат может быть доверяем напрямую или по некоторой цепочке, уходящей к напрямую доверяемому корневому сертификату.

В PGP именно такое представление о доверии. PGP использует цифровые подписи как собственный вид поручительства. Когда один пользователь подписывает ключ другого, он становится поручителем этого ключа. Этот процесс, расширяясь, образует сеть доверия PGP. В среде PGP любой пользователь может выступать в качестве ЦС, поскольку каждый пользователь может заверить открытый ключ другого пользователя.

1.7.4 Аннулирование сертификата

Применение сертификата допустимо, только пока он достоверен. Опасно полагаться на то, что сертификат будет защищён и надёжен вечно. В большинстве организаций и во всех PKI сертификат имеет ограниченный срок «жизни». Это сужает период, в который система может оказаться под угрозой, если сертификат окажется взломан.

Сертификат создаётся с определённым заданным периодом достоверности, начинающимся с даты создания и заканчивающимся датой истечения. Сертификат может быть использован в течение всего периода действия, по истечении которого перестаёт быть верным, поскольку достоверность его идентификационно-ключевой пары более не может быть гарантирована.

Но иногда появляется потребность сделать сертификат недействительным до истечения срока его «жизни», например, в случае увольнения владельца сертификата с настоящего места работы или когда у владельца возникает подозрение, что закрытый ключ данного сертификата был скомпрометирован. Такой процесс называется **отзывом**, или **аннулированием**. Аннулированный сертификат гораздо более подозрителен, нежели истекший. Истекший сертификат более непригоден к использованию, однако не несёт такой угрозы скомпрометированности, как аннулированный.

В системе PGP любой пользователь, заверивший сертификат, в любой момент может отозвать с него свою подпись, используя тот же закрытый ключ, которым её создавал. Отозванная подпись указывает на то, что заверитель счёл, что открытый ключ и идентификационная информация больше не связаны друг с другом или что открытый ключ сертификата был скомпрометирован. Отозванная подпись имеет практически такое же значение, как и аннулированный сертификат.

В системе PKI в случае сертификатов X.509 отозванная подпись фактически представляет то же самое, что и аннулированный сертификат, поскольку лишь одна подпись была поручительством подлинности сертификата – подпись ЦС. PGP предоставляет дополнительную возможность аннулирования всего сертификата, а не только подписей на нём, если вы вдруг посчитаете, что он был каким-либо образом скомпрометирован.

Только владелец сертификата может аннулировать сертификат Open PGP. Сертификат X.509 может быть отозван только его издателем – ЦС по запросу владельца.

Уведомление об аннулировании сертификата. После аннулирования сертификата крайне важно оповестить всех потенциальных корреспондентов, что он более недействителен. Наиболее простой способ оповещения в среде PGP – размещение аннулированного сертификата на сервере-депозитарии. Таким образом, все, кто могут решить связаться с вами, будут предупреждены не использовать этот открытый ключ.

В среде PKI уведомление об аннулировании сертификатов осуществляется посредством специального механизма, называемого списком аннулированных сертификатов (Certificate Revocation List, CRL), публикуемым ЦС. CRL содержит датированный заверенный список всех аннулированных непросроченных сертификатов системы. ЦС обновляет CRL через регулярные промежутки времени.

1.7.5 Так что же всё-таки лучше – PGP или PKI?

Все версии программы PGP способны обслуживать не более, скажем, 20 000 пользователей и не рассчитаны на получение сертификатов от сторонних ЦС, поскольку этого не допускает используемый ими протокол OpenPGP. Система PGP изначально создавалась под потребности в основном частных пользователей и предназначена в основном для работы с электронной почтой. Имея дело с PGP-сертификатом, каждый пользователь может выступать в качестве заверителя содержащихся в нём сведений (за исключением случаев, когда эта возможность намеренно ограничена политикой безопасности). В последующем PGP стали приспосабливать под потребности защиты электронного документооборота. Все бы хорошо, но возникает проблема доверия сторонним корреспондентам. Что толку от заверений сертификата пользователя, который прислал вам письмо, если вы не знакомы и не доверяете никому из подписавших полученный вами сертификат?

Система PGP вполне может выступить удачным решением для внутрикорпоративных целей, когда заверителем сертификата является уполномоченное администрацией компании лицо. Но, отправляя документы во внешний мир, вы должны подтверждать свою личность отправителя. Аналогично и в случае с получением корреспонденции: вы должны быть уверены в том, что сообщение отправлено именно тем, кто назвался отправителем. Для юридически правильного документооборота в этом случае необходимо заверение подлинности сертификата сторонней уполномоченной организацией. Такую возможность может предоставить только протокол X.509, на основе которого и построена PKI.

Но проблема аутентификации не сводится только к определению авторства послания. С помощью PKI можно организовать систему доступа к данным, например с помощью ключей, размещённых на внешних носителях.

Задача криптографической защиты электронных почтовых сообщений, файлов и документов сводится к шифрованию данных. И PGP, и программные продукты, реализующие логику PKI, позволяют шифровать информацию. Для частной переписки этих мер безопасности может быть вполне достаточно, но для работы государственных и коммерческих организаций этих возможностей недостаточно. Прежде всего потому, что PGP не работает с национальными стандартами, а значит, не отвечает ни необходимому уровню безопасности, ни существующим законодательным нормам разных стран.

Обе технологии рассчитаны на использование в незащищённых сетях, работающих по протоколу IP. Однако различия в возможностях этой работы существенны. Благодаря наличию системы сертификатов PKI может реализовать взаимодействие в сетях по протоколам HTTPS и SSL, в то время как область применения PGP – это фактически только электронная почта.

Программы шифрования данных особенно полезны именно при работе в незащищённых сетях. Даже в случае использования незащищенного HTTP с помощью программ шифрования можно организовать обмен зашифрованными документами через хранилища данных. Пользователь может зашифровать документы любым доступным встроенным криптопровайдером, поместить зашифрованный файл в хранилище и дать ссылку на скачивание своему корреспонденту. В системе PGP в этом варианте обмена не возникает вообще никаких проблем, в системе PKI необходимо, чтобы у пользователей были одинаковые криптопровайдеры.

Системы PGP и PKI очень похожие, но всё же разные. PGP предназначена для неструктурированного защищённого обмена данными. PKI обеспечивает обмен зашифрованными данными как на локальном, так и на межсетевом уровне с достаточной степенью защищённости и достоверности. В настоящее время обе технологии развиваются в сторону обоюдной совместимости.

Замечание – Авторы вынуждены констатировать, что на настоящий момент среди обычных пользователей, обменивающихся между собой обычной почтовой корреспонденцией с помощью популярных клиентских почтовых программ The Bat, Thunderbird, Outlook, использование шифрования с открытым ключом и методов асимметричного шифрования – большая редкость. Во-первых, это говорит о том, что обе существующие системы открытых ключей и их сертификация достаточно сложны и громоздки при работе и обычным пользователем не воспринимаются и отвергаются. Во-вторых, привносит свой вклад путаница двух систем сертификации: openPGP и PKI, и если корреспонденты, решившие при переписке воспользоваться открытыми ключами, сертифицировали их в разных системах, которые описаны выше, то «головная боль» при настройке их клиентских почтовых программ им обеспечена. Как показывает практика, это не простая задача даже для профессионалов, не говоря уже про обычного пользователя.

2 Практическая часть

Одним из основных компонентов организации безопасности информационной системы является криптозащита и шифрование данных, использующее методы криптографии. Авторы, читая этот курс несколько лет, отказались от изложения серьезной математической базы и математических основ криптографии и ограничились кратким обзором методов симметричного шифрования и методов шифрования с открытым ключом. Но сделали акцент на практическом использовании и освоении этих методов на примере конкретных программных продуктов. Далее предлагается выполнить ряд несложных практических заданий, иллюстрирующих изложенный в первой части теоретический материал.

Практические задания выполняются с использованием несложных бесплатных программных пакетов, которые любой читатель без труда загрузит по приведенным в надлежащем месте ссылкам.

2.1 Основные понятия и применение стеганографии

В большинстве случаев в практике обмена и передачи данных и сообщений нужно не столько зашифровать данные, сколько просто скрыть сам факт обмена и передачи сообщений. Практика написания текста молоком между строк с целью дальнейшего проявления и визуализации сообщения тоже восходит к тем временам. Методы, использующие маскировку и сокрытие самого факта наличия сообщения и его передачи, изучает стеганография.

Что изменилось в стеганографии с появлением компьютеров, и как всё это работает с использованием компьютерных программ и данных? Принципы стеганографии в компьютерную эру подверглись минимальным изменениям и остались такими же простыми. Файл с данными, факт передачи которого вы хотите скрыть, может быть любым: это может быть текст, изображение, бинарный файл, мультимедийный объект. С другой стороны, эти данные специальным образом внедряются в так называемый файл-носитель (carrier file). Естественно, файл-носитель внешне должен быть совершенно безобидным и не вызывать никаких подозрений у хакеров, нацеленных вас атаковать, – рекламная картинка, текст шлягера, нейтральные фотографии цветов и домашних животных.

Второе требование к файлу-носителю – он должен быть «рыхлым», то есть содержать достаточное количество избыточных данных. Такие файлы обычно очень хорошо упаковываются архиваторами. Форматы таких файлов известны – *.bmp, *.txt, *.html, *.pdf.

Именно такие файлы используются в качестве файлов-носителей. Программа, осуществляющая все эти функции, подходит для нашей задачи. Такие программы обычно небольшие и имеются в свободном доступе в Интернете. Стоит вам набрать в любой поисковой системе «program steganography» и вы получите список из сотен названий.

Выполним некоторые практические задания. Используем программу wbStego4, которая написана программистом Werner Bailer и загружена с его web-страницы. В дистрибутивном пакете автор любезно предоставил регистрационное имя и регистрационный ключ.

2.1.1 Внедрение данных в файл-носитель программой wbStego4

Работа с программой состоит из ряда этапов.

Первый этап. На первом этапе запускаем программу и видим интерактивное окно (рисунок 6). Программа информирует о своих возможностях и предлагает сделать выбор режима работы. По умолчанию предлагается пошаговый режим. Программа сообщает основные сведения: кодирование/декодирование, режимы работы, типы файлов-носителей.

Для продвинутых пользователей имеется другой, альтернативный режим. Выбор режима работы и переход к другому стилю интерфейса осуществляется нажатием на Flowchart-Mode (режим блок-схемы).

При щелчке по кнопке Help программа предлагает некоторые сведения из стеганографии и справку по интерфейсу управления программой.

Упражнение для самостоятельной работы 1. Нажмите кнопку Help и просмотрите сведения, которые хочет сообщить вам программа.

Упражнение для самостоятельной работы 2. Нажмите кнопку Flowchart-Mode и перейдите в этот режим. Проделайте приведенное ниже упражнение в режиме «Блок-схема».

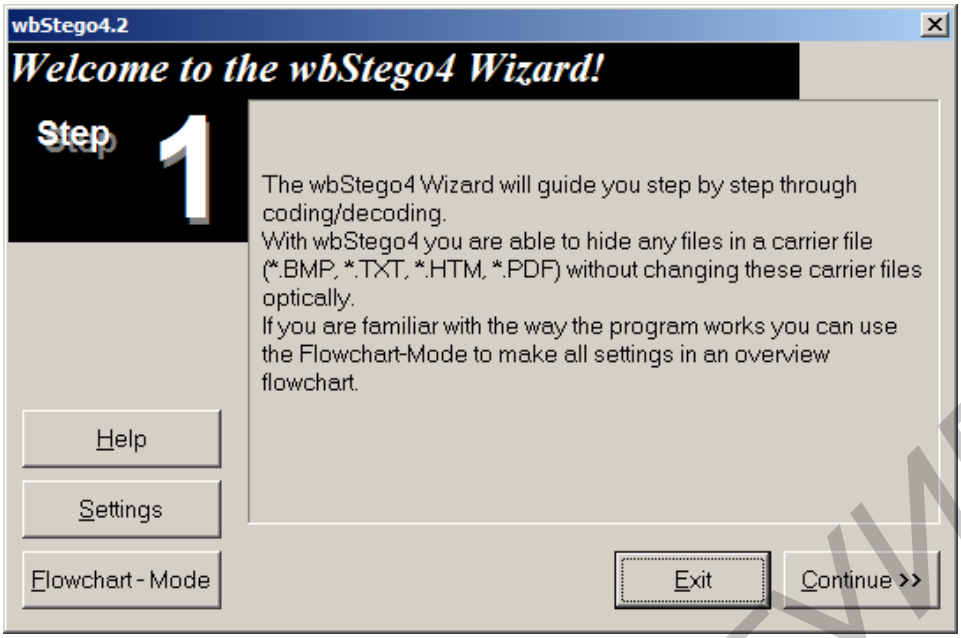


Рисунок 6 – Первый шаг работы
с программой wbStego4

Упражнение для самостоятельной работы 3. Нажмите кнопку Setting и просмотрите, доступ к каким настройкам разрешает программа wbStego4.

Если вы передумали, хотите «откатить назад» и выйти из программы, для этого нажмите Exit. Переход к следующему этапу – кнопка Continue.

Второй этап. На втором этапе делаем выбор: отправлять данные (внедрять – encode) или получать (извлекать – decode) (рисунок 7).

Для извлечения данных из файла-носителя выставляем флажок у команды Decode. Если требуется внедрить туда данные, то выставляем флажок у команды Encode. Выбираем Encode и нажимаем кнопку Continue.

Третий этап. Программа просит указать, где расположены ваши конфиденциальные данные.

На этом этапе вы должны выбрать и указать полный путь в файловой системе для файла с вашими конфиденциальными данными, которые вы хотите скрыть. На рисунке 8 указан файл secret.txt.

Четвёртый этап. Программа просит указать файл-носитель (рисунок 9). Во-первых, из списка File Type, приведенного в диалоговом окне, необходимо указать тип файла-носителя (у нас выделен тип *.bmp), если у пользователя возникли затруднения, можно по-

пытаться активировать поиск файла-носителя с помощью самой программы, нажимая Find carrier file.

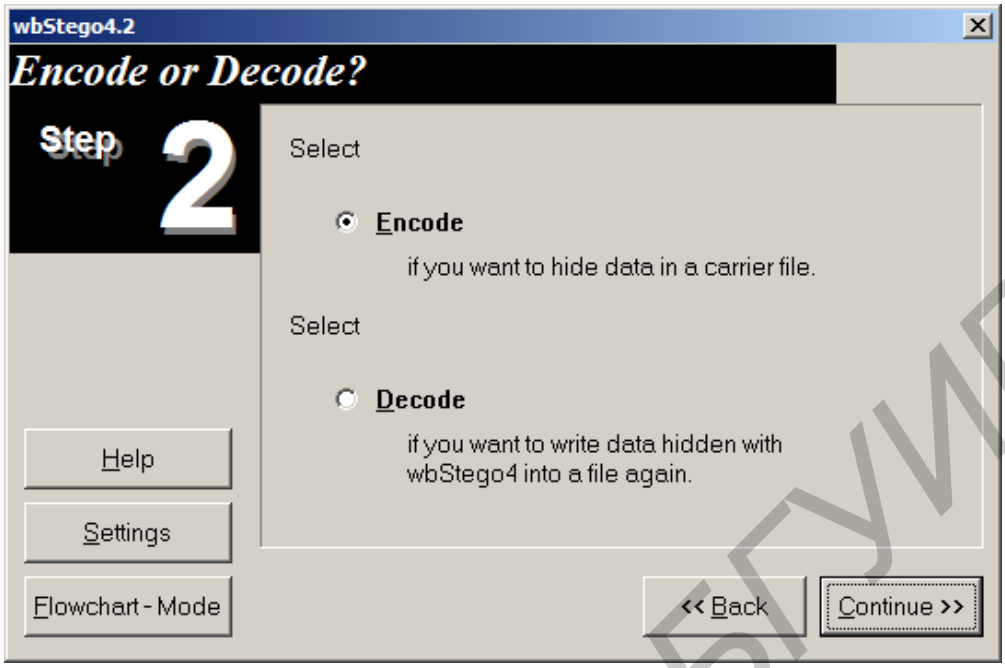


Рисунок 7 – Выбор режима работы программы wbStego4

Рисунок 8 – Указание пути к файлу с конфиденциальными данными

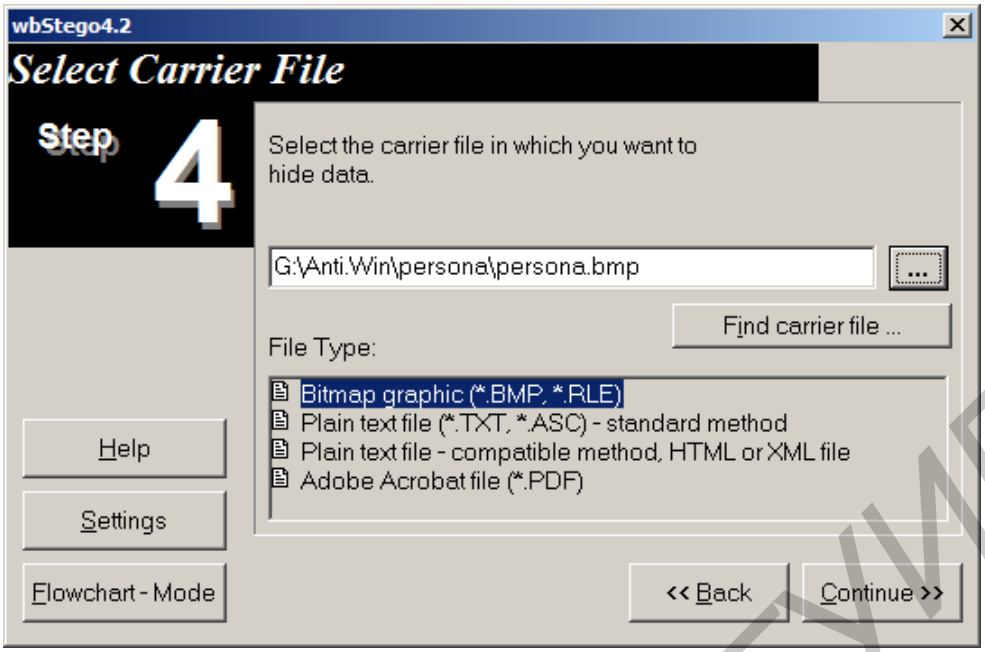


Рисунок 9 – Указание пути к файлу-носителю

Во-вторых, необходимо учесть, что размеры файла-носителя и данных, которые вы хотели бы туда внедрить, коррелируют. Файл-носитель, конечно, «рыхлый», но не «резиновый» и естественное ограничение – его собственный размер. Данные, превосходящие по размеру файл-носитель, внедрить туда никак нельзя. Речь может идти только о какой-то части, о какой-то доле размера файла-носителя, обычно, не превышающей 50 %. Точных рецептов и точных аналитических формул для расчёта этого параметра нет. Общее правило такое: для сокрытия большого массива конфиденциальных данных нужен соответствующего большего размера файл-носитель.

В-третьих, обсуждаемый параметр зависит от типа данных: наиболее «рыхлые» – чёрно-белые картинки формата *.bmp, минимум избыточных данных в формате *.pdf.

В-четвёртых, проверку обсуждаемого параметра осуществляет сама программа и вы можете увидеть сообщение, показанное на рисунке 10.

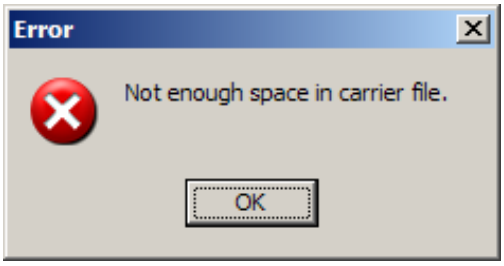


Рисунок 10 – Предупреждающее сообщение программы wbStego4

Сообщение означает, что программа wbStego4 сама произвела проверку и определила, что ваши данные предназначенные для сокрытия, не поместятся в файл-носитель. Для исправления необходимо вернуться в поле Select the carrier file, нажать кнопку [...] и выбрать файл-носитель подходящего размера. Можно также вернуться к шагу 3, нажимая кнопку Back (см. рисунок 9), и подобрать конфиденциальные данные меньшего размера.

Пятый этап. Следует заметить, что метод стеганографии не предполагает никакого кодирования данных, и то, что мы видим на рисунке 11 – дополнительная функция пакета wbStego4, реализованная разработчиком-программистом.

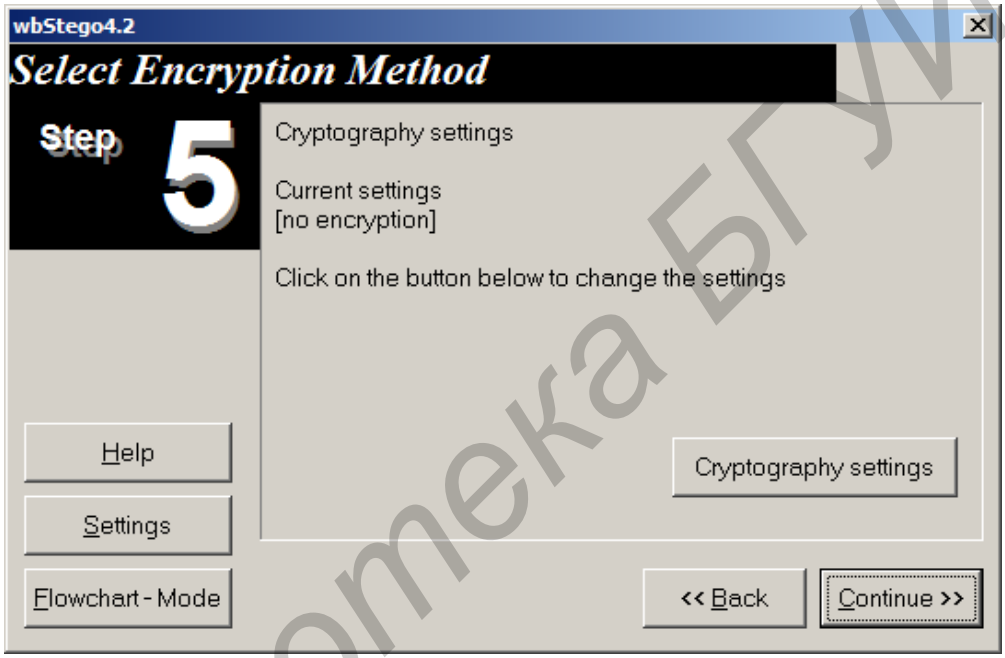


Рисунок 11 – Установка криптографических параметров программы wbStego4

Нажав кнопку Cryptography setting, переходим к следующему окну, выбрав криптографический алгоритм (рисунок 12). В этом окне мы видим предлагаемые алгоритмы шифрования.

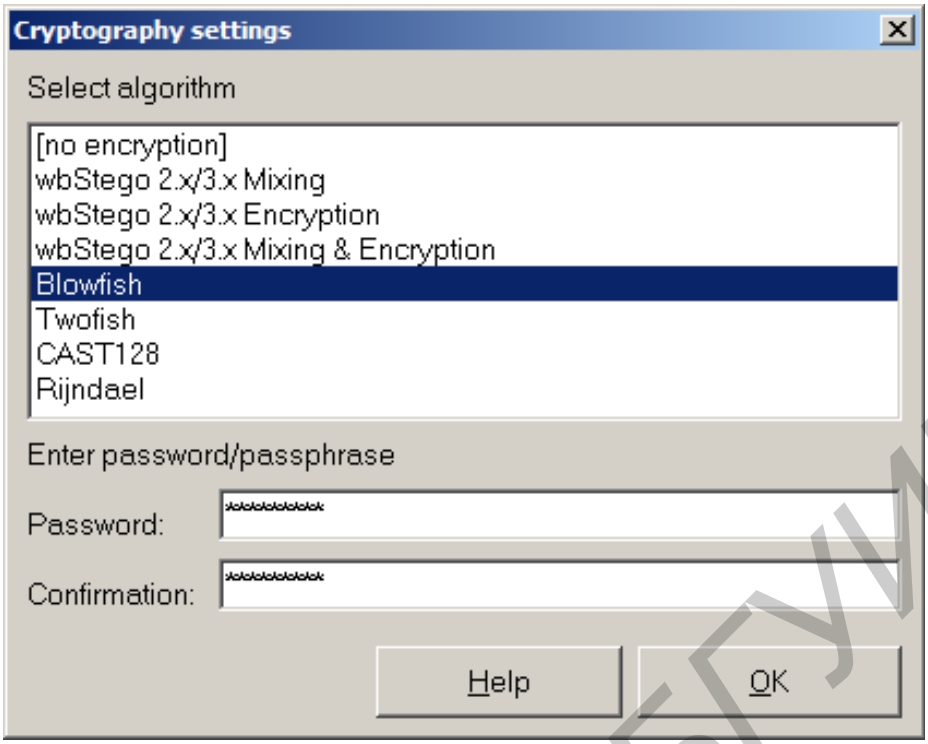


Рисунок 12 – Выбор криптографического алгоритма в программе wbStego4

Можно отказаться от шифрования, выбрав [no encryption], а можно выбрать один из предложенных стандартных алгоритмов: Blowfish, Twofish, Cast128, Rijndael. Это популярные традиционные алгоритмы симметричного шифрования, упомянутые в данном пособии в пункте 1.6.3 и разработанные математиками двенадцати стран. При выборе шифрования необходимо дважды ввести пароль (см. рисунок 12).

Шестой этап. Здесь возможны некоторые манипуляции с файлом-носителем. Мы выбрали persona.bmp (см. рисунки 9; 16, а). После этого программа модифицирует файл-носитель, внедрив туда наши данные и изменяя его (при необходимости сохраняя зарезервированную копию файла). Чтобы исходный файл не изменять, файл-носитель после внедрения туда данных лучше переименовать. Именно это продемонстрировано на рисунке 13.

Мы переименовали и указали полный путь в файловой системе для файла-носителя, куда программа внедрит ваши конфиденциальные данные. Переименованный файл-носитель – personaHide.bmp (рисунок 16, б).

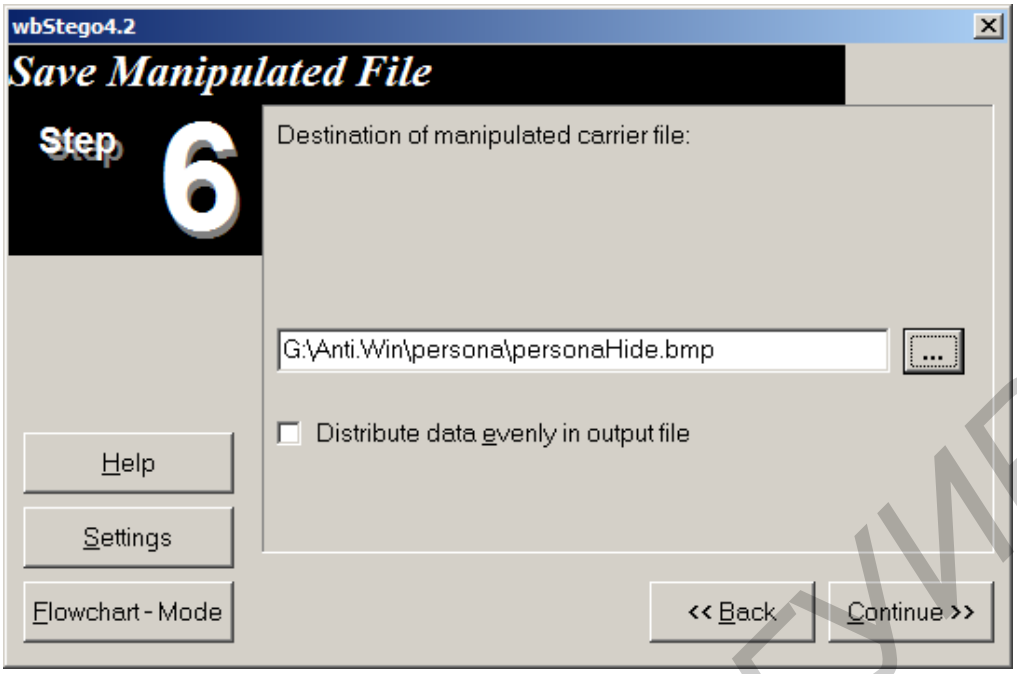


Рисунок 13 – Указание пути к модифицированному файлу-носителю с внедрёнными в него данными

На этом этапе программа предоставляет ещё одну возможность – распределять внедрённые данные в выходной файл равномерно (Distribute data evenly in output file). Если вы уверены в структуре распределения данных этого файла, отметьте флажок, но лучше положиться на программу, она разберется, как распределить внедрённые данные в файле-носителе. Для продолжения нажмите кнопку Continue.

Седьмой этап. На этом итоговом этапе программа завершила подготовительную работу: собрала все сведения и приготовила все данные. Итоговые данные выводятся пользователю в специальном диалоговом окне (рисунок 14). Окно содержит пользовательские установки (current setting):

- файл-носитель (carrier file) persona.bmp;
- шифруемые данные (encode data) secret.txt;
- выходной файл (manipulated file) personaHide.bmp;
- метод кодирования (encryption method) Blowfish.

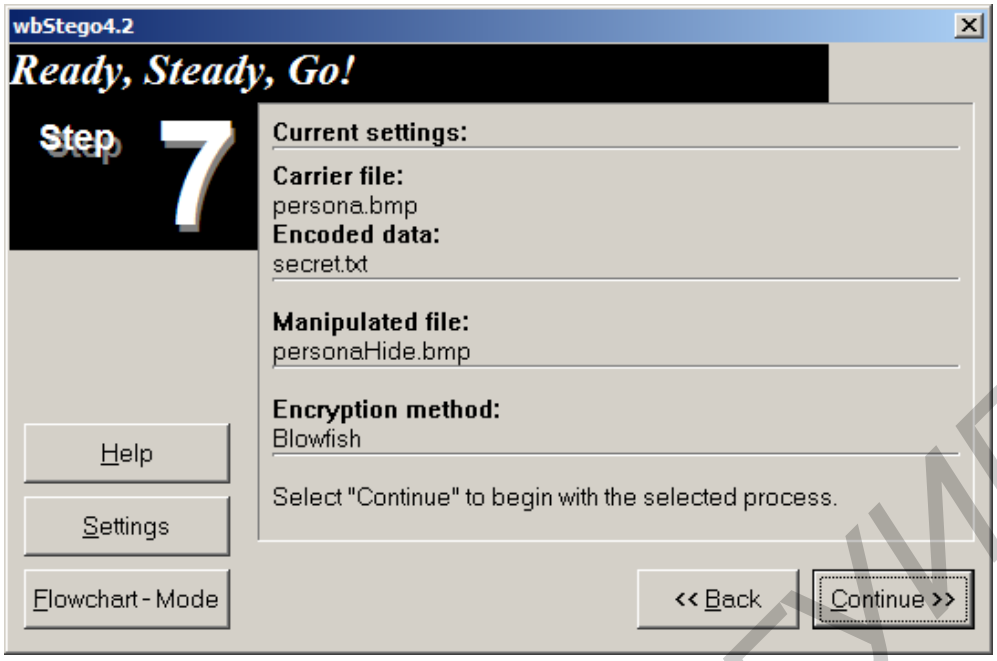


Рисунок 14 – Завершение этапа работы программы wbStego4 и краткий отчёт о завершённом кодировании

При необходимости внесения изменений используется кнопка Back для «отката» на предыдущий этап. Для продолжения работы нажмите кнопку Continue.

После нажатия Continue программа обработает все данные и в случае успешного завершения процесса подтвердит это сообщением, представленным на рисунке 15. На этом этапе работа с программой завершена. Программа информирует, что процесс шифрования данных и их внедрение в файл-носитель успешно завершён.

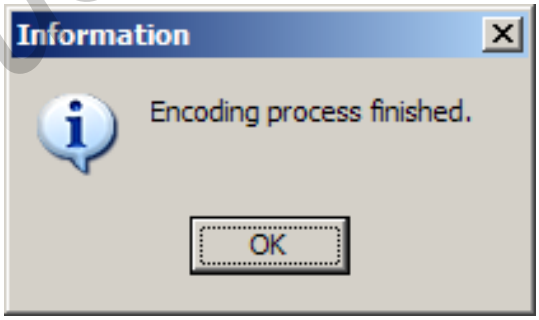
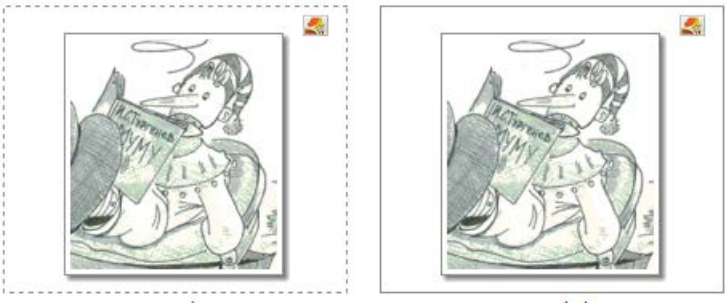


Рисунок 15 – Завершающее сообщение программы wbStego4

Восьмой этап. На этом этапе проводится мониторинг файла-носителя и его визуальный контроль.

Парадигма метода стеганографии заключается в скрытии факта внедрения данных в файл-носитель. На рисунке 16 приведены два

изображения файла-носителя: а – до внедрения данных; б – после внедрения. Отличия вы не найдёте, более того, размеры у этих файлов одинаковы.



а б

а – пустой файл-носитель (persona.bmp)
б – файл-носитель с внедрёнными данными (personaHide.bmp)
Рисунок 16 – Фрагмент окна
программы-просмотрщика ACDSee с файлом-носителем

2.1.2 Извлечение данных из файла-носителя программой wbStego4

Для решения обратной задачи – извлечения данных из файла-носителя – последовательно выполняем несколько этапов работы.

Первый этап. Запускаем программу (см. рисунок 6) и нажимаем кнопку Continue.

Второй этап. Для извлечения данных из файла-носителя выставляем флажок у команды Decode (рисунок 17) и нажимаем кнопку Continue.

Третий этап. На этом этапе, воспользовавшись кнопкой  , в поле ввода данных указываем полный путь к файлу-носителю (рисунок 18) и из списка File Type выбираем тип файла-носителя. Нажимаем кнопку Continue.

Четвёртый этап. На этом этапе программа просит ввести пароль (рисунок 19). Выполняем ввод пароля и нажимаем кнопку Continue.

Пятый этап. На этом этапе программа предлагает присвоить имя файлу конфиденциальных данных, которые она извлечёт из файла-носителя (рисунок 20). Рекомендуется дать файлу данных другое имя, отличающееся от первоначального. Нажимаем кнопку Continue.

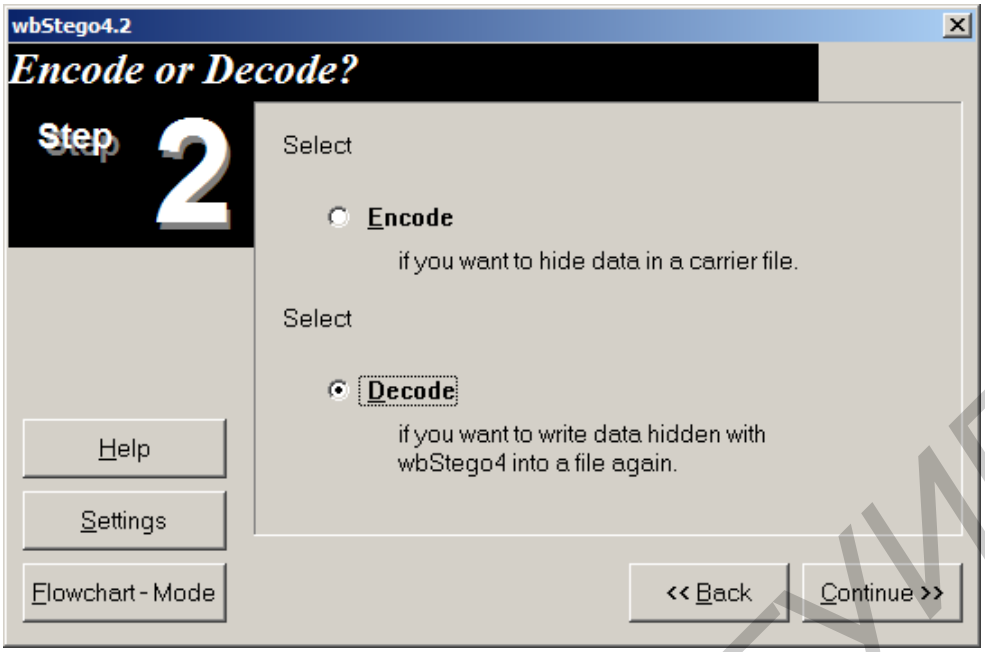


Рисунок 17 – Выбор режима работы программы wbStego4 для извлечения данных

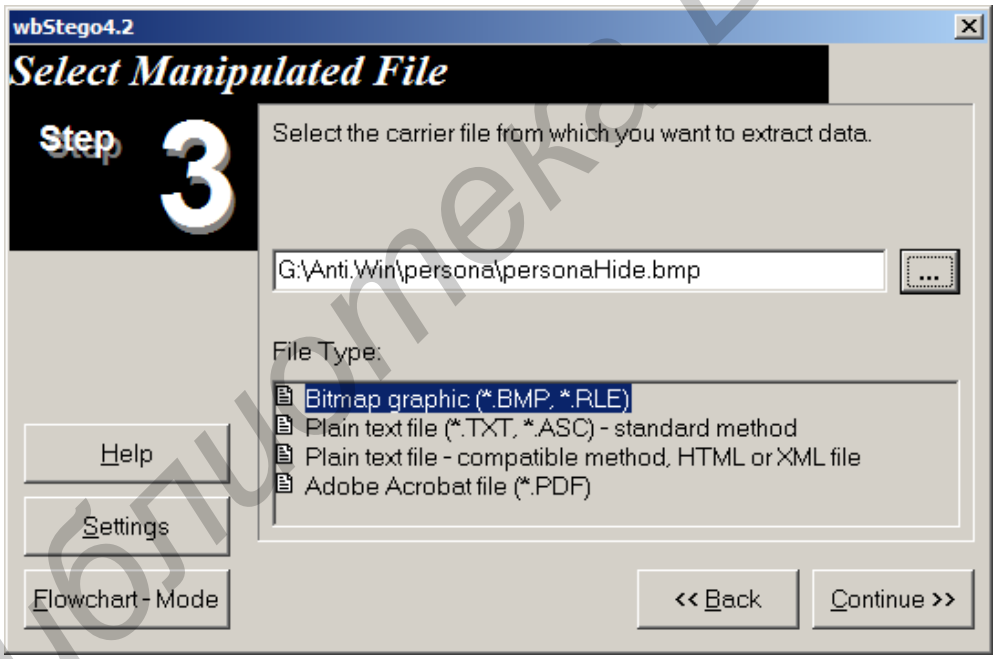


Рисунок 18 – Указание пути к файлу-носителю

Рисунок 19 – Указание пароля для начала процесса декодирования

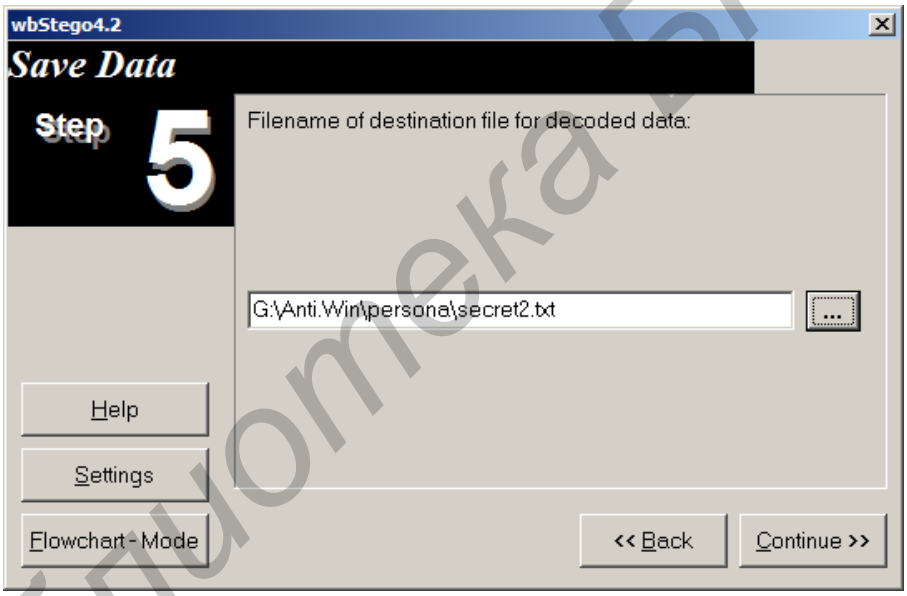


Рисунок 20 – Указание пути к файлу с конфиденциальными данными

Шестой этап. Программа завершила подготовительный этап, собрала все сведения и приготовила все данные. Эти итоговые данные она выводит пользователю в специальном диалоговом окне (рисунок 21). Окно содержит пользовательские установки (current setting):

– выходной файл (manipulated file) personaHide.bmp;
– извлечённый файл (result) secret2.txt;
– метод кодирования (encryption method) automatic decoding.

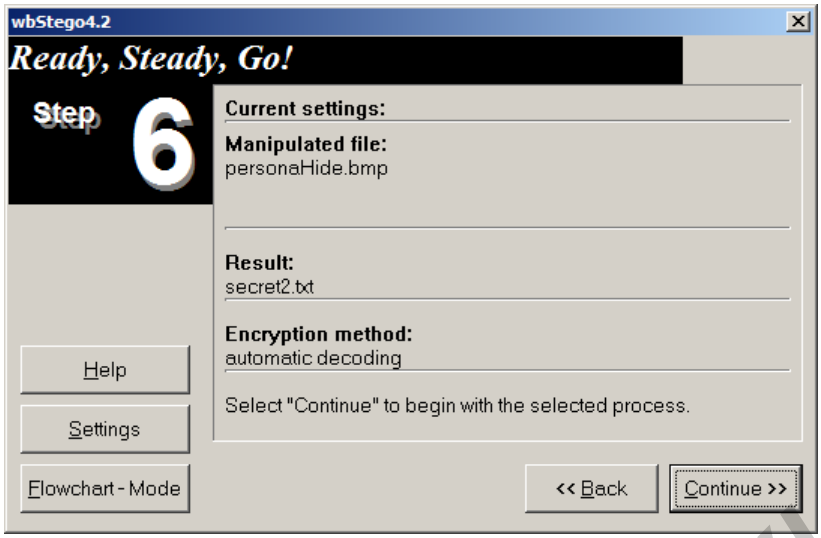


Рисунок 21 – Завершение этапа работы программы wbStego4 и краткий отчёт о завершённом декодировании

Программа автоматически определяет метод шифрования (automatic decoding) и пользователь избавлен от необходимости заполнять это вручную. Обычно пользователь-получатель не знает метода шифрования, который употребил пользователь-отправитель. Ему достаточно знать пароль.

Нажав Continue, завершаем работу программы и получаем завершающее сообщение (рисунок 22): Decoding process finished (процесс декодирования завершён).

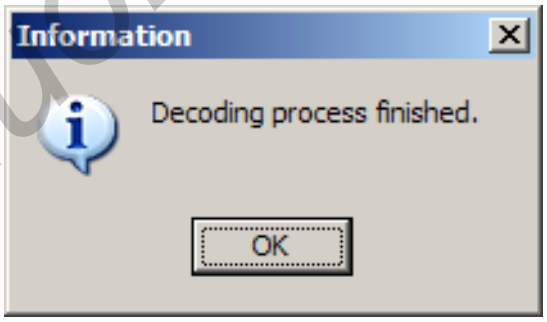


Рисунок 22 – Завершающее сообщение программы wbStego4 об успешном извлечении данных и декодировании

Упражнение для самостоятельной работы 4. Сравните содержимое файлов secret.txt и secret2.txt. В каком случае может возникнуть несовпадение?

2.2 Вычисление хэш-функции по алгоритму MD5 в Windows

2.2.1 Использование хэш-функции для контроля целостности данных

О важности и значимости функции хэширования достаточно подробно изложено в пункте 1.6.6 данного пособия. Здесь в практической части мы рассмотрим, как можно реализовать идею использования хэш-функции для контроля целостности ваших данных.

Для реализации этой идеи рассмотрим решение нескольких примеров с помощью несложной программы MD5summer, вычисляющей хэш-функцию по алгоритму MD5. Эта программа распространяется бесплатно и не требует инсталляции, поместив её в отдельный каталог, можно приступать к работе.

Допустим, пользователь Persona нашёл мультимедийные данные на одном из ftp-серверов о своих друзьях Coach и Actor. Он загрузил эти данные на свой компьютер и теперь озабочен вопросом, не произошёл ли какой-либо сбой при передаче по компьютерным сетям таких больших файлов. Этика предоставления ftp-сервиса клиентам требует, чтобы разместивший свои данные (файлы) на ftp-сервере снабжал их контрольными значениями хэш-функций. Это даёт возможность пользователям при загрузке файлов проверить их целостность. Продемонстрируем эту ситуацию на простейших примерах на локальном компьютере.

2.2.2 Создание файла с хэш-функцией на основе данных произвольной длины

После запуска MD5summer программа требует определить каталог с данными и тип действия (рисунок 23): создание суммы MD5 или её проверка.

Для начала мы создадим хэш-функцию. Нажав кнопку «Создать сумму», переходим к диалоговому окну (рисунок 24), в котором приведен список папок с песнями и кинофильмами, которые загрузил Persona. Данная программа может вычислить значение хэш-функции для произвольного количества файлов в отмеченном каталоге. Выберем сначала один, например persona.avi.

Выбрав файл persona.avi и нажав кнопку «Добавить», видим появление ссылки на этот файл в правом списке (рисунок 25).

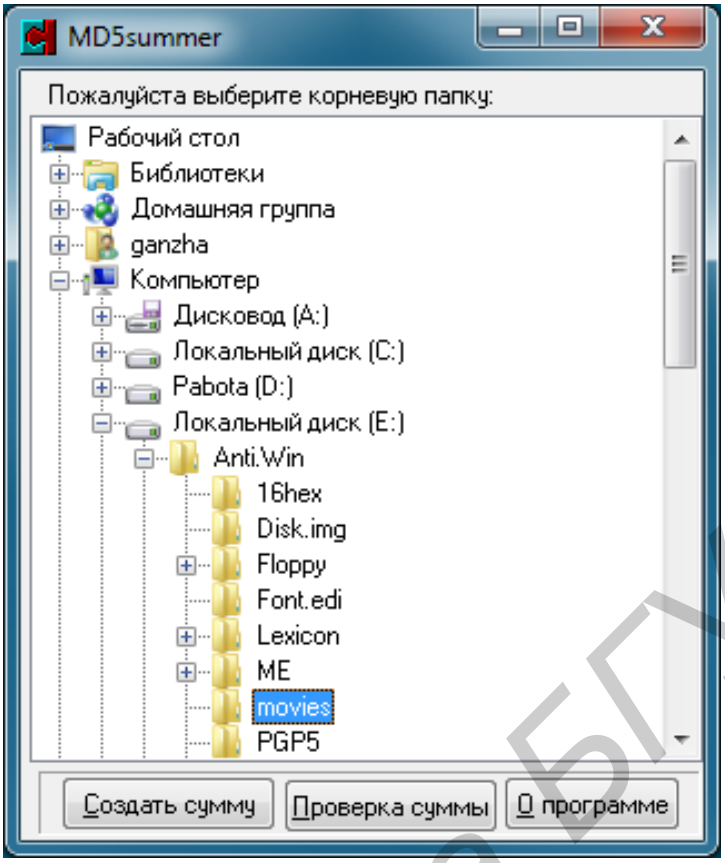


Рисунок 23 – Определение рабочего каталога программы MD5summer

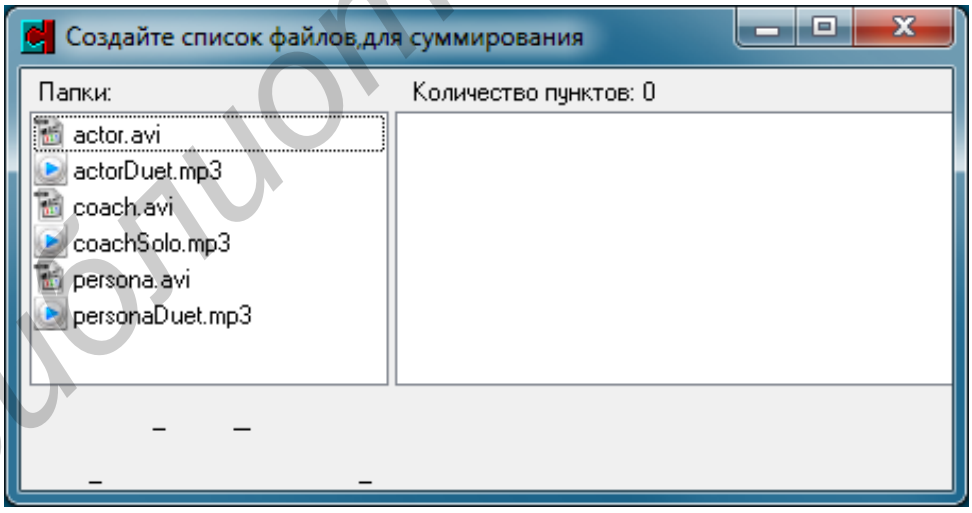


Рисунок 24 – Окно выбора данных программы MD5summer

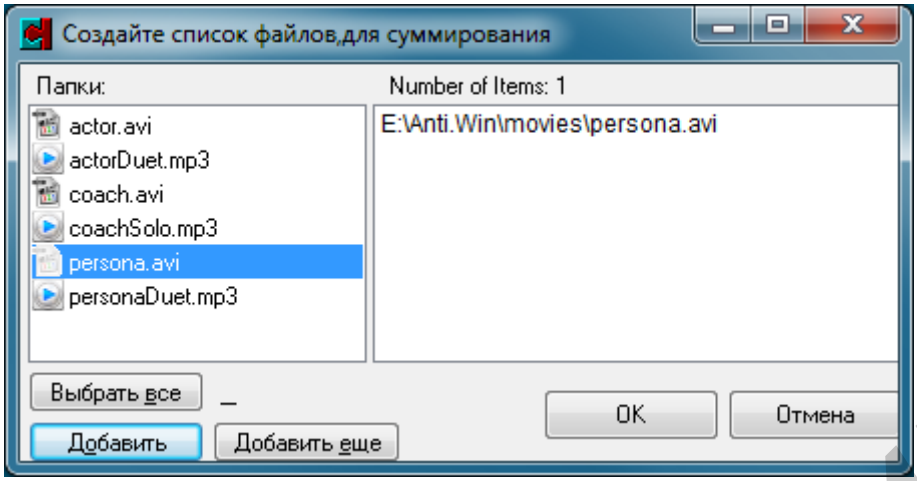


Рисунок 25 – Указание файлов с данными для вычисления хэш-суммы

Выделив этот файл в правом списке и подтвердив этот выбор кнопкой OK, открываем окно, представленное на рисунке 26. В окне представлена информация об имени файла (слева) и значение хэш-функции в виде 16-ричного 32-разрядного числа (справа). Один разряд такого числа состоит из четырёх бит, значит, полное значение хэш-функции составляет 32 × 4 = 128 бит, в полном соответствии с алгоритмом MD5. В левом нижнем углу указано время, которое программа затратила на вычисление хэш-функции. В данном случае на файл persona.avi (94 Мб) затрачена 1 с.

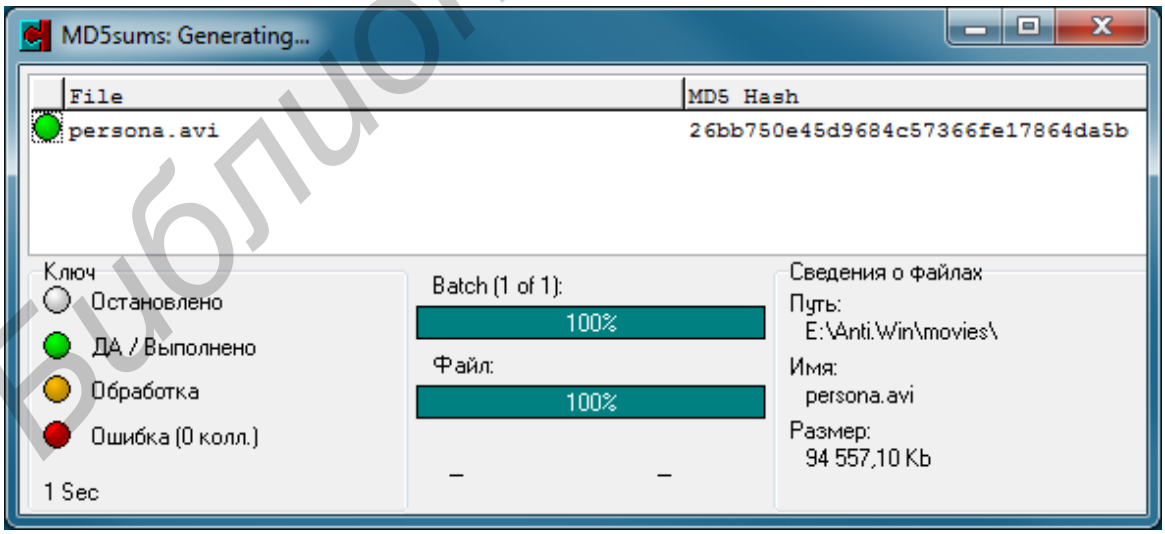


Рисунок 26 – Рабочее окно программы MD5summer

Упражнение для самостоятельной работы 5. Создайте для каждого мультимедийного файла в каталоге Anti.Win с помощью

программы MD5summer файл с хэш-суммой с расширением *.md5.

Упражнение для самостоятельной работы 6. Доступными вам средствами найдите 128-битовое значение хэш-функции в файле persona.md5. Проверьте, соответствует ли это значение приведенному на копии экрана ранее (см. рисунок 26). Если 128 бит перевести в байты, то получится 16 байт, почему тогда все размеры файлов (рисунок 27) с расширением *.md5 значительно превышают этот размер?

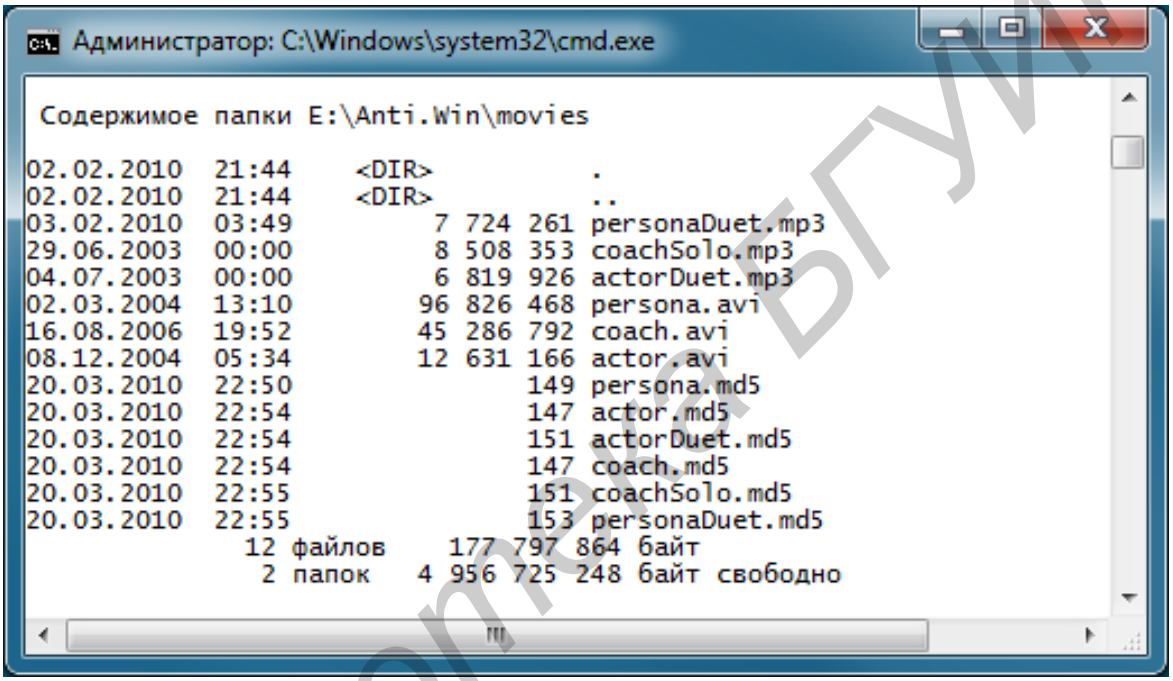


Рисунок 27 – Копия экрана со списком файлов рабочего каталога программы MD5summer

2.2.3 Проверка целостности данных программой MD5summer

Программа MD5summer позволяет проверить, не нарушена ли целостность данных в файлах, для которых предварительно была рассчитана хэш-сумма по алгоритму MD5. Убедимся в этом. Теперь, запустив программу и отыскав каталог с данными, нажимаем кнопку «Проверка суммы» (рисунок 28). В окне появится список всех файлов с контрольными суммами *.md5

Необходимо указать, какую сумму мы хотим проверить. Для данного примера – это persona.md5. Нажимаем кнопку «Открыть» и программа производит проверку. Получаем такой же экран, как на рисунке 26. Слева от имени файла появится зелёный флажок (кружок), что говорит об успешном завершении проверки.

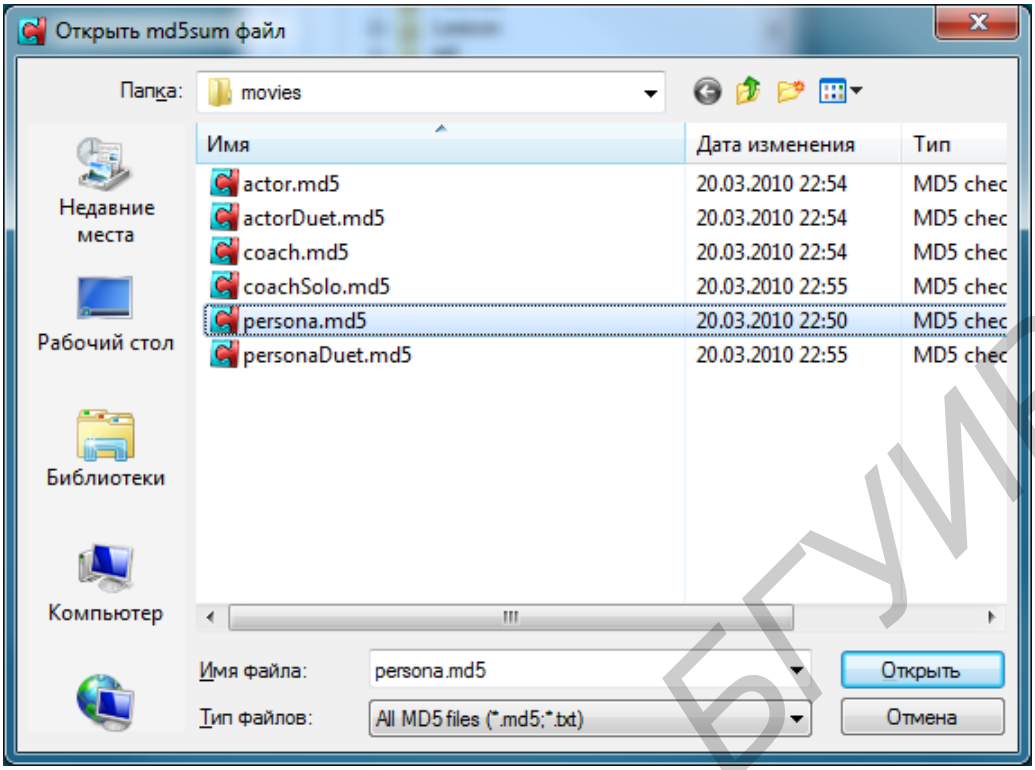


Рисунок 28 – Копия экрана со списком отфильтрованных файлов рабочего каталога программы MD5summer

2.2.4 Сбой проверки целостности данных

При передаче данных по сетям вполне вероятны сбои и ошибки. В этом случае хэш-функция, вычисленная до передачи данных и после, конечно же не совпадает. Как реагирует программа MD5summer в этом случае? Выясним на следующем примере.

Упражнение для самостоятельной работы 7. Доступными вам средствами измените один бит данных в файле personaDuet.mp3.

Запустив программу MD5summer и получив диалоговое окно как на рисунке 22, нажмите кнопку «Проверка суммы». Укажите программе, что вы хотите проверить файл personaDuet.md5. Программа произведёт вычисления и выведет сообщение (рисунок 29).

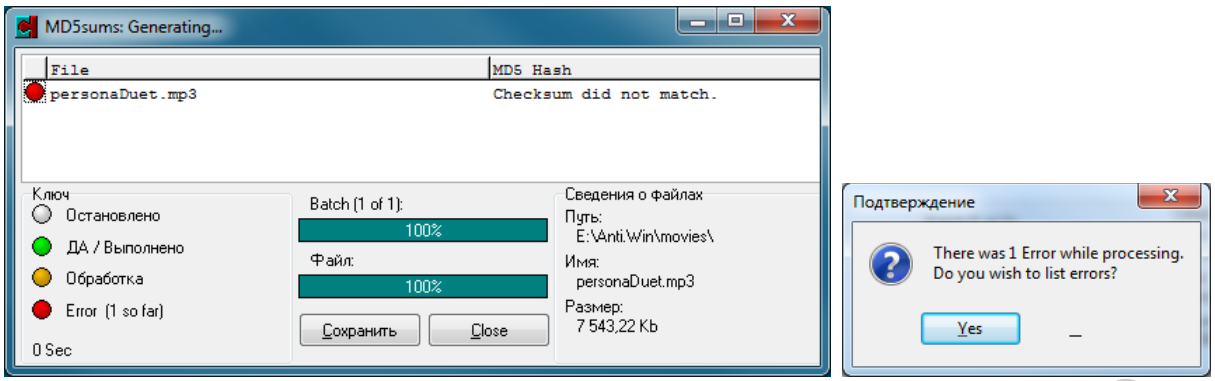


Рисунок 29 – Рабочее окно программы в процессе обнаружения несоответствия хэш-функции. Отчёт программы MD5summer об обнаружении несоответствия

В этом случае мы получим, во-первых, диалоговое окно с красным флажком (кружком) и с надписью «Checksum did not match» (Несоответствие контрольной суммы), а во-вторых, дополнительное сообщение «There was 1 Error while processing. Do you wish to list errors?» (В процессе обработки обнаружена 1 ошибка. Показать список ошибок?).

2.3 Вычисление хэш-функции в ОС Linux

В операционных системах Linux вычисление хэш-функции и её контроль осуществляется в командной строке так же просто как и в операционной системе Windows, использующей графический интерфейс. Исторически сложилось, что проверке данных в системах UNIX/Linux всегда уделялось значительное внимание. Здесь даже не надо устанавливать программу от сторонних производителей.

Работа с функцией MD5 встроена и поддерживается на уровне операционной системы. Это даёт возможность пользователю работать с совершенно простыми командами. Ниже приводится несколько примеров, преследующих те же цели, что мы рассмотрели в подразделе 2.2. для операционной системы Windows. В результате тоже получается 128-битный хэш-код постоянной длины.

Начнём с простого: изучим команду md5sum, освоим её ключи, научимся считать значение хэш-функции и выводить это значение в стандартный вывод – на экран. Это осуществляется следующим образом.

В консоли Linux в командной строке вводим команду

```
md5sum -b file_name.
```

Приведенная команда с ключом -b (--binary) посчитает хэш-функцию бинарного файла file_name и выведет её значение на экран. Это можно видеть на рисунке 30, где подсчитана контрольная сумма для файла ho.txt.

```
linux-8ks6:/home/duralei/Музыка # md5sum -b ho.txt
2ec1032b06ce41772abde4058d2d4569 *ho.txt
```

Рисунок 30 – Вывод программой значения хэш-функции

Если мы хотим сохранить значение контрольной суммы в файл для будущего применения, мы, используя стандартные средства Linux, перенаправляем вывод с экрана в файл и выполняем следующую команду:

md5sum -b file_name1 > file_name2.

Вместо file_name1 мы можем использовать регулярное выражение, описывающее группу файлов, и тогда в file_name2 мы получим список контрольных сумм для этой группы файлов. На рисунке 31 показано, как перенаправить результат работы программы md5sum в файл ho.txt.md5 .

```
linux-8ks6:/home/duralei/Музыка # md5sum -b ho.txt > ho.txt.md5
linux-8ks6:/home/duralei/Музыка #
```

Рисунок 31 – Перенаправление вывода программы. Экран пустой

Если случится, что мы забудем какие-то ключи команды, то мы набираем в командной строке md5sum --help и получаем краткую справку, либо, воспользовавшись стандартной man-справкой Linux, наберём в командной строке

man md5sum.

Покажем, как осуществляется проверка целостности данных в файле с использованием просчитанной контрольной суммы. Используем для этого уже полученную хэш-функцию, записанную в файл ho.txt.md5. Теперь если у нас возникли сомнения в целостности файла ho.txt, мы набираем в командной строке (рисунок 32)

md5sum --check ho.txt ho.txt.md5.

```
linux-8ks6:/home/duralei/Музыка # md5sum -c ho.txt ho.txt.md5
md5sum: ho.txt: no properly formatted MD5 checksum lines found
ho.txt: OK
```

Рисунок 32 – Успешное завершение проверки контрольной суммы

Как показано на рисунке 32, в случае совпадения контрольной суммы программа выводит на экран сообщение об успешном выполнении

ho.txt OK.

Если же целостность файла ho.txt нарушена, то результат выполнения этой команды будет совсем другим, что показано на рисунке 33.

```
linux-8ks6:/home/duralei/Музыка # md5sum -c ho.txt ho.txt.md5
md5sum: ho.txt: no properly formatted MD5 checksum lines found
ho.txt: FAILED
md5sum: WARNING: 1 computed checksum did NOT match
```

Рисунок 33 – Сбой верификации проверки контрольной суммы

Программа выводит сообщение о сбое верификации ho.txt: FAILED и предупреждает о несовпадении контрольной суммы md5sum: WARNING: 1 computed checksum did NOT match .

Эта программа для расчёта хэш-функций является стандартной утилитой в ОС Linux и поэтому поддерживает работу с регулярными выражениями. На рисунке 34 показан пример вычисления контрольных хэш-сумм для большого списка файлов фильтрующихся по маске, получающейся из регулярного выражения, записанного в одну строку.

```
linux-8ks6:/home/duralei/Музыка # md5sum --binary Gera00[0-9].[p,s]kr > gkr.md5
```

Рисунок 34 – Использование регулярного выражения в командной строке

Здесь приведена командная строка, где программе md5sum предписывается рассчитать хэш-функции для всех файлов по маске Gera00☺.☺kr, в имени которых на 7-м знакоместе может быть

любая цифра, а на 9-м – буква или *p* или *s*. После этого вывод самих контрольных сумм перенаправляется в файл gkr.md5 .

Эта команда выглядит следующим образом:

md5sum --binary Gera[0-9].[p,s]kr > gkr.md5.

В результате перенаправления вывода мы никаких сообщений о работе программы на экране не видим. Этим же регулярным выражением мы воспользуемся для проверки контрольных сумм (рисунок 35).

```
linux-8ks6:/home/duralei/Музыка # md5sum --check Gera00[0-9].[p,s]kr gkr.md5
md5sum: Gera001.pkr: no properly formatted MD5 checksum lines found
md5sum: Gera001.skr: no properly formatted MD5 checksum lines found
md5sum: Gera002.pkr: no properly formatted MD5 checksum lines found
md5sum: Gera002.skr: no properly formatted MD5 checksum lines found
md5sum: Gera003.pkr: no properly formatted MD5 checksum lines found
md5sum: Gera003.skr: no properly formatted MD5 checksum lines found
md5sum: Gera004.pkr: no properly formatted MD5 checksum lines found
md5sum: Gera004.skr: no properly formatted MD5 checksum lines found
Gera001.pkr: OK
Gera001.skr: OK
Gera002.pkr: OK
Gera002.skr: OK
Gera003.pkr: OK
Gera003.skr: OK
Gera004.pkr: OK
Gera004.skr: OK
```

Рисунок 35 – Результат выполнения команды, содержащей в качестве параметров регулярное выражение

По выведенному на экран сообщению мы видим, что по маске определяемой регулярным выражением в текущем каталоге найдено восемь файлов. Все восемь файлов программа приводит списком на экране и докладывает, что целостность их не нарушена.

Упражнение для самостоятельной работы 8. Используя описанную выше команду md5sum, посчитайте хэш-функции для файлов в вашем домашнем каталоге ~/bin и результат запишите в файл CR01.md5.

Упражнение для самостоятельной работы 9. Используя описанную выше команду md5sum, проверьте правильность контрольной суммы для файлов из вашего домашнего каталога ~/bin, используя файл CR01.md5, который был создан в предыдущем упражнении.

Упражнение для самостоятельной работы 10. Попытайтесь нарушить целостность одного файла из вашего каталога ~/bin. Используйте для этого один из доступных редакторов vim или nano Используя команду md5sum, убедитесь, что произошёл сбой верификации хэш-функции для этого файла.

2.4 Практическая работа с пакетом PGP

В данном пособии приводится несколько примеров использования пакета PGP (Pretty Good Privacy – вполне надёжная секретность). Для иллюстрации различных методов шифрования используется пакет PGP 5.0, управляемый текстовым интерфейсом из командной строки. Выбор этой версии пакета объясняется не консерватизмом авторов (в настоящее время существуют и продаются коммерческие версии PGP 8.0-9.0), а тем, что последние версии, используя графический интерфейс, прячут и скрывают «кухню» работы алгоритмов шифрования. Быть может, это удобно для «конечного пользователя», работающего с электронной почтой и шифрующего поток своих конфиденциальных данных, но совершенно недопустимо для будущих программистов, обучающихся навыкам защиты информации.

Пакет PGP 5.0 в 32-разрядной среде Windows может работать только в текстовой консоли виртуальной машины MS DOS. Методическая цель упражнений с этим пакетом – во-первых, напомнить студентам работу с текстовым интерфейсом, что совершенно необходимо будущим программистам, поскольку работа с конфигурационными файлами, с различными тонкими настройками системы чаще всего предполагает работу с текстовым интерфейсом командной строки. Во-вторых, только так можно увидеть и понять все детали работы пакета PGP и различные аспекты криптографической защиты.

Обучаемые «вживую» могут увидеть и воспользоваться функциями симметричного шифрования, закодировав файлы, предлагаемые им в методическом задании, используя для этого алгоритм симметричного шифрования IDEA. Пакет PGP обеспечивает также все функции асимметричного шифрования, которыми могут воспользоваться обучаемые:

– создание пары ключей для организации шифрованной переписки с корреспондентом (для генерации случайной последовательности при создании ключей используются индивидуальные биометрические особенности пользователя);

– передача сообщений, зашифрованных публичным ключом по алгоритму RSA, по открытым каналам связи, в том числе по электронной почте;
– передача секретного ключа симметричного шифрования по алгоритму DSS/Diffie-Hellman по открытым каналам связи, включая электронную почту;
– приём зашифрованных сообщений и их раскодирование private-ключом;
– создание с помощью private-ключа аутентифицированного сообщения и цифровой подписи между двумя корреспондентами;
– организация арбитражной цифровой подписи в случае отсутствия доверия у корреспондентов.

Вот неполный перечень возможностей этого небольшого пакета PGP, который может быть успешно использован.

Пакет PGP5 работает только в шестнадцатиразрядном окне (консоли), в формате записи файлов 8.3 (восемь_точка_три). Терминал (консоль текстовой строки) для работы с интерфейсом текстовой строки и ввода команд в операционных системах Microsoft можно открыть сочетанием клавиш [⊞+R]. После чего в появившемся текстовом поле набираем cmd.

Используя команды dir и cd, перейдите в каталог, где у вас установлен пакет PGP и сделайте его текущим.

Для выполнения всех упражнений в этой части практических работ предполагается, во-первых, присутствие у студентов самых элементарных понятий и навыков работы в операционных системах Linux, во-вторых, доступ к самой ОС Linux. Это может быть удалённый консольный доступ к серверу, например, по протоколу SSH, либо Linux функционирует в виртуальной машине, например, vmware, либо, что очень хорошо, обучаемый имеет монопольный доступ к ОС Linux, установленной у него на desktop или notebook.

2.4.1 Шифрование данных симметричным ключом. Алгоритм IDEA

Создание и декодирование зашифрованного файла необходимо выполнить до создания пары «открытый/закрытый ключ», иначе приложение будет «конфликтовать» с вами, пытаясь использовать имеющийся ключ в текущем каталоге.

Создание зашифрованного файла

В окне терминала введите команду

pgpe -c file_name.

После ввода команды программа спросит пароль, который потребуется ввести два раза, после чего происходит шифрование по стандартному алгоритму IDEA и параллельно архивация (рисунок 36). Следует иметь в виду, что при вводе пароля приложение не показывает ни «звёздочек», ни «пробелов», и курсор остаётся на месте.

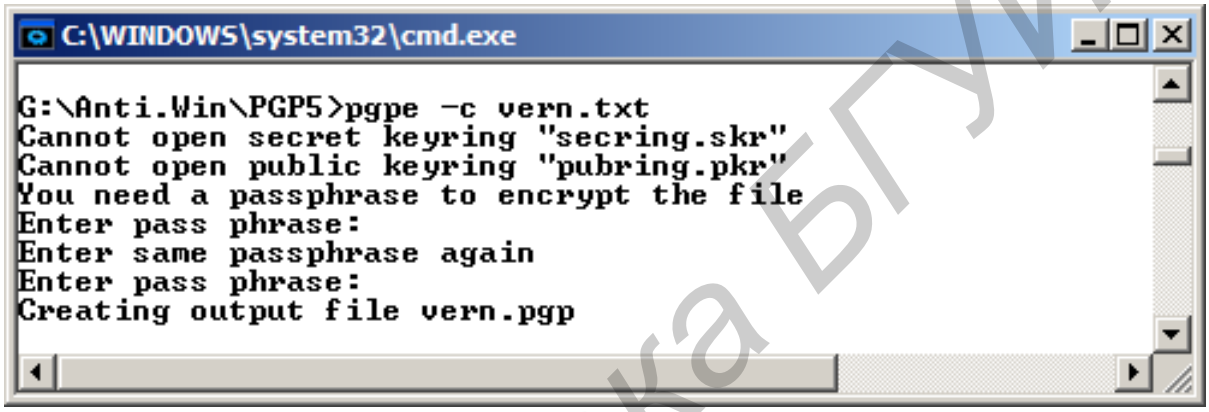


Рисунок 36 – Сообщения, выводимые на терминал программой PGP
в процессе шифрования файла

Отметим, что все команды программы, разбираемые здесь, имеют полезный ключ «–o», позволяющий присвоить выходному файлу любое пользовательское имя, например:

pgpe -c file_name –o output_file_name.

В противном случае по умолчанию пакет к имеющемуся имени файла добавляет своё расширение *.pgp, как мы видим на копии экрана.

Декодирование зашифрованного файла

В окне терминала выполняем команду (рисунок 37):

pgpv file_name –o output_file_name.

Для того чтобы сохранить существующий файл vern.txt, имя выходного файла изменено на vern1.txt. Естественно, что программа

осуществит декодирование только в том случае, если введен соответствующий правильный пароль.

Упражнение для самостоятельной работы 8. Сравните размеры и содержимое файлов vern.txt и vern1.txt и сделайте соответствующие выводы.

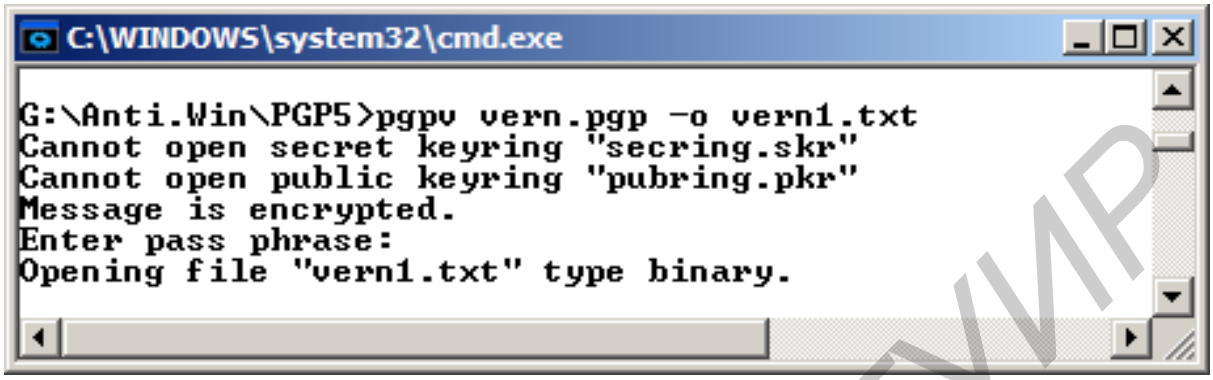


Рисунок 37 – Сообщения, выводимые на терминал программой PGP при декодировании зашифрованного файла

2.4.2 Создание пары ключей для осуществления метода асимметричного шифрования

Для создания пары ключей в окно терминала вводится команда

pgpk –g.

После этого программа в интерактивном режиме вступает в диалог с пользователем и требует от него правильных ответов и дополнительных действий (рисунок 38).

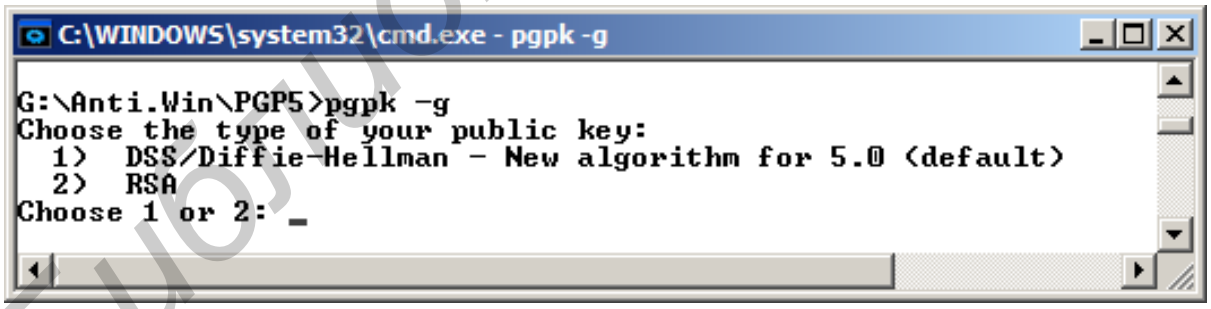


Рисунок 38 – Запрос программы алгоритма кодирования

Требуется выбрать алгоритм асимметричного шифрования. Программа предлагает нам два варианта: алгоритм DSS/Диффи-Хеллмана и – RSA. Алгоритм Диффи-Хеллмана используется только для кодированной передачи ключа сессии симметричного шифрования и не пригоден для кодирования обычных сообщений, поэтому выбираем RSA, для чего набираем цифру 2 (рисунок 39).

Теперь программа спрашивает, какой длины ключ шифрования мы хотели бы выбрать? От длины ключа зависит криптостойкость зашифрованных данных, но, с другой стороны, шифрование с длинными ключами происходит медленнее. Для мощного компьютера выбираем третий вариант (рисунок 40).

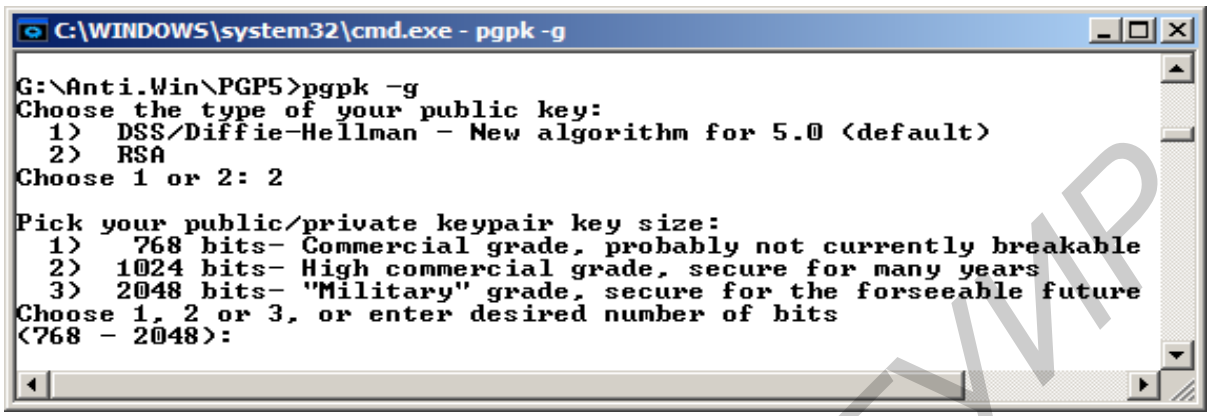


Рисунок 39 – Запрос размера ключа

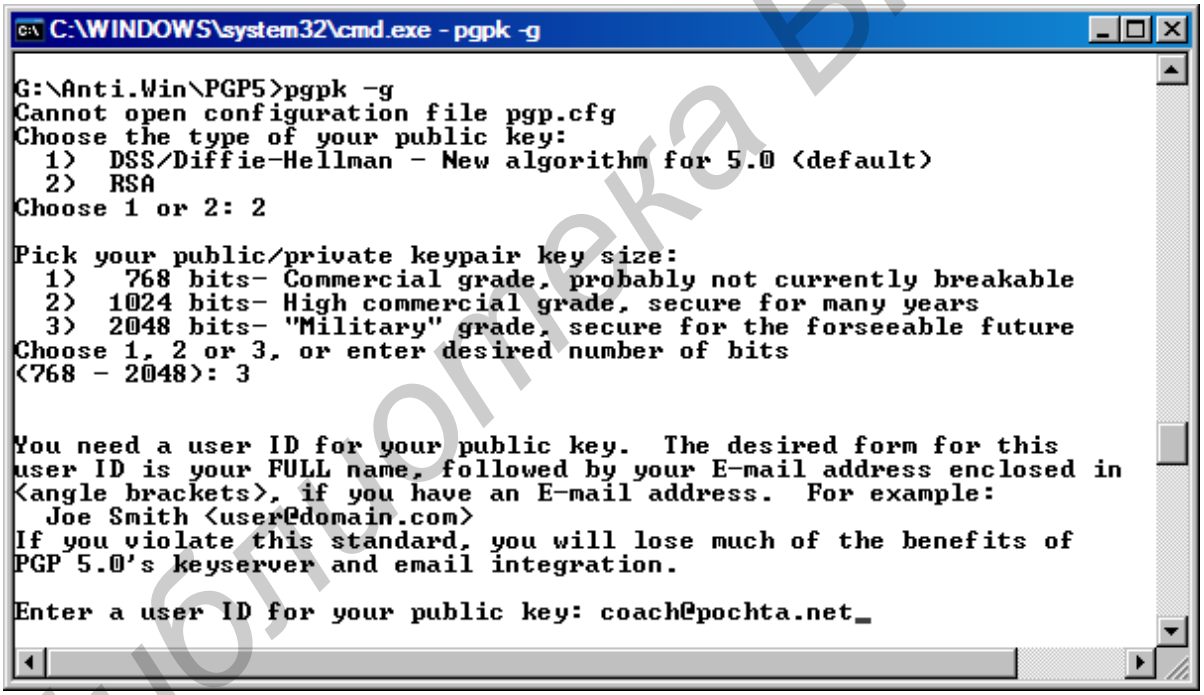


Рисунок 40 – Запрос открытого идентификатора пользователя

Далее программа спрашивает идентификатор пользователя, создающего ключи. Нет особых ограничений на длину и синтаксис этого идентификатора – открытые данные, которые впоследствии могут использовать ваши корреспонденты по переписке. Чаще всего это адрес электронной почты пользователя, как показано на копии экрана.

Следующий шаг – программа спрашивает: «Сколько дней вы собираетесь пользоваться вашими ключами?». Принимаем решение – год (365 дней) (рисунок 41).

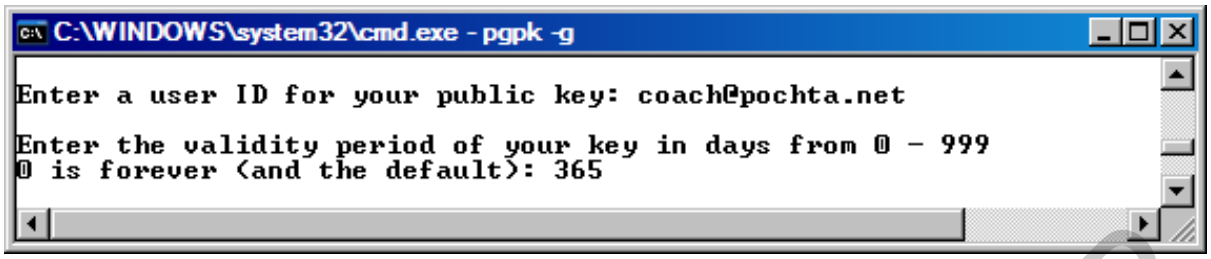


Рисунок 41 – Запрос срока действия пары создаваемых ключей

Дальше программа просит ввести пароль для private-ключа. Эти данные являются закрытыми, и никто, кроме вас, ключ этот знать не должен (рисунок 42).

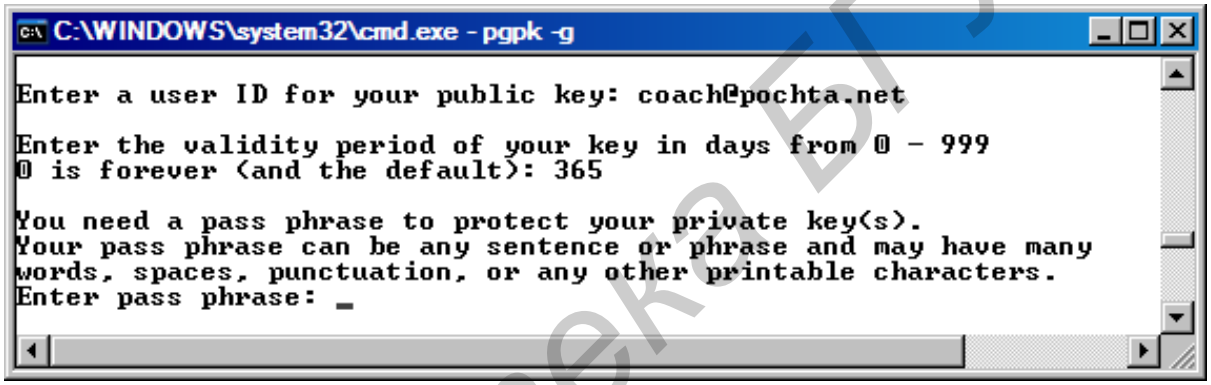


Рисунок 42 – Запрос на ввод закрытого пароля для private-ключа

Пароль вводится два раза.

Для создания ключей программа просит выступить вас в роли генератора случайных чисел: путём случайного и беспорядочного постукивания по клавиатуре вы создаёте для программы набор случайных временных интервалов, создавая тем самым случайную последовательность. Выполните это (рисунок 43).

После информации об успешном создании ключей программа вас с этим поздравит, выйдет из интерактивного режима, и вы вновь окажетесь в режиме командной строки (рисунок 44).

Упражнение для самостоятельной работы 11. Проверьте доступными вам средствами, появились ли в текущем каталоге файлы пары ключей pubring.pkr и secring.skr?

Упражнение для самостоятельной работы 12. Как вы думаете, зачем программа создала в текущем каталоге ещё пару файлов –

pubring.bak и secring.bak – причём нулевой длины? В каком случае эти файлы будут иметь нулевую длину? Сделайте выводы.

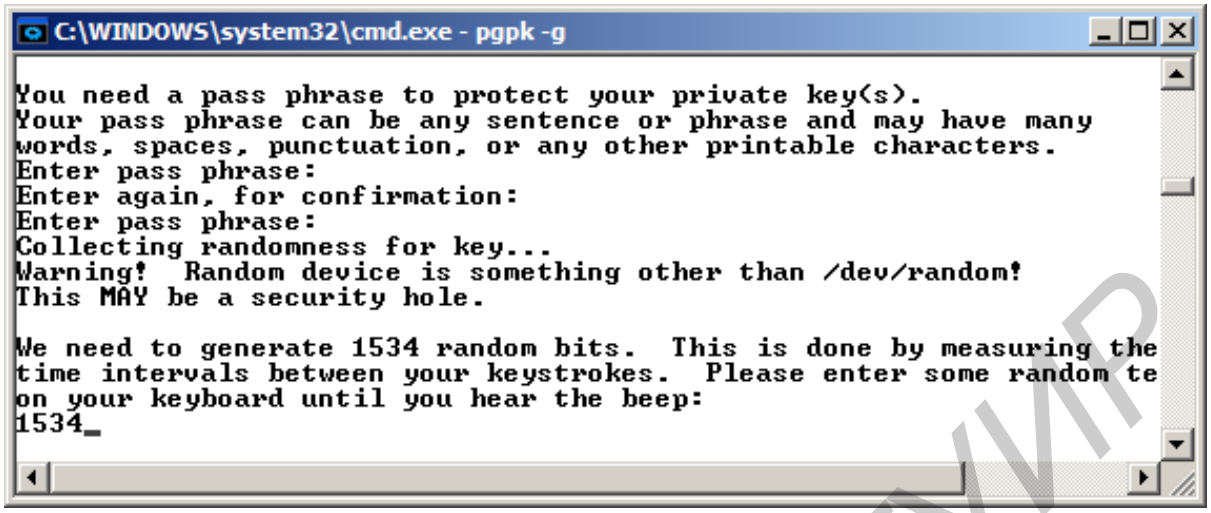


Рисунок 43 – Запрос активных действий пользователя для создания случайной последовательности

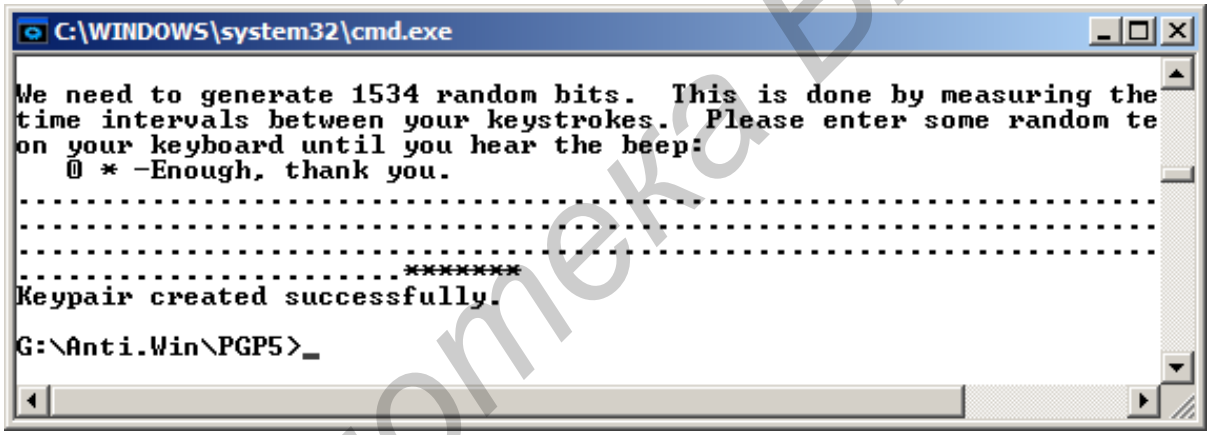


Рисунок 44 – Выведенное на терминал сообщение программы PGP об успешном создании пары ключей

2.4.3 Создание зашифрованного сообщения с помощью открытого ключа

Теперь всё готово для кодирования сообщений методом асимметричного шифрования по алгоритму RSA. Закодируем файл vern.txt. Для этого в командной строке набираем (рисунок 45)

pgpe –r ID file_name –o output_file_name.

Все опции в командной строке вам уже знакомы за исключением ID – это тот самый открытый идентификатор вашего корреспондента, например, адрес его электронной почты. Свой публичный ключ pubring.pkr он уже выложил на почтовый сервер или прислал

вам непосредственно и вы его заблаговременно положили в текущий каталог.

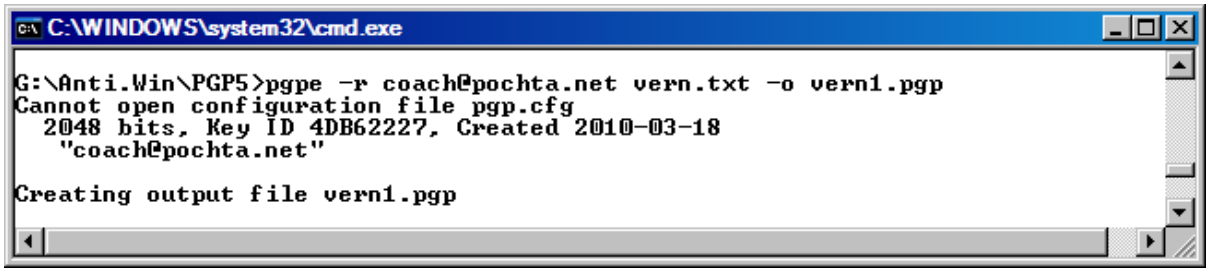


Рисунок 45 – Сообщение программы PGP об успешном кодировании данных

Зашифрованный файл vern1.pgp создан, можно его посылать своему корреспонденту по открытым сетям, пусть хакеры «поломают голову» над его криптоанализом. Только ваш корреспондент, имея private-ключ secring.skr, может декодировать и прочитать сообщение.

Здесь следует отметить ещё одну полезную опцию программы. На очередной копии экрана показана опция –a. Во-первых, это позволяет провести шифрование, используя только первую половину кодовой таблицы ASCII, то есть используя не весь байт (восемь бит), а только младшие разряды байта – 7-битовую кодировку. Шифрованное сообщение будет состоять только из печатаемых символов. За такое «удобство» приходится расплачиваться более длинным (на 20–30 %) зашифрованным сообщением. Во-вторых, вы обезопасите себя, если ваше зашифрованное сообщение vern1a.pgp «забредёт» в старые компьютерные сети, поддерживающие только 7 младших бит и усекающие старший 8-й бит, то целостность сообщения нарушена не будет (рисунок 46).

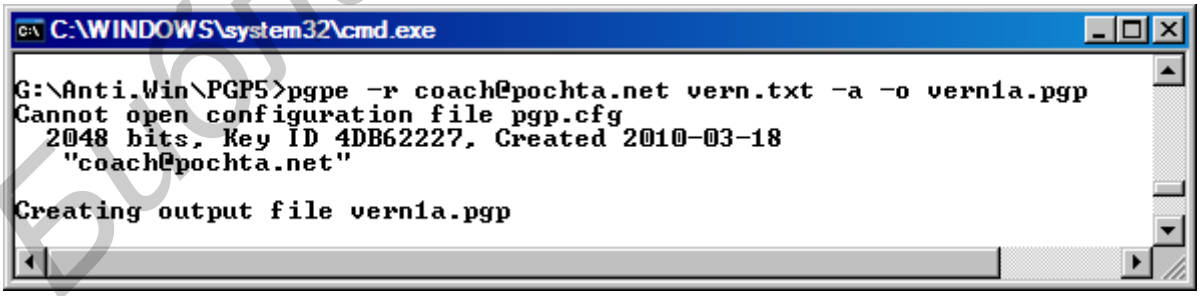


Рисунок 46 – Сообщение программы PGP об успешном кодировании данных первой половиной стандартной кодовой таблицы

Упражнение для самостоятельной работы 13. Сравните размеры полученных зашифрованных файлов vern1.pgp и vern1a.pgp.

Доступными вам средствами просмотрите файлы vern1.pgp и vern1a.pgp. Обратите внимание, что в файле vern1.pgp присутствуют все 256 символов из всего диапазона кодовой таблицы (00 – FFhex), в то время как в файле vern1a.pgp – только первые 128 символов (00 – 79hex). Сделайте выводы.

2.4.4 Декодирование принятого зашифрованного сообщения с помощью private-ключа

Ваш корреспондент получил зашифрованное сообщение. Для того чтобы его прочитать, корреспондент должен декодировать (расшифровать) сообщение, используя свой private-ключ. Для этого в командной строке терминала он должен набрать

pgpv file_name –o output_file_name.

Теперь программа, прежде чем приступить к процессу декодирования, в интерактивном режиме попросит вас ввести пароль. Будьте аккуратны, имейте в виду, при вводе пароля эта программа никак не реагирует на ваши нажатия клавиш: не подает звуковых сигналов, курсор не перемещается, никаких символов «звёздочка» не появляется. Завершение ввода пароля подтверждается клавишей [Enter].

Если вы сделали всё правильно, программа сообщит, что пароль правильный (Pass phrase is good) и декодирует зашифрованный файл (рисунок 47).

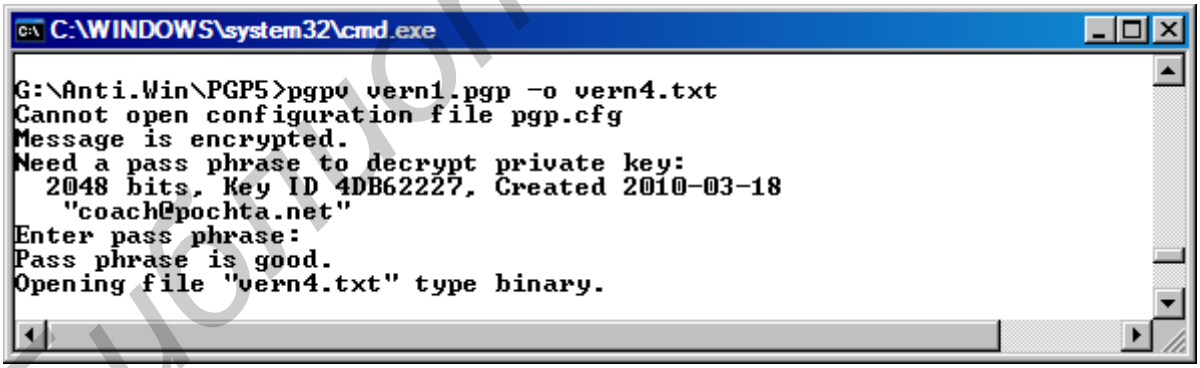


Рисунок 47 – Запрос пароля и сообщение программы PGP об успешном декодировании данных

В этой операции программа будет использовать private-ключ и спросит пароль (видно на копии экрана). В случае совпадения паролей будет проведено декодирование файла vern1.txt в файл vern4.txt.

Декодирование 7-битного файла vern1a.txt в файл vern4a.txt происходит аналогично, с теми же опциями командной строки, не надо указывать, что в файле присутствуют символы только первой половины кодовой таблицы, программа сама в этом разберётся. Это показано на следующей копии экрана (рисунок 48).

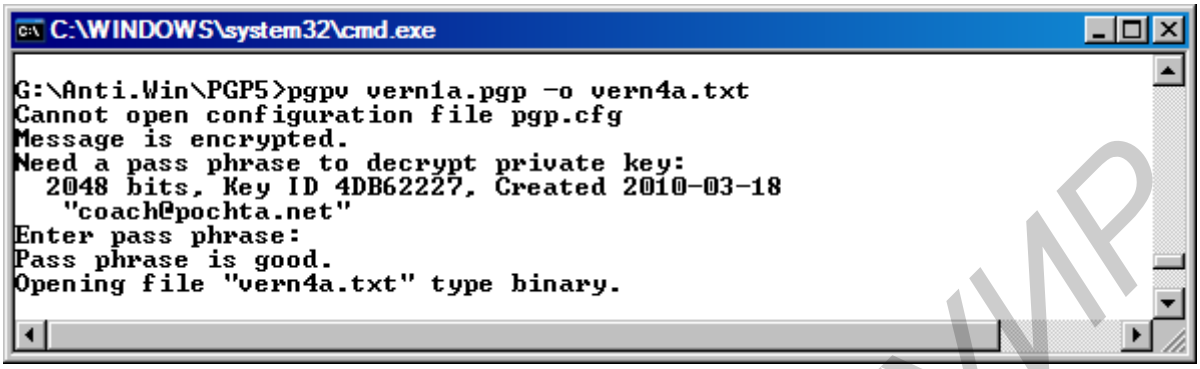


Рисунок 48 – Запрос пароля и сообщение программы PGP об успешном декодировании данных, использующих только первую половину кодовой таблицы

Упражнение для самостоятельной работы 14. Сравните размеры и содержимое файлов vern.txt, vern4.txt и vern4a.txt. Одинаковое ли у них содержимое? Нарушена ли целостность файла vern.txt при кодировании/декодировании алгоритмом RSA? Сделайте соответствующие выводы.

2.4.5 Кодирование и декодирование сообщения с помощью произвольного ключа

Вполне нормальна и типична ситуация, если пакетом PGP и алгоритмом RSA захочет воспользоваться другой пользователь этого компьютера и создать свою пару ключей. Но программа по умолчанию создаст пару pubring.pkr и secring.skr и затрёт ваши существующие ключи. Для решения этой проблемы и однозначной идентификации обычно каждый пользователь переименовывает свою пару ключей в соответствии со своими вкусами. В командной строке пакету PGP необходимо указать, ключ какого пользователя вы хотите употребить. Новые ключи необходимо зафиксировать в файле pgp.cfg. Если этого не сделать, программа будет сообщать об отсутствии этого файла, что проиллюстрировано на рисунке 49. Это не является ошибкой и не сказывается на работоспособности пакета, просто вы должны теперь каждый раз указывать эти параметры в командной строке.

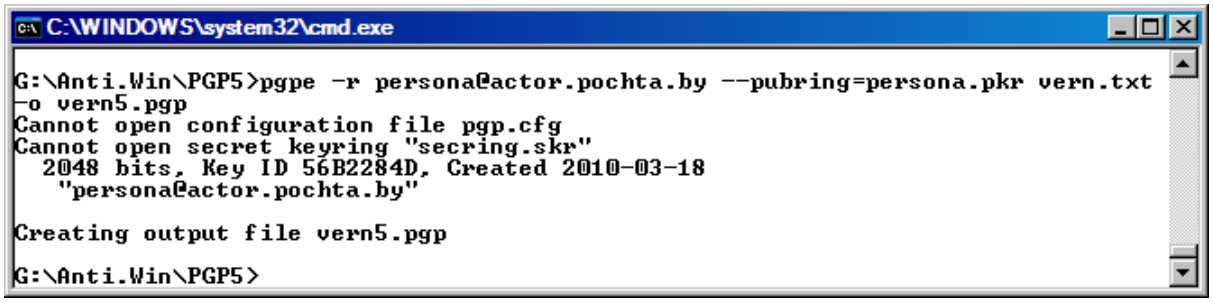


Рисунок 49 – Кодирование данных программой PGP произвольным публичным ключом пользователя

Это делается следующим образом. В командной строке терминала набираем

pgpe –r ID –-pubring=persona.pkr file_name –o output_file_name.

Идентификатор (ID) должен быть того пользователя, чей ключ вы используете, того, кому вы намереваетесь отправить сообщение, иначе получите от программы сообщение об ошибке. В данном примере пользователю Persona соответствует идентификатор его электронной почты persona@actor.pochta.by и пара ключей persona.pkr, и persona.skr.

На приведенной копии экрана (см. рисунок 44) видна работа программы и её сообщения.

Если ключи persona.pkr, persona.skr находятся не в текущем каталоге, то в соответствующей опции командной строки --pubring=.. указываем полный путь к файлу:

--pubring=full_path\persona.pkr

Аналогичная ситуация, если пользователь Persona получил закодированное сообщение и хочет его расшифровать, употребив свой не стандартно именованный private-ключ, тогда в командной строке он набирает

pgpv —secring=persona.skr file_name –o output_file_name.

Работа программы проиллюстрирована на рисунке 50.

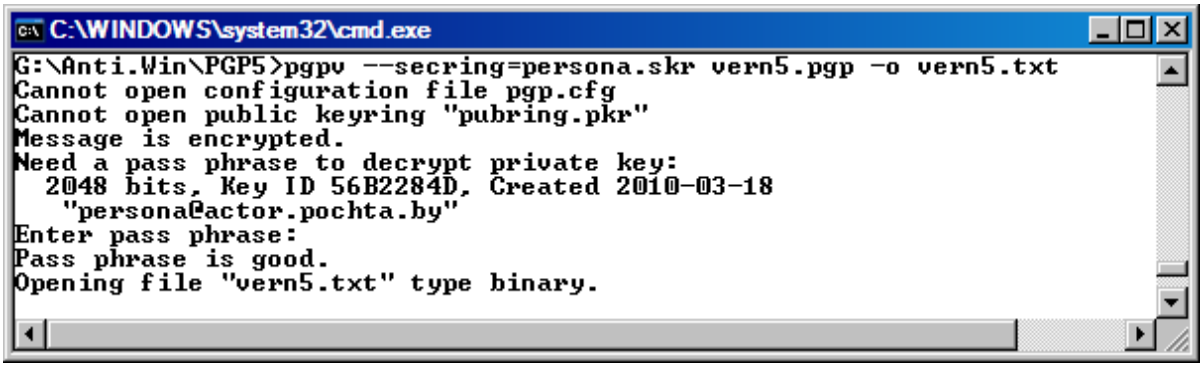


Рисунок 50 – Запрос пароля и декодирование данных программой PGP произвольным private-ключом пользователя

2.4.6 Аутентификация сообщения. Создание простой цифровой подписи в пакете PGP

Весьма важной является задача опознания, аутентификация пришедшего электронного сообщения, то есть действительно ли пришедшее сообщение принадлежит тому лицу, подпись которого стоит под электронным сообщением? Здесь проблема конфиденциальности уходит на второй план, а на первый план выступает проблема целостности, то есть не «лазил» ли кто в документ и не изменил ли его содержимое, и не приписал ли «лишний нолик» в цифре вашего платежа?

Описываемый пакет PGP предлагает несколько вариантов. Рассмотрим их. Persona решил разослать своим корреспондентам важное сообщение, но шифрует сообщение private-ключом! Нет ли здесь ошибки? Всё правильно. Алгоритм RSA работает в обе стороны и кодировать сообщение можно private-ключом. Но тогда каждый имеющий публичный ключ Persona может легко расшифровать сообщение. И где же тут секретность, конфиденциальность? А она тут не нужна, гораздо важнее то, что декодировав сообщение Persona ключом persona.pkr, вы убедились, что ключ подошёл, тем самым решаются два важнейших момента:

1) это сообщение действительно от Persona (иначе этим ключом какое-либо иное сообщение декодировать бы не удалось), тем самым осуществлена аутентификация сообщения Persona;
2) целостность сообщения в пути следования не нарушена, для этого злоумышленник должен, во-первых, расшифровать сообщение (это он может сделать, имея persona.pkr – публичный ключ Persona), во-вторых, изменить данные и, в-третьих, снова их зашифровать (подписать), но это невоз-

можно, поскольку private-ключ вместе с паролем находится только у Persona.

2.4.7 Создание цифровой подписи и сообщения единым файлом

В командной строке терминала набираем

pgps -u ID –-secring=persona.skr file_name –o output_file_name.

Эта команда кодирует файл vern.txt private-ключом и создаёт файл vern.pgp, содержащий цифровую подпись (рисунок 51).

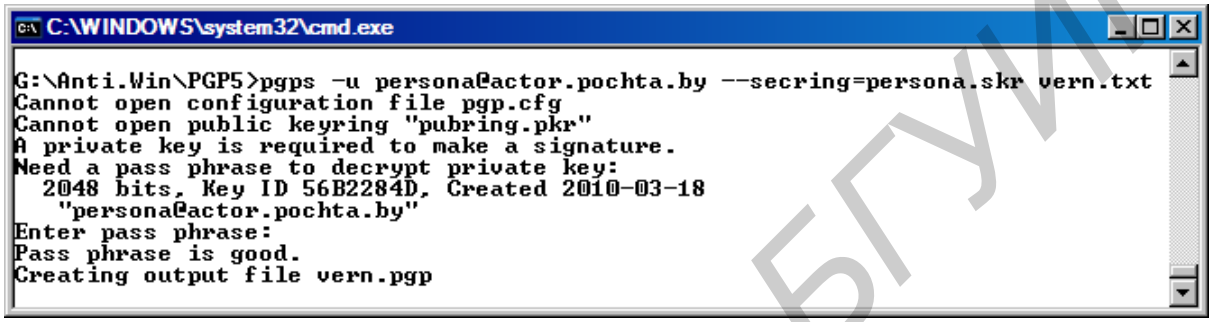


Рисунок 51 – Запрос на ввод пароля и создание цифровой подписи программой PGP произвольным private-ключом пользователя

На копии экрана (см. рисунок 46) мы видим, что для этой операции программа PGP требует пароль.

2.4.8 Создание цифровой подписи отдельным файлом

Можно создать цифровую подпись отдельным файлом, а само сообщение оставить открытым и незашифрованным.

В командной строке терминала набираем предыдущую команду, указав дополнительную опцию –b (рисунок 52):

pgps -u ID -b –-secring=persona.skr file_name.

На копии экрана мы видим, что программа создала файл цифровой подписи vern.sig. Изначальный файл vern.txt с данными не модифицировался.

Упражнение для самостоятельной работы 15. Посмотрите, какой размер у файла vern.sig. Много это или мало?

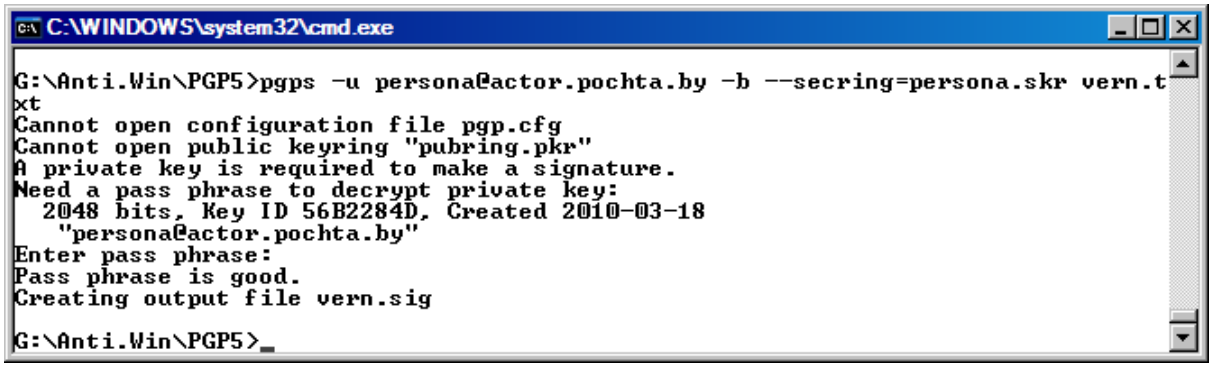


Рисунок 52 – Запрос на ввод пароля и создание цифровой подписи отдельным файлом

2.4.9 Верификация цифровой подписи

Вы получили файл vern.txt вместе с цифровой подписью vern.sig и хотите проверить (верифицировать), соответствует ли эта цифровая подпись полученным данным. Для этого нам нужен только лишь открытый, публичный ключ того лица, которое создало и подписало эти данные. Публичный ключ Persona у нас имеется. В командной строке набираем

pgpv vern.sig –-pubring=persona.pkr.

После запуска этой команды программа PGP переходит в интерактивный режим и просит вас ввести файл с данными, соответствующий цифровой подписи vern.sig. Мы набираем vern.txt, если файл с данными находится в другом каталоге, необходимо указать полный путь к файлу (рисунок 53).

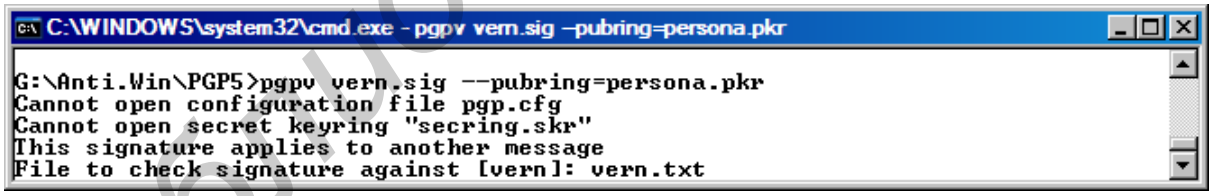


Рисунок 53 – Этап верификации цифровой подписи: запрос на ввод имени верифицируемого файла

Когда путь к файлу введен, программа продолжит работу: сообщит, во-первых, идентификатор пользователя, поставившего цифровую подпись и, во-вторых, результат верификации. Сообщение на копии экрана (рисунок 54) «Good signature» означает, что верификация прошла успешно.

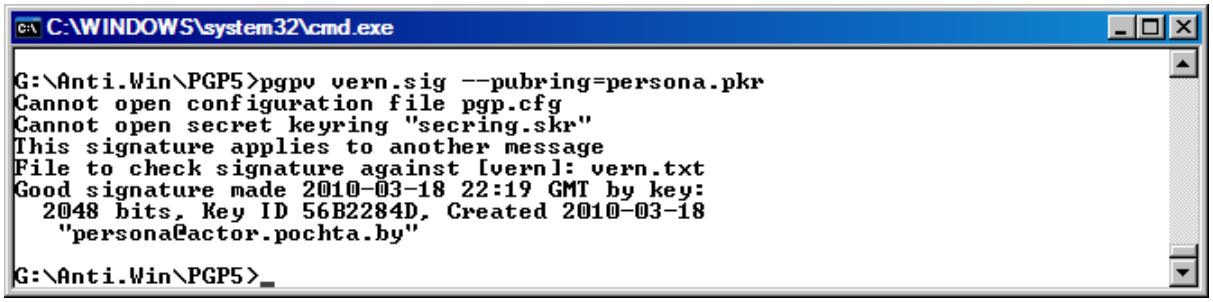


Рисунок 54 – Этап верификации цифровой подписи: завершение и сообщение об успешной верификации

Упражнение для самостоятельной работы 16. Измените хотя бы один бит в файле vern.txt. Постарайтесь не менять размер файла. Попытайтесь сделать верификацию так, как описано выше.

2.4.10 Сбой верификации цифровой подписи

Допустим, злоумышленник изменил подписанные данные (или вы старательно и правильно выполнили последнее самостоятельное упражнение). Как пройдёт верификация?

Осуществляем верификацию, как описано в предыдущем пункте 2.4.9. Выполняем ту же команду

pgpv vern.sig –-pubring=persona.pkr.

Снова указываем путь к файлу с данными, но теперь обратите внимание (рисунок 55), программа, во-первых, вывела идентификатор лица, поставившего цифровую подпись и, во-вторых, вывела тревожное сообщение: «BAD signature» – это означает, что целостность файла нарушена, данные кто-то менял и модифицировал, уже после того как была создана цифровая подпись.

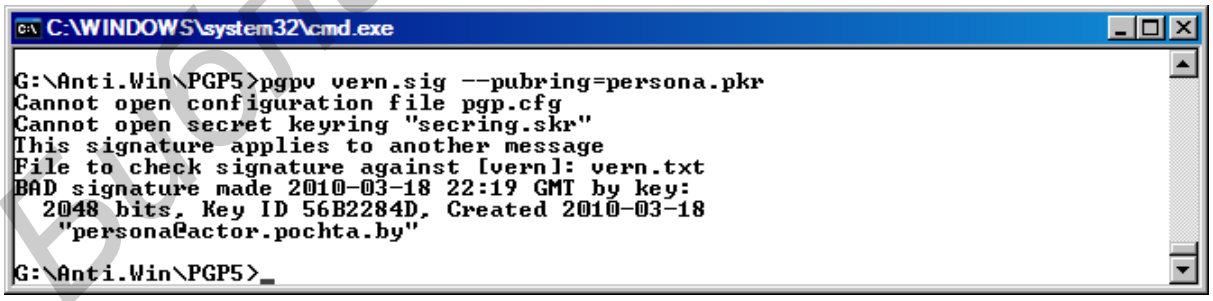


Рисунок 55 – Этап верификации цифровой подписи: завершение и сообщение о сбое верификации

Мы научились создавать и верифицировать простую цифровую подпись. Этот механизм хорошо работает, если стороны доверяют друг другу. Однако в жизни, в производственной деятельности, в

бизнесе возникают ситуации, когда мы с подозрением относимся к партнёру.

Типичный пример – вся финансовая работа банка видна из правильно составленного годового бухгалтерского баланса, тем не менее в финансовых кругах не очень доверяют такому балансу. И администрация банка вынуждена, часто за значительные средства, привлекать независимую нейтральную организацию-арбитра, называемую аудиторской фирмой, которая составляет такой баланс. Баланс, составленный независимой аудиторской фирмой, имеет значительно более высокий рейтинг и может служить «доверительной» визитной карточкой финансового благополучия предприятия.

Так и с арбитражной цифровой подписью – если в своей переписке корреспонденты не доверяют друг другу, то они ищут третью, нейтральную доверительную сторону – арбитра. Механизм создания и использования арбитражной подписи подробно разобран в пункте 1.6.7, поэтому практическое задание по созданию арбитражной подписи выносится в самостоятельное упражнение.

Упражнение для самостоятельной работы 17. Привлеките на помощь друга. Создайте ещё пару ключей. Выберите арбитра. Создайте арбитражную цифровую подпись, как описано в пункте 1.6.7.

2.5 Практическая работа с пакетом GnuPG в Linux

Рассмотрим некоторые возможности работы пакета GnuPG в Linux. Упражнения, приведенные ниже, демонстрируют возможности системы PGP, поскольку в системе PKI пользователь лишён возможности создавать сам себе ключи (см. подраздел 1.7). Это является серьезной брешью безопасности системы X.509. Действительно, почему пользователь должен доверять секретному ключу и паролю который сформировал кто-то другой, в каком-то другом месте и переслал ему? Этот пакет подпадает под лицензию с открытым исходным кодом и в ОС Linux именуется GnuPG. Сама программа вызывается из командной строки как gpg.

Программа поддерживает следующие алгоритмы шифрования с открытым ключом: RSA, ELG, DSA. Программа поддерживает также симметричные шифры IDEA, 3DES, CAST5, BLOWFISH, AES, TWOFISH и следующие хэш-функции: MD5, SHA1, SHA512. Помимо этого, программа поддерживает алгоритмы сжатия ZIP, ZLIB, BZIP2.

Упражнение для самостоятельной работы 18. Программа gpg2 имеет огромное количество ключей и дополнительных параметров, которые сложно удержать в памяти, но справка по ним всегда доступна из командной строки. Самое быстрое и краткое – это набрать gpg2 --help. Можно воспользоваться стандартной UNIX-справкой man gpg2. Доступна так же справка info gpg2. Наберите в консоли последовательно все эти три варианта и ознакомьтесь с приведенной электронной документацией.

2.5.1 Создание пары ключей

Программа вызывается из командной строки как gpg2. Весь пакет лежит в каталоге /usr/bin/gpg2. Для совместимости с предыдущими версиями в этом же каталоге имеется ссылка /usr/bin/gpg, указывающая на gpg2, поэтому неважно, что вы набрали в командной строке gpg или gpg2, результат будет один и тот же. Создадим теперь пару ключей. Это осуществляется следующей командой:

gpg2 --gen-key,

после чего программа запускается в интерактивном режиме, предлагая пользователю ввести необходимые параметры для генерации ключей (рисунок 56). Рассмотрим поэтапно, что это за параметры и какова альтернатива их выбора. Внимательный читатель наверняка заметит, что это – **те же самые параметры**, которые мы с вами разбирали в пункте 2.4.2. Сменилась операционная система и интерфейс программы, но сердцевина схемы асимметричного алгоритма шифрования RSA осталась.

В ОС Linux все ключи хранятся в определённом месте – в домашнем каталоге пользователя, в подкаталоге ~/.gnupg. Программа gpg не создаёт отдельный файл для каждого ключа. Здесь публичные и приватные ключи хранятся в так называемых связках, в одном файле. Это связано с тем, что предполагается – у вас обширная переписка со множеством корреспондентов и естественно, для каждого корреспондента у вас имеется его публичный ключ. Связка публичных ключей хранится в файле ~/.gnupg/pubring.gpg, а связка приватных – соответственно в ~/.gnupg/secring.gpg.

Первое, что спрашивает программа при создании пары ключей, какой алгоритм шифрования нам нужен. У нас есть выбор из двух алгоритмов, которые распадаются на четыре варианта. Алгоритм DSA расчитан только на создание цифровой подписи и передачу ключей симметричного шифрования, поэтому, если мы хотим

шифровать сообщения, мы выбираем алгоритм RSA из первого варианта. Это видно из рисунка 56.

```
linux-8ks6:~ # gpg2 --gen-key
gpg (GnuPG) 2.0.19; Copyright (C) 2012 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (sign only)
   (4) RSA (sign only)
Your selection? 1
```

Рисунок 56 – Создание пары ключей

Итак, на рисунке 56 мы выбрали «1». Обратите внимание на схожесть вопросов в аналогичном скриншоте, приведенном на рисунке 38, в другой операционной системе.

Дальше программа предлагает нам выбрать (рисунок 57) длину ключа и срок его действия. Выбираем длину 2048 бит, а срок – один год, что видно из приведенного скриншота. Здесь также можно отметить аналогию с рисунком 39 и рисунком 41, где у пользователя программа запрашивает размер ключа и его срок действия соответственно.

После введения всех необходимых параметров появляется подтверждающее сообщение, показанное на рисунке 58, благодаря которому пользователь имеет возможность ещё раз проверить введенные параметры и, в случае необходимости, отменить данные и вернуться назад для коррекции.

Следующий шаг – программа запрашивает у нас дополнительную информацию для построения идентификаторов ключей. Аналогичная информация запрашивалась в PGP5 (см. рисунок 40), но там требовался всего лишь открытый идентификатор для опознания пользователей.

```
linux-8ks6:~ # gpg2 --gen-key
gpg (GnuPG) 2.0.19; Copyright (C) 2012 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (sign only)
   (4) RSA (sign only)
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048) 2048
Requested keysize is 2048 bits
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0) 1y
```

Рисунок 57 – Выбор размера ключа и срока его действия

```
Requested keysize is 2048 bits
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0) 1y
Key expires at Thu Jun 12 21:16:45 2014 FET
Is this correct? (y/N) y
```

Рисунок 58 – Подтверждающее сообщение программы

На рисунке 59 показан скриншот запроса программой трёх дополнительных параметров:

– реальное имя;
– электронный адрес;
– комментарий.

Эти дополнительные параметры отличают криптосистему **openGPG** от криптосистемы **PGP**. Они будут в дальнейшем исполь-

зоваться в системе сертификатов и подключей, с которыми не работает криптосистема PGP.

```
GnuPG needs to construct a user ID to identify your key.

Real name: gerasim2013
Email address: gerasim2013@mumu.sobaka.by
Comment: для целей обучения
```

Рисунок 59 – Запрос программой дополнительных параметров

Пункт «комментарий» мы ввели символами кириллицы. Предупредительная программа сообщает нам далее, что мы использовали нелатинские символы в кодировке UTF-8. Это нелишнее предупреждение говорит о двух особенностях. Во-первых, программа информирует, что криптосистема openGPG оказывает поддержку и шифрование при использовании на многих языках. Во-вторых, выдает дублирующее предупреждение, что несмотря на многоязычную поддержку, программа не гарантирует защиту пользователя от трудностей, с которыми он может столкнуться в процессе работы и передачи сообщений, содержащих символы кодировки UTF-8.

Наконец, программа повторно хочет удостовериться в правильности введенных данных и предлагает пользователю перепроверить введенные данные. Теперь, когда почти всё готово, мы должны ввести пароль для приватного ключа – последняя строка скриншота (рисунок 60).

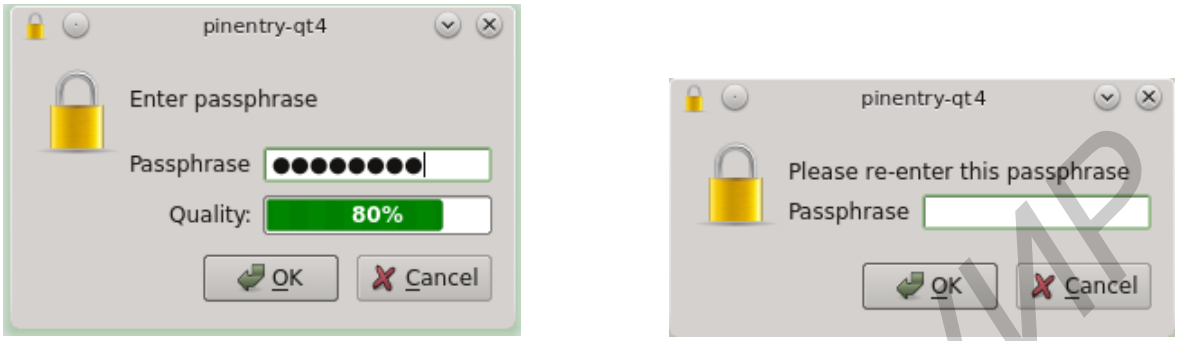
```
Real name: gerasim2013
Email address: gerasim2013@mumu.sobaka.by
Comment: для целей обучения
You are using the `utf-8' character set.
You selected this USER-ID:
    "gerasim2013 (для целей обучения) <gerasim2013@mumu.sobaka.by>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? o
You need a Passphrase to protect your secret key.
```

Рисунок 60 – Запрос о корректности
введенных пользователем данных

Для ввода пароля программа переходит в систему xWindow с графическим интерфейсом для возможности показа окна ввода дан-

ных и кнопок «OK», «Cancel». Эти окна используют графический интерфейс также для контроля качества вводимого пароля. Программа предлагает ввести пароль – Enter passphrase. Пароль мы должны ввести в поле Passphrase (рисунок 61).



а б

а – ввод пароля; б – повторный ввод пароля

Рисунок 61 – Ввод пароля

Считается, что защита пароля от взлома и качество шифрования напрямую зависит от количества символов содержащихся в пароле. Этот факт программа сообщает пользователю в поле Quality цветом и числом процентов: «плохой» пароль – красным цветом, по мере ввода символов цвет меняется на зелёный (рисунок 62).

Рисунок 62 – Проверка надежности пароля в поле Quality

Ввиду важности информации о пароле окно для его ввода появится дважды и нам придётся вводить его повторно. Если при вводе пароля мы ошиблись и, вводя его второй раз в окне Please re-enter this passphrase рисунок 61, б, нажали не ту клавишу, то программа это заметит – Does not match – try again, введенную информацию не воспримет и заставит повторить всю процедуру ввода пароля сначала. Что отображено на рисунке 62. Если же всё пра-

вильно, фокус ввода перемещается операционной системой снова в консоль, где программа выводит окончательное резюме (рисунок 63).

```
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: key C30636C2 marked as ultimately trusted
public and secret key created and signed.

gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid:   3  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 3u
gpg: next trustdb check due at 2014-06-11
pub   2048R/C30636C2 2013-06-12 [expires: 2014-06-12]
      Key fingerprint = 8E6F 1B64 E4DE 7E1E C359  313A 1742 901B C306 36C2
uid                  gerasim2013 (для целей обучения) <gerasim2013@mumu.sobaka.by>
sub   2048R/B8CE6604 2013-06-12 [expires: 2014-06-12]
```

Рисунок 63 – Заключительное сообщение программы

В заключительном сообщении на рисунке 63 мы видим рекомендации о создании качественных, криптостойких ключей, а также окончательную итоговую информацию о созданном ключе. Теперь имеет смысл проверить информацию о созданном ключе в связке наших ключей (рисунок 64).

```
linux-8ks6:~ # gpg2 --list-keys
/root/.gnupg/pubring.gpg
------------------------
pub   2048R/DFC35394 2013-06-11 [expires: 2014-06-11]
uid                  gerasim80 (else key inSuSE 12.3) <gerasim80@mumu.sobaka.by>
sub   2048R/3466530D 2013-06-11 [expires: 2014-06-11]

pub   4096R/50F3C2BB 2013-06-11 [expires: 2014-06-11]
uid                  gerasim81 (open SuSE 13.1) <gerasim81@mumu.sobaka.by>
sub   4096R/E53F6BCF 2013-06-11 [expires: 2014-06-11]

pub   2048R/C30636C2 2013-06-12 [expires: 2014-06-12]
uid                  gerasim2013 (для целей обучения) <gerasim2013@mumu.sobaka.by>
sub   2048R/B8CE6604 2013-06-12 [expires: 2014-06-12]
```

Рисунок 64 – Окно вывода списка всех ключей пользователя

Для проверки и вывода списка всех ключей пользователя набираем в командной строке

gpg2 --list-keys.

В первой строке программа выводит полный путь к файлу (см. рисунок 64), где хранятся наши ключи, а в следующих строках выводится список всех ключей и их характеристики: длина ключа, идентификатор, срок действия, имя, адрес электронной почты и комментарий. Новый, только что созданный нами ключ тоже появляется в списке уже используемых ключей (обведен овалом). Для вывода списка приватных ключей используется следующая команда:

gpg2 --list-secret-keys.

Замечание – Поскольку Linux – многопользовательская операционная система, все манипуляции с пакетом GnuPG, описываемые в этом разделе, будут работать через консоль удалённого доступа, например, по протоколу SSH.

Мы можем создавать ключевую пару удалённо, через соединение SSH, запустив в терминале такую же команду gpg2 --gen-key и попав в интерактивное меню. Вопросы нам будут заданы точно такие же. Правда, есть небольшие различия: поскольку графический режим xWindow протокол SSH не поддерживает, то дойдя до приглашения ввести пароль для нового приватного ключа, система не будет формировать цветное графическое окно, но возникнет монохромная рамка из символов псевдографики с приглашением ввести пароль. На рисунке 65 показана такая рамка для ввода пароля, аналогичная тому, что изображено на рисунке 61.

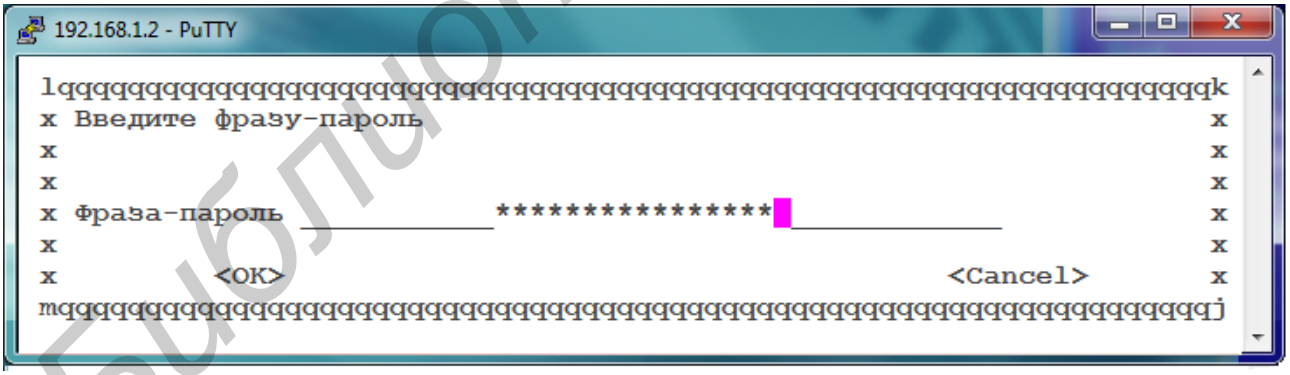


Рисунок 65 – Ввод пароля в консоли удалённого доступа

Упражнение для самостоятельной работы 19. Создайте несколько пар ключей, которые понадобятся вам для выполнения дальнейших упражнений. Используйте пары с алгоритмами RSA и DSA и несколько пар с различной длиной ключа.

2.5.2 Создание зашифрованного сообщения и его дешифрование

Зашифруем подходящий файл открытым ключом gerasim. Для этого в командной строке набираем команду, такую же как на рисунке 66.

```
strekoza:~/ke # gpg2 --encrypt --recipient gerasim2013 --output ho10.gpg ho.txt
```

Рисунок 66 – Команда шифрования файла

Команда --encrypt будет шифровать данные; параметр --recipient указывает идентификатор пользователя, открытым ключом которого вы шифруете сообщение для последующей пересылки ему этого сообщения. В данном случае это идентификатор некоего пользователя gerasim2013, который указывается сразу после параметра. Затем параметр --output указывает, что данные необходимо записать в файл ho10.gpg, если этот параметр опустить, то шифрование по умолчанию будет осуществлятся в файл ho.txt.gpg. И самым последним в синтаксисе команды указывается файл с открытыми данными ho.txt.

Дешифрование осуществляется аналогичным образом. Допустим, мы сейчас являемся тем самым пользователем gerasim2013, который получил зашифрованное сообщение в файле ho10.gpg. Команда дешифрования, которую мы должны набрать, выглядит следующим образом:

gpg2 --decrypt --output ho10.txt ho10.gpg.

Обратите внимание, что здесь отсутствует идентификатор пользователя, но программа должна сама его определить.

На рисунке 67 показан результат выполнения этой команды.

```
strekoza:~/ke # gpg2 --decrypt --output ho10.txt ho10.gpg

You need a passphrase to unlock the secret key for
user: "gerasim2013 (для целей обучения) <gerasim2013@mumu.sobaka.by>"
2048-bit RSA key, ID B6255352, created 2013-07-04 (main key ID F8063696)
```

Рисунок 67 – Команда дешифрования файла. Начало

Программа действительно опознала, что это зашифрованное сообщение предназначено пользователю gerasim2013, сообщила ряд дополнительных сведений о публичном ключе и самое главное потребовала ввести пароль в поле для ввода, показанное на рисунке 68.

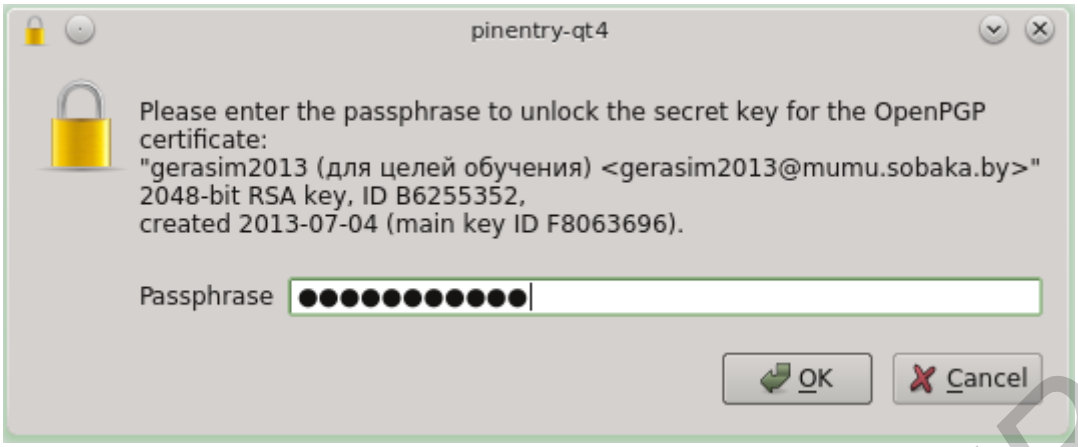


Рисунок 68 – Приглашение для ввода пароля

Для ввода пароля программа перешла в систему xWindow с графическим интерфейсом для возможности показа окна ввода и кнопок «OK», «Cancel». Мы видим, что в этом окне программа дублирует некоторую информацию, которая перед этим была выведена в текстовую консоль.

После ввода пароля нажимаем «OK» и, если пароль введен правильно, графическое окно исчезает, и в текстовой консоли (рисунок 69) мы видим продолжение вывода, где в предпоследней строке программа докладывает об успешной расшифровке данных: gpg: encrypted with 2048 key.

```
strekoza:~/ke # gpg2 --decrypt --output ho10.txt ho10.gpg

You need a passphrase to unlock the secret key for
user: "gerasim2013 (для целей обучения) <gerasim2013@mumu.sobaka.by>"
2048-bit RSA key, ID B6255352, created 2013-07-04 (main key ID F8063696)

gpg: encrypted with 2048-bit RSA key, ID B6255352, created 2013-07-04
      "gerasim2013 (для целей обучения) <gerasim2013@mumu.sobaka.by>"
```

Рисунок 69 – Команда дешифрования файла.
Успешное завершение

Если введенный пароль неверен, то результат будет иным (рисунок 70). Графическое окно не исчезает и на нём появляется красная надпись «Invalid passphrase: please try again» (Неправильный пароль, попробуйте ещё раз). Программа предоставит вам несколько попыток для ввода правильного пароля.

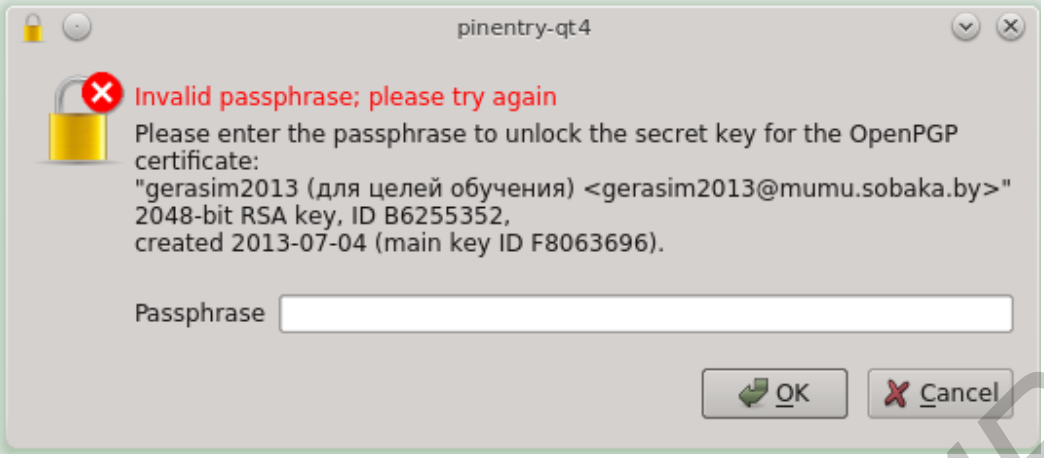


Рисунок 70 – Повторное приглашение для ввода пароля

В случае неудачного ввода пароля графическое окно исчезает, но текст сообщения в консоли будет уже другим (рисунок 71). Программа выводит дублирующее сообщение о неправильном пароле и краткое сообщение со служебной информацией о неудачной попытке расшифровки файла: gpg: decryption failed: No secret key.

Замечание – Если для каких-то целей необходимо несколько раз зашифровать один и тот же открытый файл, например ho.txt, одним и тем же открытым ключом, командой, приведенной на рисунке 65, то результат выполнения программы будет каждый раз разный, то есть каждый раз мы будем получать другой файл ho10.gpg. Это связано с тем, что система шифрования gpg рандомизированная, о чём упоминалось в пункте 1.6.7. Но при дешифровании приватным ключом из всех различных шифрованных файлов вы всегда будете получать правильный результат – это всегда будет изначальный файл ho.txt.

```
strekoza:~/ke # gpg2 --decrypt --output ho10.txt ho10.gpg

You need a passphrase to unlock the secret key for
user: "gerasim2013 (для целей обучения) <gerasim2013@mumu.sobaka.by>"
2048-bit RSA key, ID B6255352, created 2013-07-04 (main key ID F8063696)

gpg: Invalid passphrase; please try again ...

You need a passphrase to unlock the secret key for
user: "gerasim2013 (для целей обучения) <gerasim2013@mumu.sobaka.by>"
2048-bit RSA key, ID B6255352, created 2013-07-04 (main key ID F8063696)

gpg-agent[14043]: command get_passphrase failed: Operation cancelled
gpg: cancelled by user
gpg: encrypted with 2048-bit RSA key, ID B6255352, created 2013-07-04
      "gerasim2013 (для целей обучения) <gerasim2013@mumu.sobaka.by>"
gpg: public key decryption failed: Operation cancelled
gpg: decryption failed: No secret key
```

Рисунок 71 – Команда дешифрования файла.
Сбой ввода пароля

Упражнение для самостоятельной работы 20. Теперь у вас есть несколько пар ключей, которые вы создали в предыдущем упражнении. Обменяйтесь открытыми ключами с вашими друзьями. Зашифруйте открытым ключом вашего друга по описанной выше методике свою фотографию и отправьте ему, пусть попытается рассмотреть фотографию. Попросите его в свою очередь зашифровать вашим открытым ключом «роман», который он пишет и пусть вышлет вам этот файл. Попытайтесь его прочесть.

2.5.3 Электронная цифровая подпись

Криптография на основе открытого ключа может использоваться не только для шифрования, но и для аутентификации. Вот почему одна из важнейших и востребованных в настоящее время функций, осуществляемая схемой асимметричного шифрования – создание цифровой подписи. Мы достаточно подробно разобрали этот вопрос в подразделе 2.5. для пакета PGP 5. Пакет GnuPG под ОС Linux, конечно же, тоже поддерживает эти функции.

Аутентификация с помощью цифровой подписи в данном случае возможна, потому что получатель может открытым ключом расшифровать данные, зашифрованные закрытым ключом. Пакет GnuPG значительно более гибок и более функционален, нежели PGP 5. На рисунке 72 показано, как создать цифровую подпись и осуществить её верификацию.

```
strekoza:~/ke # gpg2 --detach-sign --local-user gerasim2013 --output ho.txt.sig ho.txt

You need a passphrase to unlock the secret key for
user: "gerasim2013 (для целей обучения) <gerasim2013@mumu.sobaka.by>"
2048-bit RSA key, ID F8063696, created 2013-07-04
```

Рисунок 72 – Создание цифровой подписи и сопутствующие сообщения программы

Наберите в командной строке следующую последовательность команд:

gpg2 --detach-sign --local-user gerasim2013 --output ho.txt.sig ho.txt.

Здесь используются следующие команды и параметры:

--detach-sign предписывает программе создать отдельную цифровую подпись;

--local-user разрешает употребить идентификатор локального пользователя, и на самом последнем месте в строке стоит имя файла ho.txt , подпись для которого будет создана.

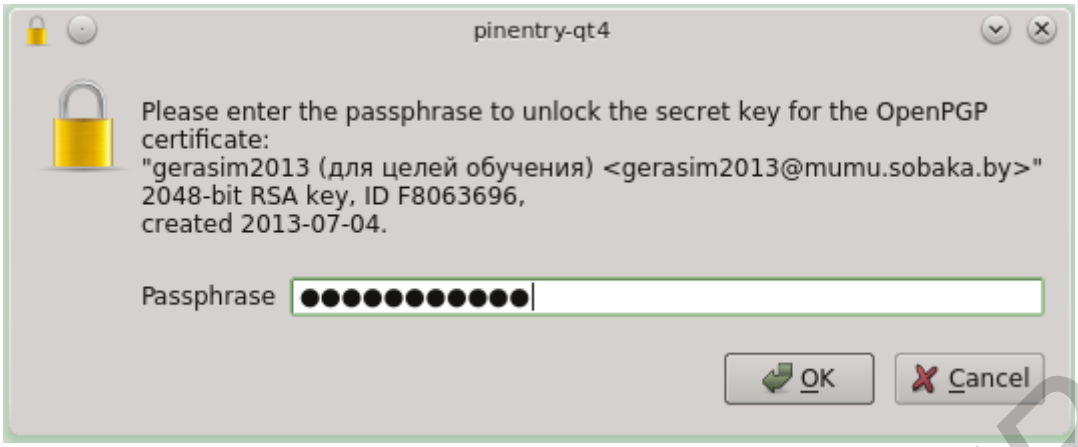


Рисунок 73 – Команда дешифрования файла. Сбой ввода пароля

Поскольку для создания цифровой подписи требуется закрытый ключ, то, естественно, при запуске снова возникнет графическое окно с полем для ввода пароля. В этом окне можно ещё раз увидеть идентификатор и прочую информацию о том локальном пользователе, ключом которого будет подписан файл. Если пароль введен правильный, графическое окно исчезнет и мы увидим заключительное сообщение, как на рисунке 72.

По умолчанию файл подписи будет создан в том же каталоге путём добавления расширения .sig. Для создания подписи в другом месте и с другим именем используем параметр --output. Если всё сделано правильно, результат работы программы будет аналогичен показанному на рисунке 72.

OpenPGP предоставляет два способа добавления цифровых подписей: встраивание и отделение. Встроенная цифровая подпись добавляется в исходное сообщение, то есть сами данные и цифровая подпись сохраняются в одном файле. В случае с отдельной цифровой подписью исходный файл не изменяется, а подпись сохраняется во втором файле, в этом случае обычно получается два файла. Желательно использовать метод с отдельной подписью, потому что он может использоваться для подписания файлов любого типа, тогда как метод со встроенной подписью – только для простых текстовых файлов.

```
strekoza:~/ke # gpg2 ho.txt.sig
gpg: Signature made Fri Jul  5 14:56:21 2013 FET using RSA key ID F8063696
gpg: Good signature from "gerasim2013 (для целей обучения) <gerasim2013@mumu.sobaka.by>"
strekoza:~/ke #
```

Рисунок 74 – Успех верификации цифровой подписи

Чтобы убедиться в подлинности цифровой подписи, получатель тоже вычисляет контрольную сумму и сравнивает её значение с

тем, что содержится в подписи. Если эти значения совпадут: во-первых, можно считать, что данные не были изменены и целостность не нарушена, и, во-вторых, сообщение было подписано с помощью закрытого ключа. Если данные были изменены, контрольные суммы совпадать не будут. Аналогично, если оригинальная контрольная сумма была зашифрована некоторым другим закрытым ключом, результат расшифровки с помощью открытого ключа будет представлять собой бессмысленный набор символов и контрольные суммы также не будут совпадать.

Команда проверки цифровой подписи необыкновенно проста. Мы набираем для этого следующее (см. рисунок 74):

gpg2 ho.txt.sig.

В данном случае не требуется ни идентификатор пользователя, информация о котором содержится в самой подписи, ни ввод пароля, поскольку для проверки цифровой подписи нужен **открытый** ключ. В случае успешной верификации программа сообщает: «Good signature» (см. рисунок 74).

Если же контрольные суммы не совпадают, вывод сообщения программы будет другим: «BAD signature» (рисунок 75).

```
strekoza:~/ke # gpg2 ho.txt.sig
gpg: Signature made Fri Jul  5 14:56:21 2013 FET using RSA key ID F8063696
gpg: BAD signature from "gerasim2013 (для целей обучения) <gerasim2013@mumu.sobaka.by>"
```

Рисунок 75 – Сбой верификации цифровой подписи

Упражнение для самостоятельной работы 21. Воспользовавшись ключами, созданными в упражнениях 17, 18, подпишите своим секретным ключом свою фотографию и отправьте другу, которому вы предоставили свой открытый ключ. Попросите друга, открытый ключ которого у вас имеется, создать цифровую подпись «романа», который он вам отошлёт почитать. Убедитесь в целостности присланного «романа». Как вы думаете, если его кто-то читал в «дороге», будет ли нарушена цифровая подпись? Как вы думаете, если вашу отосланную фотографию кто-то рассматривал в «дороге», ваш друг обнаружит этот факт, сверяя цифровую подпись?

Упражнение для самостоятельной работы 22. Выполнив предыдущее упражнение, проделайте следующее: измените только одну букву в присланном «романе» друга. Как пройдёт верификация цифровой подписи в этом случае? Разрешите вашему другу, получившему вашу фотографию, нарисовать у вас на носу ма-

ленькую родинку, и пусть сохранит этот файл. Спросите его после этого, как теперь прошла верификация цифровой подписи?

Упражнение для самостоятельной работы 23. Теперь вам необходимо сделать следующее. Как заставить программу GnuPG и зашифровать вашу фотографию, и сделать цифровую подпись, то есть объединить всё в одной команде? Обратите внимание: шифрование будет осуществлено **открытым ключом** вашего друга, создание цифровой подписи будет осуществлено вашим **закрытым ключом**.

Подсказка: разберитесь в следующей командной строке:

gpg2 --sign --encrypt --recipient МОЙ_ДРУГ [портрет.jpg].

В случае правильного выполнения команда заработает. Если у вас все получилось, научите и друга – пусть присылает свой зашифрованный и подписанный «роман».

2.5.4 Импорт ключей

В системе GnuPG для Linux мы можем импортировать ключи, полученные в другой ОС. Продемонстрируем это. В пункте 2.4.2 мы создали пару ключей в текстовой консоли Windows для виртуальной машины MS DOS. Импорт осуществляется следующей командой:

gpg2 --import [file_name].

Сообщение программы показано на рисунке 76. Результат её действия таков: в связке открытых ключей ~/.gnupg/pubring.gpg появится импортированный ключ. Проверим это командой gpg2 --list-keys.

```
strekoza:~ # gpg2 --import /home/duralei/ke/Gera002.pkr
gpg: key 9B3F475B: public key "Gera002" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
```

Рисунок 76 – Импорт публичного ключа

Программа выводит сообщение, и мы видим, что ключ с идентификатором Gera002 появился в связке (рисунок 77).

```
strekoza:~ # gpg2 --list-keys
/root/.gnupg/pubring.gpg
------------------------
pub   2048R/F8063696 2013-07-04 [expires: 2014-07-04]
uid                  gerasim2013 (для целей обучения) <gerasim2013@mumu.sobaka.by>
sub   2048R/B6255352 2013-07-04 [expires: 2014-07-04]

pub   1024R/9B3F475B 2013-06-12
uid                  Gera002

pub   2048R/3A98327F 2013-07-05 [expires: 2014-07-05]
uid                  gerasim2012 (для целей образования) <gerasim2012@mumu.sobaka.by>
sub   2048R/848D0F09 2013-07-05 [expires: 2014-07-05]
```

Рисунок 77 – Появление импортированного ключа в связке ключей

Аналогичным образом можно импортировать и секретный ключ командой, описанной выше.

На рисунке 78 мы видим отчёт об импорте секретного ключа Gera002.skr в связку пользователя ~/.gnupg/secring.gpg .

```
strekoza:~ # gpg2 --import /home/duralei/ke/Gera002.skr
gpg: key 9B3F475B: secret key imported
gpg: key 9B3F475B: "Gera002" not changed
gpg: Total number processed: 1
gpg:              unchanged: 1
gpg:       secret keys read: 1
gpg:   secret keys imported: 1
```

Рисунок 78 – Импорт секретного ключа

Упражнение для самостоятельной работы 24. Импортируйте из системы PGP5 свою пару ключей в систему GnuPG Linux. Проверьте командой gpg2 --list-secret-keys, появился ли в домашнем каталоге в вашей связке этот секретный ключ.

Заключение

Авторы смогли лишь прикоснуться к маленькому краю огромной проблемы, называемой «безопасная работа в компьютерной сети». Ни в объём настоящей работы, ни в намерения авторов не входит попытка осветить проблему целиком. Иначе это будет многотомное издание.

Однако основные «вехи», которые очерчивают проблему, изложены. Информационная безопасность – проблема комплексная и многоуровневая, решать которую необходимо тоже комплексно на законодательном, административном, процедурном и программно-техническом уровнях. И всё равно мы не достигнем стопроцентного результата. Полная защита – недостижимая мечта. Реальная информационная защита – это компромисс. Компромисс реальной стоимости вашей информации с реальными средствами, которые вы готовы вложить в финансирование вашей системы защиты.

В данном пособии рассмотрены две основные системы шифрования: симметричная и с открытым ключом. Показано, что система с открытым ключом породила много интересных методов криптозащиты, широко используемых на практике, включая:

- простое шифрование почтовых сообщений;
- различные системы аутентификации, основанные на вычислении хэш-функций;
- цифровые подписи;
- сложную систему сертификации цифровых подписей.

Изложенные методы по возможности проиллюстрированы в практической работе в операционных системах: Linux и MS Windows. Практически показано следующее:

- шифрование данных с использованием симметричного алгоритма шифрования;
- использование методов стеганографии для скрытия передачи данных;
- как получить и проверить контрольную сумму, используя хэш-функцию;
- создание ключевой пары и использование открытого ключа для шифрования, расшифровывание секретным ключом;
- подпись электронной цифровой подписью своих данных, верификация этих данных открытым ключом.

Ещё раз напомним – нельзя забывать, что чудовищно огромная информационная система, называемая «Глобальная компьютерная

сеть» вышла из научных лабораторий, где высокоинтеллектуальные и высокосознательные научные работники и не помышляли о том, что их детище через двадцать лет проникнет в каждый дом, в каждую семью, в каждую организацию. И никаких проблем информационной безопасности не существовало. Позволим себе сравнение. Допустим на мгновение, что чисто научная уникальная установка типа синхрофазотрона или адронного коллайдера через двадцать лет вдруг обнаружит «страшно» полезные свойства для массового пользователя, найдут дешёвые технологии для её производства, и она заполонит нашу жизнь, став маленьким утилитарным прибором. А о безопасности никто и не подумает, и будущие хакеры тут же возьмут на вооружение в свой арсенал этот новый прибор. Примерно такая метаморфоза произошла с компьютерными сетями – мы оказались совершенно не готовыми и беззащитными перед новой технологией, которая нам столь необходима в нашей повседневной жизни и отказаться от которой мы уже не в состоянии.

Успехов и удачи!

Список использованных источников

1 Баричев, С. Г. Криптография без секретов / С. Г. Баричев. – М. : Горячая Линия – Телеком, 2004. – 43 с.

2 Вишневский, А. Сетевые технологии Windows 2000 / А. Вишневский. – СПб. : Питер, 2000. – 591 с.

3 Домарёв, В. В. Защита информации и безопасность компьютерных систем / В. В. Домарёв. – Киев : Диалектика, 2004. – 670 с.

4 Касперски, К. Искусство дизассемблирования: наиболее полное руководство / К. Касперски, Е. Рокко. – СПб. : БХВ-Петербург, 2008. – 884 с.

5 Левин, М. Как стать хакером: интеллектуальное руководство по хакингу и фрикингу / М. Левин. – М. : Новый издательский дом, 2005. – 319 с.

6 Мак-Клар, С. Хакинг в Web. Атаки и защита / С. Мак-Клар, С. Шах, Ш. Шах ; пер. с англ.– М. : Вильямс, 2003. – 374 с.

7 Норткат, С. Обнаружение нарушений безопасности в сетях / С. Норткат, Д. Новак ; пер. с англ. – М. : Вильямс, 2003. – 447 с.

8 Олифер, В. Г. Компьютерные сети. Принципы технологии, протоколы / В. Г. Олифер, Н. А. Олифер. – СПб. : Питер, 2010. – 945 с.

9 Основы организацизационного обеспечения информационной безопасности объектов информатизации : учеб. пособие / С. Н. Семкин [и др.]. – М. : Гелиос АРВ, 2005. – 185 с.

10 Скляров, Д. В. Искусство защиты и взлома информации / Д. В. Скляров. – СПб. : БХВ-Петербург, 2004. – 276 с.

11 Столингс, В. Криптография и защита сетей: принципы и практика / В. Столингс. – М. : Вильямс, 2001. – 669 с.

12 Хогланд, Г. Взлом программного обеспечения: анализ и использование кода / Г. Хогланд, Г. Мак-Гроу. – М. : Вильямс, 2005. – 396 с.

13 Шнайер, Б. Секреты и ложь : безопасность данных в цифровом мире / Б. Шнайер. – СПб. : Питер, 2003. – 367 с.

14 Ярочкин, В. И. Информационная безопасность : учебник для вузов по гуманитарным и социально-экономическим специальностям / В. И. Ярочкин. – М. : Академический проект, 2005. – 543 с.

Св. план 2013, поз. 33

Учебное издание

Ганжа Виктор Александрович
Сидорик Валерий Владимирович
Чичко Ольга Ильинична

КОМПЬЮТЕРНЫЕ СЕТИ. ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ И СОХРАНЕНИЕ ИНФОРМАЦИИ

УЧЕБНО-МЕТОДИЧЕСКОЕ ПОСОБИЕ

Редактор *Е. И. Герман*
Корректор *Е. Н. Батурчик*
Компьютерная правка, оригинал-макет *А. А. Лысеня*

Подписано в печать 14.08.2014. Формат 60x84 1/16. Бумага офсетная. Гарнитура «Bookman».
Отпечатано на ризографе. Усл. печ. л. 7,44. Уч.-изд. л. 8,0. Тираж 200 экз. Заказ 269.

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий №1/238 от 24.03.2014,
№2/113 от 07.04.2014, №3/615 от 07.04.2014.
ЛП №02330/264 от 14.04.2014.
220013, Минск, П. Бровки, 6