МИНИСТЕРСТВО ВЫСШЕГО И СРЕДНЕГО СПЕЦИАЛЬНОГО ОБРАЗОВАНИЯ
РЕСПУБЛИКИ УЗБЕКИСТАН
ТАШКЕНТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ИМЕНИ АБУ РАЙХАНА БЕРУНИ

ФАКУЛЬТЕТ «ЭЛЕКТРОНИКА И АВТОМАТИКА»
КАФЕДРА «ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В УПРАВЛЕНИИ»

На правах рукописи

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

ТОШМАТОВОЙ Шохсанам Зокир кизи

на тему: «Программное обеспечение алгоритма RSA при шифровании потоков данных в системах управления» по направлению 5330200 - «Информатика и информационные технологии (в управлении)» для получения степени бакалавра

Зав.кафедрой к.т.н., доц. Севинов Ж.У.

Руководитель д.т.н., доц. Зарипов О.О.

Ташкент – 2016 г.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЯ

В настоящее время всеобщей компьютеризации благополучие и даже жизнь многих людей зависят от обеспечения информационной безопасности множества компьютерных систем обработки информации, а также контроля и управления различными техническими и технологическими объектами. К таким объектам можно отнести системы телекоммуникаций, банковские системы, атомные станции, системы управления динамическим объектом, а также системы обработки и хранения секретной и конфиденциальной информации. Для нормального и безопасного функционирования этих систем необходимо поддерживать их безопасность и целостность. В настоящее время для проникновения в чужие секреты и нанесения ущерба используется множество возможностей, а с появлением новых методов и средств защиты в компьютерных системах и сетях растет и уровень методов несанкционированного доступа, кражи информации для использования ее в корыстных целях или для нанесения вреда, что может быть обусловлено различными мотивами. Кроме того, информация может быть искажена или утеряна не только в результате несанкционированного доступа, но и по неосторожности или неосмотрительности пользователей информационных систем. Это требует разработки специализированных методов и средств обеспечения информационной безопасности.

Проблемы защиты информации в системах управления и обработки данных (СУОД) с использованием электронно-вычислительной техники постоянно находятся в центре внимания не только специалистов по разработке и использованию этих систем, но и широкого круга пользователей. Под защитой информации понимается использование специальных средств, методов и мероприятий с целью предотвращения утери или искажения информации, находящейся в СУОД.

Проблема защиты информации путем ее преобразования, исключающего ее прочтение посторонним лицом волновала человеческий ум с давних времен.

Бурное развитие криптографические системы получили в годы первой и второй мировых войн. Начиная с послевоенного времени и по нынешний день появление вычислительных средств ускорило разработку и совершенствование криптографических методов.

Проблема использования криптографических методов в информационных системах управления стала в настоящий момент особо актуальна потому что:

– с одной стороны, расширилось использование компьютерных сетей, в частности глобальной сети Internet, по которым передаются большие объемы информации государственного, военного, коммерческого и частного характера, не допускающего возможность доступа к ней посторонних лиц;

– другой стороны, появление новых мощных компьютеров, технологий сетевых и нейронных вычислений сделало возможным дискредитацию криптографических систем еще недавно считавшихся практически не раскрываемыми.

В настоящий момент существует огромное количество методов шифрования. Главным образом эти методы делятся, в зависимости от структуры используемых ключей и асимметричные методы. Существует много методов и способов защиты информации в системах управления и как вариант – ее шифрование, для чего могут использоваться специальные программы. Современные утилиты шифрования данных – это: специализированные программы, преобразовывающие открытые данные в зашифрованные или наоборот с использованием специальных ключей; программные средства, системы и комплексы, реализующие алгоритмы криптографического преобразования информации и предназначенные для защиты информации от несанкционированного доступа при ее передаче по каналам связи и (или) при ее обработке и хранении.

Таким образом, тема выпускной работы, посвященная изучению и разработке программного обеспечения алгоритмов шифровании потоков данных в системах управления, является актуальной.

ГЛАВА I. СТРУКТУРНАЯ СХЕМА ЗАЩИТЫ ИНФОРМАЦИИ В СИСТЕМАХ УПРАВЛЕНИЯ

1.1. Потенциальные угрозы безопасности информации

Для создания средств защиты информации необходимо определить природу угроз, формы и пути их возможного проявления и осуществления в автоматизированной системе управления (АСУ). Для решения поставленной задачи все многообразие угроз и путей их воздействия приведем к простейшим видам и формам, которые были бы адекватны их множеству в автоматизированной системе.

Случайные угрозы. Исследование опыта проектирования, изготовления, испытаний и эксплуатации автоматизированных систем говорят о том, что информация в процессе ввода, хранения, обработки, вывода и передачи подвергается различным случайным воздействиям.

Причинами таких воздействий могут быть: отказы и сбои аппаратуры, помехи на линии связи от воздействий внешней среды, ошибки человека как звена системы, системные и системотехнические ошибки разработчиков, структурные, алгоритмические и программные ошибки, аварийные ситуации и другие воздействия.

Частота отказов и сбоев аппаратуры увеличивается при выборе и проектировании системы, слабой в отношении надежности функционирования аппаратуры. Помехи на линии связи зависят от правильности выбора места размещения технических средств АСУ относительно друг друга и по отношению к аппаратуре соседних систем.

К ошибкам человека как звена системы следует относить ошибки человека как источника информации, человека-оператора, неправильные действия обслуживающего персонала и ошибки человека как звена, принимающего решения.

Ошибки человека могут подразделяться на логические (неправильно принятые решения), сенсорные (неправильное восприятие оператором информации) и оперативные, или моторные (неправильная реализация решения). Интенсивность ошибок человека может колебаться в широких пределах: от 1-2% до 15-40% и выше от общего числа операций при решении задачи.

К угрозам случайного характера следует отнести аварийные ситуации, которые могут возникнуть на объекте размещения автоматизированной системы. К аварийным ситуациям относятся: отказ от функционирования АСУ в целом, например выход из строя электропитания; стихийные бедствия: пожар, наводнение, землетрясение, ураганы, удары молнии и т.д.

Вероятность этих событий связана прежде всего с правильным выбором места размещения АСУ, включая географическое положение.

Преднамеренные угрозы. Преднамеренные угрозы связаны с действиями человека, причинами которых могут быть определенное недовольство своей жизненной ситуацией, сугубо материальный интерес или простое развлечение с целью самоутверждения своих способностей, как у хакеров и т.д.

Для вычислительных систем характерны следующие штатные каналы доступа к информации: терминалы пользователей; терминал администратора системы; терминал оператора функционального контроля; средства отображения информации; средства загрузки программного обеспечения; средства документирования информации; носители информации; внешние каналы связи.

Имея в виду, что при отсутствии защиты нарушитель может воспользоваться как штатными, так и другими физическими каналами доступа, назовем возможные каналы несанкционированного доступа (ВКНСД) в вычислительной системе, через которые возможно получить доступ к аппаратуре, ПО и осуществить хищение, разрушение, модификацию информации и ознакомление с нею: все перечисленные штатные средства при их использовании законными пользователями не по назначению и за пределами

своих полномочий; все перечисленные штатные средства при их использовании посторонними лицами; технологические пульты управления; внутренний монтаж аппаратуры; линии связи между аппаратными средствами данной вычислительной системы; побочное электромагнитное излучение аппаратуры системы; побочные наводки по сети электропитания и заземления аппаратуры; побочные наводки на вспомогательных и посторонних коммуникациях; отходы обработки информации в виде бумажных и магнитных носителей.

Очевидно, что при отсутствии законного пользователя, контроля и разграничения доступа к терминалу квалифицированный нарушитель легко воспользуется его функциональными возможностями для несанкционированного доступа к информации путем ввода соответствующих запросов и команд. При наличии свободного доступа в помещение можно визуально наблюдать информацию на средствах отображения и документирования, а на последних похитить бумажный носитель, снять лишнюю копию, а также похитить другие носители с информацией: листинги, магнитные ленты, диски и т.д.

Особую опасность представляет собой бесконтрольная загрузка программного обеспечения в ЭВМ, в которой могут быть изменены данные, алгоритмы или введена программа типа “троянский конь”, выполняющая дополнительные незаконные действия: запись информации на посторонний носитель, передачу в каналы связи другого абонента вычислительной сети, внесение в систему компьютерного вируса и т.д.

Опасной является ситуация, когда нарушителем является пользователь системы, который по своим функциональным обязанностям имеет законный доступ к одной части информации, а обращается к другой за пределами своих полномочий. Со стороны законного пользователя существует много способов нарушить работу вычислительной системы, злоупотреблять ею, извлекать, модифицировать или уничтожать информацию. Свободный доступ позволит ему обращаться к чужим файлам и банкам данных и изменять их случайно или преднамеренно. При техническом обслуживании (профилактике и ремонте)

аппаратуры могут быть обнаружены остатки информации на магнитной ленте, поверхностях дисков и других носителях информации. Обычное стирание информации не всегда эффективно. Ее остатки могут быть легко прочитаны. При транспортировке носителя по неохраняемой территории существует опасность его перехвата и последующего ознакомления посторонних лиц с секретной информацией.

Не имеет смысла создание системы контроля и разграничения доступа к информации на программном уровне, если не контролируется доступ к пульту управления ЭВМ, внутреннему монтажу аппаратуры, кабельным соединениям. Срабатывание логических элементов обусловлено высокочастотным изменением уровней напряжений и токов, что приводит к возникновению в эфире, цепях питания и заземления, а также в параллельно расположенных цепях и индуктивностях посторонней аппаратуры, электромагнитных полей и наводок, несущих в амплитуде, фазе и частоте своих колебаний признаки обрабатываемой информации. С уменьшением расстояния между приемником нарушителя и аппаратными средствами вероятность приема сигналов такого рода увеличивается. Непосредственное подключение нарушителем приемной аппаратуры и специальных датчиков к цепям электропитания и заземления, к каналам связи также позволяет совершить несанкционированное ознакомление с информацией, а несанкционированное подключение к каналам связи передающей аппаратуры может привести и к модификации информации. За последнее время в разных странах проведено большое количество исследовательских работ с целью обнаружения потенциальных каналов несанкционированного доступа к информации в вычислительных сетях. При этом рассматриваются не только возможности нарушителя, получившего законный доступ к сетевому оборудованию, но и воздействия, обусловленные ошибками программного обеспечения или свойствами используемых сетевых протоколов. Несмотря на то, что изучение каналов НСД продолжается до сих пор, уже в начале 80-х годов были сформулированы пять основных категорий угроз безопасности данных в вычислительных сетях: раскрытие содержания

передаваемых сообщений; анализ трафика, позволяющий определить принадлежность отправителя и получателя данных к одной из групп пользователей сети, связанных общей задачей; изменение потока сообщений, что может привести к нарушению режима работы какого-либо объекта, управляемого с удаленной ЭВМ; неправомерный отказ в предоставлении услуг; Несанкционированное установление соединения.

1.2. Средства защиты информации

Принято различать пять основных средств защиты информации: технические; программные; криптографические; организационные; законодательные.

Рассмотрим эти средства подробнее и оценим их возможности в плане дальнейшего их использования при проектировании конкретных средств защиты информации в ЛВС. Общая схема способов и средства защиты информации в системах обработки данных приведена на рис.1.1.

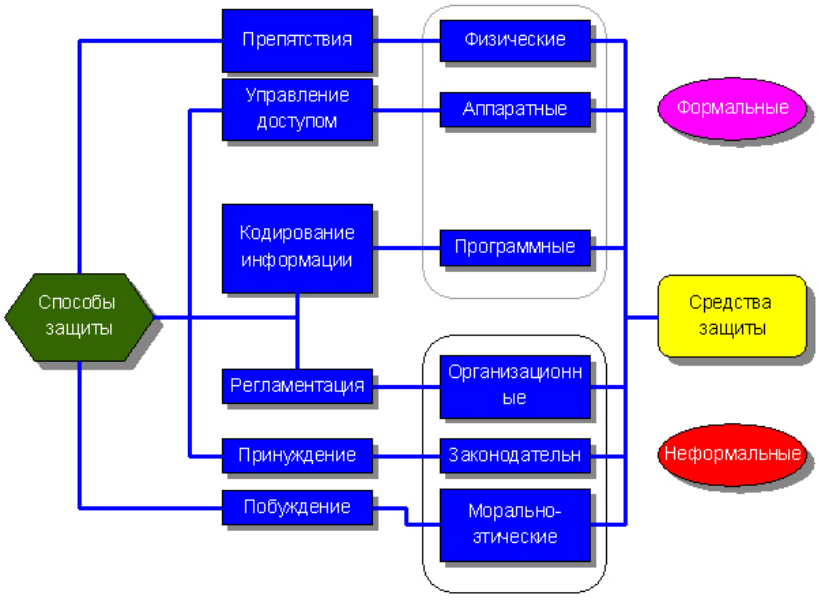


Рис.1.1. Способы и средства защиты информации в системах управления и обработки данных

Технические средства защиты информации. Технические средства защиты – это механические, электромеханические, оптические, радио,

радиолокационные, электронные и другие устройства и системы, способные выполнять самостоятельно или в комплексе с другими средствами функции защиты данных.

Технические средства защиты делятся на физические и аппаратные. К физическим средствам относятся замки, решетки, охранные сигнализации, оборудование контрольно-пропускных пунктов и др.; к аппаратным – замки, блокировки и системы сигнализации о вскрытии, которые применяются на средствах вычислительной техники и передачи данных.

Программные средства защиты информации. Программные средства защиты – это специальные программы, включаемые в состав программного обеспечения системы, для обеспечения самостоятельно или в комплексе с другими средствами, функций защиты данных.

По функциональному назначению программные средства можно разделить на следующие группы:

1. Программные средства идентификации и аутентификации пользователей.

Идентификация – это присвоение какому-либо объекту или субъекту уникального образа, имени или числа. Установление подлинности (аутентификация) заключается в проверке, является ли проверяемый объект (субъект) тем, за кого себя выдает. В качестве синонима слова "аутентификация" иногда используют сочетание "проверка подлинности".

Конечная цель идентификации и установления подлинности объекта в вычислительной системе – допуск его к информации ограниченного пользования в случае положительного результата проверки или отказ в допуске в противном случае.

Одним из распространенных методов аутентификации является присвоение лицу уникального имени или числа – пароля, – и хранение его значения в вычислительной системе. При входе в систему пользователь вводит свой пароль, вычислительная система сравнивает его значение со значением, хранящимся в своей памяти, и при совпадении открывает доступ к разрешенной

функциональной задаче, а при несовпадении – отказывает в нем. Наиболее высокий уровень безопасности входа в систему достигается разделением кода пароля на две части: одну, запоминаемую пользователем и вводимую вручную, и вторую, размещаемую на специальном носителе – карточке, устанавливаемой пользователем на специальное считывающее устройство, связанное с терминалом.

2. Средства идентификации и установления подлинности технических средств. Такие средства являются дополнительным уровнем защиты по отношению к паролям пользователей.

В ЭВМ хранится список паролей и другая информация о пользователях, которым разрешено пользоваться определенными терминалами, а также таблица ресурсов, доступных с определенного терминала конкретному пользователю.

3. Средства обеспечения защиты файлов.

Вся информация в системе, хранимая в виде файлов, делится на некоторое количество категорий по различным признакам, выбор которых зависит от функций, выполняемых системой. Наиболее часто можно встретить разделение информации: по степени важности; по степени секретности; по выполняемым функциям пользователей; по наименованию документов; по видам документов; по видам данных; по наименованию томов, файлов, массивов, записей; по имени пользователя; по функциям обработки информации: чтению, записи, исполнению; по областям оперативной и долговременной памяти; по времени и т.д.

Доступ должностных лиц к файлам осуществляется в соответствии с их функциональными обязанностями и полномочиями.

4. Средства защиты операционной системы и программ пользователей.

Защита операционной системы – наиболее приоритетная задача. Осуществляется запретом доступа в области памяти, в которых размещается операционная система.

Для защиты пользовательских программ применяется ограничение доступа к занимаемым этими программами областям памяти.

5. Вспомогательные средства.

К вспомогательным средствам программной защиты информации относятся: программные средства контроля правильности работы пользователей; программные уничтожители остатков информации; программы контроля работы механизма защиты; программы регистрации обращений к системе и выполнения действий с ресурсами; программы формирования и печати грифа секретности; программные средства защиты от компьютерных вирусов и др.

Криптографические средства защиты информации. Криптографические средства защиты – это методы специального шифрования данных, в результате которого их содержание становится недоступным без применения некоторой специальной информации и обратного преобразования.

Суть криптографической защиты заключается в преобразовании составных частей информации (слов, букв, слогов, цифр) с помощью специальных алгоритмов, либо аппаратных решений и кодов ключей, т.е. приведении ее к неявному виду. Для ознакомления с закрытой информацией применяется обратный процесс: декодирование (дешифровка). Использование криптографии является одним из распространенных методов, значительно повышающих безопасность передачи данных в сетях ЭВМ, данных, хранящихся в удаленных устройствах памяти, и при обмене информацией между удаленными объектами.

Организационные средства защиты информации. Организационные средства защиты – специальные организационно-технические и организационно-правовые мероприятия, акты и правила, осуществляемые в процессе создания и эксплуатации системы для организации и обеспечения защиты информации.

Законодательные средства защиты информации. Законодательные средства защиты – это законодательные акты, которые регламентируют

правила использования и обработки информации, и устанавливают ответственность и санкции за нарушение этих правил [6].

Законодательные меры по защите информации от НСД заключаются в исполнении существующих в стране или введении новых законов, постановлений, положений и инструкций, регулирующих юридическую ответственность должностных лиц – пользователей и обслуживающего персонала, – за утечку, потерю или модификацию доверенной ему информации, подлежащей защите, в том числе за попытку преднамеренного несанкционированного доступа к аппаратуре и информации. Таким образом цель законодательных мер – предупреждение и сдерживание потенциальных нарушителей.

1.3. Шифрование трафика сети

Обзор и классификация методов шифрования информации. Для преобразования (шифрования) информации обычно используется некоторый алгоритм или устройство, реализующее заданный алгоритм, которые могут быть известны широкому кругу лиц. Управление процессом шифрования осуществляется с помощью периодически меняющегося кода ключа, обеспечивающего каждый раз оригинальное представление информации при использовании одного и того же алгоритма или устройства. Знание ключа позволяет просто и надежно расшифровать текст. Однако, без знания ключа эта процедура может быть практически невыполнима даже при известном алгоритме шифрования.

Даже простое преобразование информации является весьма эффективным средством, дающим возможность скрыть ее смысл от большинства неквалифицированных нарушителей. Структурная схема шифрования информации представлена на рис.1.2.

Большинство из нас постоянно используют шифрование, хотя и не всегда знают об этом. К примеру, операционная система Microsoft Windows, хранит о пользователях (как минимум) следующую секретную информацию:

- пароли для доступа к сетевым ресурсам (домен, принтер, компьютеры в сети и т.п.);

- пароли для доступа в Интернет с помощью DialUp;

- кэш паролей (в браузере существует функция кэширования паролей, и Windows сохраняет все когда-либо вводимые в Интернете пароли);

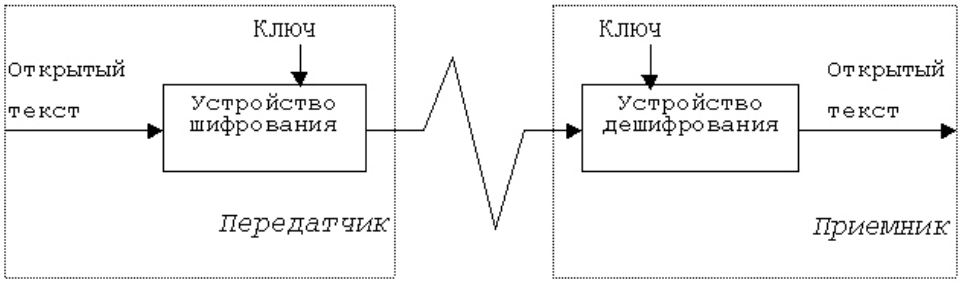


Рис.1.2. Шифрование информации

- сертификаты для доступа к сетевым ресурсам и зашифрованным данным на самом компьютере.

Эти данные хранятся либо в pwl-файле (в Windows 95), либо в SAM-файле (в Windows NT/2000/XP). Это файл Реестра Windows, и потому операционная система никому не даст к нему доступа даже на чтение. Злоумышленник может скопировать такие файлы, только загрузившись в другую ОС или с дискеты. Утилит для их взлома достаточно много, самые современные из них способны подобрать ключ за несколько часов.

Требования к криптосистемам. Процесс криптографического закрытия данных может осуществляться как программно, так и аппаратно. Аппаратная реализация отличается существенно большей стоимостью, однако ей присущи и преимущества: высокая производительность, простота, защищенность и т.д. Программная реализация более практична, допускает гибкость в использовании.

Для современных криптографических систем защиты информации сформулирована следующая система требований:

- зашифрованный текст должен поддаваться чтению только при наличии ключа шифрования;

- число операций для определения использованного ключа шифрования по фрагменту шифрованного текста и соответствующему ему открытого текста, должно быть не меньше общего числа возможных ключей;

- число операций, необходимых для расшифровывания информации путем перебора всех возможных ключей должно выходить за пределы возможностей современных компьютеров (с учетом возможности использования сетевых вычислений);

- знание алгоритма шифрования не должно влиять на надежность защиты;

- незначительные изменения ключа шифрования должны приводить к существенному изменению вида зашифрованного текста;

- незначительные изменения шифруемого текста должны приводить к существенному изменению вида зашифрованного текста даже при использовании одного и того же ключа;

- длина шифрованного текста должна быть равна длине исходного текста;

- не должно быть простых и легко устанавливаемых зависимостей между ключами, последовательно используемыми в процессе шифрования;

- любой ключ из множества возможных ключей должен обеспечивать надежную защиту информации;

- алгоритм должен допускать как программную, так и аппаратную реализацию.

Системы шифрования с секретным и открытым ключом. Криптографическая система представляет собой семейство T преобразований открытого текста. Члены этого семейства индексируются, или обозначаются символом k; параметр k является ключом. Пространство ключей K - это набор

возможных значений ключа. Обычно ключ представляет собой последовательный ряд букв алфавита.

Под Криптостойкостью понимают характеристику шифра, определяющую его стойкость к дешифрованию без знания ключа (т.е. криптоанализу). Имеется несколько показателей криптостойкости, среди которых:

- количество всех возможных ключей;

- среднее время, необходимое для криптоанализа.

Преобразование T_k определяется соответствующим алгоритмом и значением параметра k. Эффективность шифрования с целью защиты информации зависит от сохранения тайны ключа и криптостойкости шифра.

Современные широко применяемые методы шифрования можно разделить на два наиболее общих типа: с секретным ключом и с открытым ключом.

Шифрование с секретным ключом симметрично – ключ, с помощью которого текст шифруется, применяется и для его дешифровки. Оставаясь в рамках симметричной системы, необходимо иметь надежный канал связи для передачи секретного ключа. Но такой канал не всегда бывает доступен, и потому американские математики Диффи, Хеллман и Меркле разработали в 1976 г. концепцию открытого ключа и асимметричного шифрования.

Шифрование с открытыми ключами осуществляется с помощью двух ключей. Открытый ключ не является секретным – его распространение не оказывает никакого влияния на криптостойкость системы. Другой ключ, секретный, служит для дешифрования текстов, шифруемых с помощью открытого ключа.

Криптографические системы с открытым ключом используют необратимые или односторонние функции, для которых при заданном значении x относительно просто вычислить значение f(x), однако если y = f(x), то нет простого пути для вычисления значения x. Другими словами, чрезвычайно трудно рассчитать значение обратной функции.

В асимметричных системах должно удовлетворяться следующее требование: нет такого алгоритма (или он пока неизвестен), который бы из криптотекста и открытого ключа выводил исходный текст.

На практике криптографические системы с секретными ключами, как правило, быстрее систем с открытыми ключами, обеспечивающими ту же степень защиты.

Основные современные методы шифрования. Среди разнообразнейших способов шифрования можно выделить следующие основные методы:

- Алгоритмы замены или подстановки – символы исходного текста заменяются на символы другого (или того же) алфавита в соответствии с заранее определенной схемой, которая и будет ключом данного шифра. Отдельно этот метод в современных криптосистемах практически не используется из-за чрезвычайно низкой криптостойкости.

- Алгоритмы перестановки – символы оригинального текста меняются местами по определенному принципу, являющемуся секретным ключом. Алгоритм перестановки сам по себе обладает низкой криптостойкостью, но входит в качестве элемента в очень многие современные криптосистемы.

- Алгоритмы гаммирования – символы исходного текста складываются с символами некой случайной последовательности.

- Алгоритмы, основанные на сложных математических преобразованиях исходного текста по некоторой формуле. Многие из них используют нерешенные математические задачи. Например, широко используемый в Интернете алгоритм шифрования RSA основан на свойствах простых чисел.

- Комбинированные методы. Последовательное шифрование исходного текста с помощью двух и более методов.

Рассмотрим подробнее алгоритмы, построенные на сложных математических преобразованиях и комбинированные методы, как наиболее часто используемые для защиты данных в современных информационных системах.

Алгоритмы, основанные на сложных математических преобразованиях.

Алгоритм RSA. Алгоритм RSA (по первым буквам фамилий его создателей Rivest – Shamir - Adleman) основан на свойствах простых чисел (причем очень больших). Простыми называются такие числа, которые не имеют делителей, кроме самих себя и единицы. А взаимно простыми называются числа, не имеющие общих делителей, кроме 1.

Для начала необходимо выбрать два очень больших простых числа (большие исходные числа нужны для построения больших криптостойких ключей. Например, Unix-программа ssh-keygen по умолчанию генерирует ключи длиной 1024 бита). Как результат перемножения p и q определяется параметр n. Затем выбирается случайное число e, причем оно должно быть взаимно простым с числом φ(n) = (p - 1)*(q - 1). Отыскивается такое число d, для которого верно соотношение

$$(e*d) \bmod \varphi(n) = 1.$$

Mod – остаток от деления, т. е. если e, умноженное на d, поделить φ(n), то в остатке должно получиться 1. Другими словами, числа (e*d - 1) и φ(n) должны делиться нацело.

Открытым ключом является пара чисел e и n, а закрытым – d и n. При шифровании исходный текст рассматривается как числовой ряд, и над каждым его числом, которое должно быть меньше n, совершается операция

$$C(i) = (M(i)^e) \bmod n.$$

В результате получается последовательность C(i), которая и составит криптотекст. Декодирование информации происходит по формуле

$$M(i) = (C(i)^d) \bmod n.$$

Как видно, расшифровка предполагает знание секретного ключа.

Рассмотрим пример на маленьких числах. Пусть p = 3, q = 7. Тогда n = p*q = 21. Выберем e = 5. Из формулы (d*5) mod 12 = 1 вычисляем d = 17. Следовательно, открытый ключ 17, 21, секретный – 5, 21.

Зашифруем последовательность «2345»:

$$C_1 = 2^{17} \bmod 21 = 11;$$

$$C_2 = 3^{17} \bmod 21 = 12;$$

$$C_3 = 4^{17} \bmod 21 = 16;$$

$$C_4 = 5^{17} \bmod 21 = 17.$$

Криптотекст – 11 12 16 17. Проверим расшифровкой:

$$M_1 = 11^5 \bmod 21 = 2;$$

$$M_2 = 12^5 \bmod 21 = 3;$$

$$M_3 = 16^5 \bmod 21 = 4;$$

$$M_4 = 17^5 \bmod 21 = 5.$$

Как видно, результат совпал с изначальным открытым текстом.

Криптосистема RSA широко применяется в Интернете. Когда пользователи подсоединяются к защищенному серверу по протоколу SSL, устанавливает на свой ПК сертификат WebMoney либо подключается к удаленному серверу с помощью Open SSH или SecureShell, большинство даже не подозревает, что все эти программы применяют шифрование открытым ключом с использованием идей алгоритма RSA.

С момента своего создания RSA постоянно подвергалась атакам типа brute-force attack (атака методом грубой силы). В 1978 г. авторы алгоритма опубликовали статью, где привели строку, зашифрованную только что изобретенным ими методом. Первому, кто расшифрует сообщение, было назначено вознаграждение в размере 100 долларов, но для этого требовалось разложить на два сомножителя 129-значное число. Это был первый конкурс на взлом RSA. Задачу решили только через 17 лет после публикации статьи.

Криптостойкость RSA основывается на том предположении, что исключительно трудно, если вообще реально, определить закрытый ключ из открытого. Для этого требовалось решить задачу о существовании делителей огромного целого числа. До сих пор ее аналитическими методами никто не решил, и алгоритм RSA можно взломать лишь путем полного перебора.

Вероятностное шифрование. Одной из разновидностей криптосистем с открытым ключом является вероятностное шифрование, разработанное Шафи Гольвассером и Сильвио Минелли. Его суть состоит в том, чтобы алгоритм шифрования E подчинить вероятностным моделям. В чем же преимущества такого подхода? Для примера, в системе RSA не «маскируются» 0 и 1. Эту проблему успешно решают вероятностные алгоритмы, поскольку они ставят в соответствие открытому тексту M не просто криптотекст C, а некоторый элемент из множества криптотекстов CM. При этом каждый элемент этого множества выбирается с некоторой вероятностью. Другими словами, для любого открытого текста M результат работы алгоритма E будет случайной величиной. Может показаться, что в этом

случае дешифровать информацию будет невозможно, но это совсем не так. Для того чтобы сделать возможной дешифровку, нужно, чтобы для разных открытых текстов M_1 и M_2 множества CM_1 и CM_2 не пересекались. Также хочется сказать, что вероятностные алгоритмы шифрования являются более надежными, нежели детерминированные. В этой области наиболее распространены вероятностное шифрование на основе RSA-функций и криптосистема Эль-Гамала.

Комбинированные методы шифрования. Одним из важнейших требований, предъявляемых к системе шифрования, является ее высокая криптостойкость. Однако ее повышение для любого метода шифрования приводит, как правило, к существенному усложнению самого процесса шифрования и увеличению затрат ресурсов (времени, аппаратных средств, уменьшению пропускной способности и т.п.), и как следствие – времени работы криптографических систем.

Достаточно эффективным средством повышения стойкости шифрования является комбинированное использование нескольких различных способов шифрования, т.е. последовательное шифрование исходного текста с помощью двух или более методов.

Как показали исследования, стойкость комбинированного шифрования не ниже произведения стойкостей используемых способов.

Строго говоря, комбинировать можно любые методы шифрования и в любом количестве, однако на практике наибольшее распространение получили следующие комбинации:

1) подстановка + гаммирование;
2) перестановка + гаммирование;
3) гаммирование + гаммирование;
4) подстановка + перестановка.

Типичным примером комбинированного шифра является национальный стандарт США криптографического закрытия данных (DES).

Криптографический стандарт DES. В 1973 г. Национальное бюро стандартов США начало разработку программы по созданию стандарта шифрования данных на ЭВМ. Был объявлен конкурс среди фирм-разработчиков, который выиграла фирма IBM, представившая в 1974 году

алгоритм шифрования, известный под названием DES (Data Encryption Standart).

В этом алгоритме входные 64-битовые векторы, называемые блоками открытого текста, преобразуются в выходные 64-битовые векторы, называемые блоками шифротекста, с помощью двоичного 56-битового ключа К. Число различных ключей DES-алгоритма равно 2^{56}.

Алгоритм реализуется в течение 16 аналогичных циклов шифрования, где на i-ом цикле используется цикловой ключ K_i, представляющий собой алгоритмически вырабатываемую выборку 48 из 56 битов ключа K_i, $i = 1,2,\ldots,16$.

Алгоритм обеспечивает высокую стойкость, однако недавние результаты показали, что современная технология позволяет создать вычислительное устройство стоимостью около 1 млн. долларов США, способное вскрыть секретный ключ с помощью полного перебора в среднем за 3,5 часа.

Из-за небольшого размера ключа было принято решение использовать DES-алгоритм для закрытия коммерческой информации. Практическая реализация перебора всех ключей в данных условиях экономически не целесообразна, так как затраты на реализацию перебора не соответствуют ценности информации, закрываемой шифром.

DES-алгоритм явился первым примером широкого производства и внедрения технических средств в области защиты информации. Национальное бюро стандартов США проводит проверку аппаратных реализаций DES-алгоритма, предложенных фирмами-разработчиками, на специальном тестирующем стенде. Только после положительных результатов проверки производитель получает от Национального бюро стандартов сертификат на право реализации своего продукта. К настоящему времени аттестовано несколько десятков изделий, выполненных на различной элементной базе.

Достигнута высокая скорость шифрования. Она составляет в лучших изделиях 45 Мбит/с. Цена некоторых аппаратных изделий не превышает 100 долларов США.

Основные области применения DES-алгоритма:

1) хранение данных на компьютерах (шифрование файлов, паролей);

2) аутентификация сообщений (имея сообщение и контрольную группу, несложно убедиться в подлинности сообщения;

3) электронная система платежей (при операциях с широкой клиентурой и между банками);

4) Электронный обмен коммерческой информацией (обмен данными между покупателями, продавцом и банкиром защищен от изменений и перехвата.

Позднее появилась модификация DES – Triple DES («тройной DES» – так как трижды шифрует информацию «обычным» алгоритмом DES), свободная от основного недостатка прежнего варианта – короткого ключа; он здесь в два раза длиннее. Но, как оказалось, Triple DES унаследовал другие слабые стороны своего предшественника: отсутствие возможности для параллельных вычислений при шифровании и низкую скорость.

ГОСТ 28147-89.

Разработка была принята и зарегистрирована как ГОСТ 28147-89. Алгоритм был введен в действие в 1990 году. И хотя масштабы применения этого алгоритма шифрования до сих пор уточняются, начало его внедрения, в частности в банковской системе, уже положено. Алгоритм несколько медлителен, но обладает весьма высокой криптостойкостью.

В общих чертах ГОСТ 28147-89 аналогичен DES. Блок-схема алгоритма ГОСТ отличается от блок-схемы DES-алгоритма лишь отсутствием начальной перестановки и числом циклов шифрования (32 в ГОСТ против 16 в DES-алгоритме).

Ключ алгоритма ГОСТ – это массив, состоящий из 32-мерных векторов X_1, X_2,…X_8. Цикловой ключ i-го цикла K_i равен Xs,где ряду значенийi от 1 до 32 соответствует следующий ряд значений s:

1,2,3,4,5,6,7,8,1,2,3,4,5,6,7,8,1,2,3,4,5,6,7,8,8,7,6,5,4,3,2,1.

В шифре ГОСТ используется 256-битовый ключ и объем ключевого пространства составляет 2^{256}. Ни на одной из существующих в настоящее время или предполагаемых к реализации в недалеком будущемкомпьютерных систем общего применения нельзя подобрать ключ за время, меньшее многих сотен лет. Российский стандарт проектировался с большим запасом, по стойкости он на много порядков превосходит американский стандарт DES с его реальным размером ключа в 56 бит и объемом ключевого пространства всего 2^{56}, чего явно недостаточно. Ключ криптоалгоритма ГОСТ длиной 32 байта (256 бит) вчетверо больше ключа DES. Необходимое же на перебор всех ключей время при этом возрастает не в четыре раза, а в $256^{32-8} = 256^{24}$, что выливается уже в астрономические цифры). В этой связи DES может представлять скорее исследовательский или научный, чем практический интерес.

Выводы об использовании современных алгоритмов шифрования. В настоящее время наиболее часто применяются три основных стандарта шифрования: DES; ГОСТ 28147-89 – отечественный метод, отличающийся высокой криптостойкостью; RSA – система, в которой шифрование и расшифровка осуществляется с помощью разных ключей.

Недостатком RSA является довольно низкая скорость шифрования, зато она обеспечивает персональную электронную подпись, основанную на уникальном для каждого пользователя секретном ключе. Характеристики наиболее популярных методов шифрования приведены в таблице 1.1.

Таблица 1.1.

Характеристики наиболее распространенных методов шифрования

Алгоритм	*DES*	*ГОСТ 28147-89*	*RSA*
Длина ключа	56 бит	256 бит	300-600 бит
Скорость шифрования	10-200 Кбайт/с	50-70 Кбайт/с	300-500 бит/с
Криптостойкость операций	1017	1017	1023

Реализация	Программная и аппаратная	В основном аппаратная	Программная и аппаратная

1.4. Структура системы защиты информации

На основе принятой концепции средства защиты информации делятся на средства защиты от преднамеренного НСД (СЗИ ПНСД) и от случайного НСД (СЗИ СНСД). Средства управления защитой информации (СУЗИ) от НСД являются объединяющими, дающими возможность с помощью целенаправленных и взаимосвязанных функций в сочетании с наиболее полным охватом возможных каналов НСД объекта отдельными средствами защиты создать законченную и строгую систему защиты в комплексе средств автоматизации.

СЗИ ПНСД включает 1-й контур защиты – систему контроля доступа на территорию объекта (СКДТО), 2-й контур защиты – систему контроля и разграничения доступа в помещение (СКРДП) и основной контур защиты (ОКЗ). СКДТО, содержащая систему охранной сигнализации (СОС) и контрольно-пропускные пункты (КПП), служит для ограничения доступа лиц на территорию объекта, а также совместно со специальными аппаратными решениями составляет средство защиты от побочных электромагнитных излучений и наводок (ПЭМИН).

Основной контур защиты перекрывает каналы доступа по периметру комплекса средств автоматизации (КСА). Система контроля вскрытия аппаратуры (СКВА) перекрывает доступ к внутреннему монтажу, технологическим пультам управления и кабельным соединениям.

Система опознания и разграничения доступа к информации (СОРДИ) закрывает несанкционированный доступ и обеспечивает возможность контроля санкционированного доступа к информации законных пользователей и разграничения их полномочий с учетом их функциональных обязанностей.

Средства вывода аппаратуры из рабочего контура (СВАРК) обеспечивают блокировку НСД к информации при ремонте и профилактике аппаратуры. В числе средств основного контура применяются также средства защиты ресурсов (СЗР) и организационные меры. СЗР нацелены на недопущение блокировки пользователем-нарушителем работы других пользователей, а также для контроля и ограничения доступа пользователей к ресурсам.

Средства защиты информации на носителях (СЗИН) включают средства шифрования данных (СШД), средства уничтожения остатков информации на носителях (СУОИ), средства аутентификации информации на носителях (САИН), средства верификации программного обеспечения (СВПО) и организационно-технические мероприятия. Система контроля вскрытия аппаратуры включает датчики вскрытия, установленные на контролируемой аппаратуре, цепи сбора сигналов (ЦСС) и устройство контроля вскрытия аппаратуры (УКВА).

СОРДИ содержит терминал службы безопасности информации (ТСБИ), функциональные задачи программного обеспечения (ФЗ ПО), реализующие на программном уровне идентификацию и аутентификацию пользователей, а также разграничение их полномочий по доступу к информации. В целях защиты кодов паролей от НСД для них также должны быть предусмотрены средства защиты (СЗКП).

Средства защиты от случайного НСД включают средства повышения достоверности информации (СПДИ) и средства защиты информации от аварийных ситуаций (СЗИ АС). СПДИ содержат систему функционального контроля (СФК), устройство защиты от ошибок в каналах связи (УЗО КС), средства контроля целостности программного обеспечения (СКЦ ПО) и специальные технические решения (СТР). Они включают средства защиты от переадресации памяти (СЗПП), изоляции функциональных задач (СИФЗ) и другие технические решения.

Средства управления защитой информации содержат автоматизированное рабочее место службы безопасности (АРМ СБ) информации, ФЗ ПО, специально разработанные для выполнения управления защитой на

программном уровне, включая ведение журнала учета и регистрации доступа (ЖУРД) и организационные мероприятия. АРМ СБ включает терминал безопасности, УКВА, аппаратуру записи кодов в физические ключи-пароли (АЗКП), необходимое количество ключей-паролей и аппаратуру регистрации и документирования информации (АРДИ). В дополнение к указанным средствам, выполненным на аппаратном и программном уровнях, в необходимых случаях применяются организационные меры.

1.5. Требования к защите информации

Чтобы обеспечить требуемый уровень безопасности информации в системах управления подразделения, система безопасности должна иметь следующие средства:

- средства идентификации и проверки полномочий;
- средства обеспечения защиты файлов;
- средства защиты ОС и программ пользователей;
- средства шифрования/дешифрования трафика сети;
- средства уничтожения остатков информации в системе;
- средства регистрации обращений к системе.

ГЛАВА II. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ АЛГОРИТМА RSA ПРИ ШИФРОВАНИИ ПОТОКОВ ДАННЫХ В СИСТЕМАХ УПРАВЛЕНИЯ

2.1. Шифрование больших сообщений и потоков данных в системах управления

В современных информационных системах начинают применяться технологии, которые требуют передачи существенно больших объемов данных. Среди таких технологий: факсимильная, видео и речевая связь; голосовая почта; системы видеоконференций.

Так как передача оцифрованной звуковой, графической и видеоинформации во многих случаях требует конфиденциальности, то возникает проблема шифрования огромных информационных массивов. Для интерактивных систем типа телеконференций, ведения аудио или видеосвязи, такое шифрование должно осуществляться в реальном масштабе времени и по возможности быть "прозрачным" для пользователей.

Это немыслимо без использования современных технологий шифрования. Наиболее распространенным является потоковое шифрование данных. Если в описанных ранее криптосистемах предполагалось, что на входе имеется некоторое конечное сообщение, к которому и применяется криптографический алгоритм, то в системах с потоковым шифрованием принцип другой.

Система защиты не ждет, когда закончится передаваемое сообщение, а сразу же осуществляет его шифрование и передачу. Наиболее очевидным является побитовое сложение входящей последовательности (сообщения) с некоторым бесконечным или периодическим ключом, получаемым от некого генератора. Примером стандарта потокового шифрования является RC4, разработанный Ривестом. Однако, технические подробности этого алгоритма держатся в секрете.

Другим, иногда более эффективным методом потокового шифрования является шифрование блоками, т.е. накапливается фиксированный объем информации (блок), а затем преобразованный некоторым криптографическим методом передается в канал связи.

Как было неоднократно отмечено, проблема распределения ключей является наиболее острой в крупных информационных системах. Отчасти эта проблема решается (а точнее снимается) за счет использования открытых ключей. Но наиболее надежные криптосистемы с открытым ключом типа RSA достаточно трудоемки, а для шифрования мультимедийных данных и вовсе не пригодны.

Оригинальные решения проблемы "блуждающих ключей" активно разрабатываются специалистами. Эти системы являются некоторым компромиссом между системами с открытыми ключами и обычными алгоритмами, для которых требуется наличие одного и того же ключа у отправителя и получателя.

Идея метода достаточно проста. После того, как ключ использован в одном сеансе по некоторому правилу он сменяется другим. Это правило должно быть известно и отправителю, и получателю. Зная правило, после получения очередного сообщения получатель тоже меняет ключ. Если правило смены ключей аккуратно соблюдается и отправителем и получателем, то в каждый момент времени они имеют одинаковый ключ. Постоянная смена ключа затрудняет раскрытие информации злоумышленником.

Основная задача в реализации этого метода - выбор эффективного правила смены ключей. Наиболее простой путь - генерация случайного списка ключей. Смена ключей осуществляется в порядке списка. Однако, очевидно список придется каким-то образом передавать.

Другой вариант - использование математических алгоритмов, основанных на так называемых перебирающих последовательностях. На множестве ключей путем одной и той же операции над элементом получается

другой элемент. Последовательность этих операций позволяет переходить от одного элемента к другому, пока не будет перебрано все множество.

Надежность таких методов должна быть обеспечена с учетом известности злоумышленнику используемого правила смены ключей.

Интересной и перспективной задачей является реализация метода "блуждающих ключей" не для двух абонентов, а для достаточно большой сети, когда сообщения пересылаются между всеми участниками.

Эти три вида преобразования информации используются в разных целях, что можно представить в таблице.

Вид преобразования	*Цель*	*Изменение объема информации после преобразования.*
Шифрование	передача конфиденциальной информации; обеспечение аутентификации и защиты от преднамеренных изменений;	обычно не изменяется, увеличивается лишь в цифровых сигнатурах и подписях
Помехоустойчивое кодирование	защита от искажения помехами в каналах связи	увеличивается
Сжатие (компрессия)	сокращение объема передаваемых или хранимых данных	уменьшается

Как видно эти три вида преобразования информации отчасти дополняют друг друга и их комплексное использование поможет эффективно использовать каналы связи для надежной защиты предаваемой информации.

Особенно интересным представляется возможность объединения методов кодирования и шифрования. Можно утверждать, что по сути кодирование - это элементарное шифрование, а шифрование - это элементарное помехоустойчивое кодирование.

Другая возможность - комбинирование алгоритмов шифрования и сжатия информации. Задача сжатия состоит в том, чтобы преобразовать сообщение в пределах одного и того же алфавита таким образом, чтобы его длина (количество букв алфавита) стала меньше, но при этом сообщение можно было восстановить без использования какой-то дополнительной информации. Наиболее популярные алгоритмы сжатия - RLE, коды Хаффмана, алгоритм Лемпеля-Зива. Для сжатия графической и видеоинформации используются алгоритмы JPEG и MPEG.

Главное достоинство алгоритмов сжатия с точки зрения криптографии состоит в том, что они изменяют статистику входного текста в сторону ее выравнивания. Так, в обычном тексте, сжатом с помощью эффективного алгоритма все символы имеют одинаковые частотные характеристики и даже использование простых системы шифрования сделают текст недоступным для криптоанализа.

Разработка и реализация таких универсальных методов - перспектива современных информационных систем.

При выборе системы для защиты данных, прежде всего, стоит обратить внимание на используемые алгоритмы шифрования.

Теоретически, приложив достаточно усилий, злоумышленник может взломать любую криптографическую систему. Вопрос заключается лишь в том, сколько работы ему необходимо для этого проделать. В принципе, фактически любую задачу по взлому криптографической системы количественно можно сравнить с поиском, выполняемым путём полного перебора всех возможных вариантов.

По мнению специалистов, на сегодняшний день любой современной криптографической системе вполне достаточно 128-битового уровня безопасности. Это означает, что для осуществления успешной атаки на такую систему потребуется, как минимум, 2128 шагов. Согласно закону Мура, адаптированного к криптографии, достаточно даже 110 или 100 бит, однако криптографических алгоритмов, рассчитанных на такие ключи, не существует.

Сам алгоритм должен быть максимально широко распространён. Никому неизвестные «самописные» алгоритмы не изучены специалистами в области криптографии и могут содержать опасные уязвимости.

Таким образом, достаточно надёжными могут быть признаны алгоритмы: ГОСТ, AES, Twofish, Serpent с длиной ключа 128, 192 или 256 бит.

Отдельного рассмотрения заслуживают асимметричные алгоритмы шифрования. В них для шифрования и расшифрования используются разные ключи (отсюда и их название). Эти ключи образуют пару и генерируются, как правило, самим пользователем. Для шифрования информации используется т.н. открытый ключ. Этот ключ общеизвестен и любой желающий может зашифровать адресуемое пользователю сообщение с его помощью. Закрытый ключ используется для расшифрования сообщения и известен только самому пользователю, который хранит его в секрете.

Общепринятым способом распространения и хранения открытых ключей пользователей является использование цифровых сертификатов формата X.509 Цифровой сертификат в простейшем случае – это своего рода электронный паспорт, который содержит информацию о пользователе (имя, идентификатор, адрес электронной почты и т.п.), информацию об открытом ключе клиента, об Удостоверяющем центре, изготовившем сертификат, серийный номер сертификата, срок действия и т.д.

Удостоверяющий Центр (УЦ) – это третья доверительная сторона, которая наделена высоким уровнем доверия пользователей и которая обеспечивает комплекс мероприятий для использования доверяющими сторонами сертификатов. Удостоверяющий центр – это компонент системы управления сертификатами, предназначенный для формирования электронных сертификатов подчиненных центров и конечных пользователей, удостоверенных электронно-цифровой подписью УЦ.

В простейшем случае используются т.н. самоподписанные сертификаты, когда пользователь сам выступает в роли своего удостоверяющего центра.

Общепризнано, что в случае использования асимметричных алгоритмов шифрования, эквивалентная 128-битному симметричному алгоритму стойкость достигается при использовании ключей длиной не менее 1024 бит. Это связано с особенностями математической реализации таких алгоритмов.

Помимо непосредственно алгоритмов шифрования стоит обратить внимание и на способ их реализации. Программно-аппаратный комплекс может иметь встроенные алгоритмы шифрования или использовать внешние подключаемые модули. Второй вариант предпочтительнее по трём причинам. Во-первых, вы сможете повышать уровень безопасности, в соответствии с растущими потребностями компании, используя более стойкие алгоритмы Опять же, в случае изменения требований политки безопасности (например, компании потребуется переход на сертифицированные криптопровайдеры) вы сможете оперативно заменить имеющиеся криптоалгоритмы, на требуемые без какой-либо существенной задержки или сбоя в работе. Понятно, что в случае встроенной реализации алгоритма это гораздо сложнее.

Второй плюс внешней реализации заключается в том, что такое шифросредство не подпадает под соответствующие законодательные ограничения по его распространению.

И, в-третьих, не стоит забывать и о том, что реализация алгоритма шифрования – далеко не тривиальная задача. Правильная реализация требует большого опыта. Скажем, ключ шифрования никогда не должен находиться в оперативной памяти компьютера в явном виде. В серьёзных продуктах этот ключ разделяется на несколько частей, при этом на каждую из них накладывается случайная маска. Все операции с ключом шифрования производятся по частям, а на итоговый результат накладывается обратная маска. Уверенности в том, что разработчик учёл все эти тонкости при самостоятельной реализации алгоритма шифрования, к сожалению, нет.

Ещё одним фактором, влияющим на степень защищённости Ваших данных, является принцип организации работы с ключами шифрования. Здесь есть несколько вариантов, и перед выбором конкретной системы шифрования

настоятельно рекомендуется поинтересоваться, как она устроена: где хранятся ключи шифрования, как они защищаются и т.д. К сожалению, зачастую сотрудники компании-разработчика не в состоянии объяснить даже базовых принципов работы их продукта. Особенно это замечание относится к т.н. sales-менеджерам. Простейшие вопросы нередко ставят их в тупик. Пользователю же, решившему защитить свою конфиденциальную информацию, желательно разобраться во всех тонкостях.

Для определённости будем называть ключ, используемый для шифрования данных мастер-ключом.

На сегодняшний день наиболее распространенными и часто используемыми являются следующие два подхода.

1. Мастер-ключ генерируется на основе некоторых входных данных. Этот мастер-ключ используется при шифровании данных. В дальнейшем для получения доступа к зашифрованной информации пользователь вновь предоставляет системе те же самые входные данные для генерации мастер-ключа. Сам мастер-ключ, таким образом, нигде не хранится.

Основным недостатком этого способа является невозможность создания резервной копии мастер-ключа. В качестве входных данных могут быть использованы: пароль, какой-либо файл, сохранённый на внешнем носителе и т.п. Утрата любого компонента входных данных ведёт к утрате доступа к вашей информации.

2. Мастер-ключ генерируется с использованием генератора случайных чисел. Затем он шифруется каким-либо алгоритмом и после этого сохраняется вместе с данными или же на внешнем носителе. Для получения доступа сначала расшифровывается мастер-ключ, а после этого – сами данные. Для шифрования мастер-ключа целесообразно использовать алгоритм такой же стойкости, что и для шифрования самих данных. Использование менее стойкого алгоритма снижает безопасность системы, а использование более стойкого - бессмысленно, т.к. безопасность не повышает.

В общем случае, надёжность криптографической системы определяется надёжностью самого слабого её звена. Злоумышленник всегда может атаковать наименее стойкий алгоритм из двух: алгоритм шифрования данных или алгоритм шифрования мастер-ключа.

Второй подход позволяет создавать резервные копии мастер-ключа, которые возможно использовать в дальнейшем для восстановления доступа к данным в случае каких-либо форс-мажорных обстоятельств.

Ключ, на котором осуществляется шифрование мастер-ключа, также получают на основе некоторых входных данных.

Вариант первый: парольный. Пользователь вводит некоторый пароль, на основе которого с использованием, например, хэш-функции генерируется ключ шифрования (см. схему №1). Фактически надёжность системы в этом случае определяется только сложностью и длиной пароля. Использование надёжных паролей неудобно: запомнить бессмысленный набор из 10-15 символов и вводить его каждый для получения доступа к данным не так просто. А если таких паролей несколько (допустим, для доступа к разным приложениям), то и вовсе невозможно. Парольная защита также подвержена атакам методом прямого перебора, а установленный клавиатурный шпион легко позволит злоумышленнику получить доступ к данным.

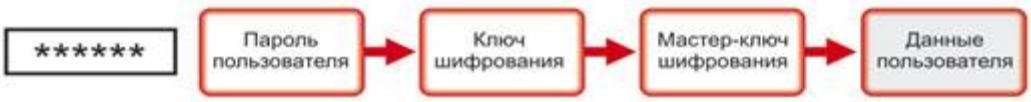


Схема №1. Шифрование мастер-ключа с использованием пароля

Вариант второй: внешнее хранение. На внешнем носителе размещаются некоторые данные, используемые для генерации ключа шифрования (схему №2). Простейшим вариантом является использование файла (т.н. ключевой файл), расположенного на дискете (CD-диске, USB-флэш памяти и т.п.) Этот способ надёжнее варианта с паролем. Для генерации используются не десяток символов пароля, а значительное количество данных, например, 64 или даже 128 байт.

В принципе, ключевой файл можно разместить и на жёстком диске компьютера, но значительно безопасней хранить его отдельно от данных.

Не рекомендуется в качестве ключевых файлов использовать файлы, создаваемые какими-либо общеизвестными приложениями (*.doc, *xls, *.pdf и т.д.) Их внутренняя структурированность может дать злоумышленнику дополнительную информацию. Например, все файлы, созданные архиватором WinRAR, начинаются с символов «Rar!» - это целых четыре байта.

Недостатком данного способа является возможность для злоумышленника легко скопировать файл и создать дубликат внешнего носителя. Таким образом, пользователь, даже на короткое время утративший контроль над этим носителем, фактически уже не может быть на 100% уверен в конфиденциальности своих данных.

В качестве внешнего носителя иногда применяются электронные USB-ключи или смарт-карты, но при этом данные, используемые для генерации ключа шифрования, просто сохраняются в памяти этих носителей и так же легко доступны для считывания.

Схема №2. Шифрование мастер-ключа с использованием данных с внешнего носителя (дискета, CD-диск).

Вариант третий: защищенное внешнее хранение. Данный способ во многом схож с предыдущим. Важным отличием является то, что для получения доступа к данным на внешнем носителе пользователь обязательно должен ввести PIN-код. В качестве внешнего носителя используются токены (электронные USB-ключи или смарт-карты). Данные, используемые для генерации ключа шифрования, размещаются в защищённой памяти токена и не могут быть прочитаны злоумышленником без знания им соответствующего PIN-кода (схему №3).

Утрата токена ещё не означает раскрытия самой информации. Для защиты от прямого подбора PIN-кода ставится аппаратная временная задержка между двумя последовательными попытками или аппаратное же ограничение на количество неправильных попыток ввода пин-кода (например, 15), после чего токен просто блокируется.

Поскольку токен может использоваться в разных приложениях, а PIN-код используется один и тот же, то можно обманным путём вынудить пользователя ввести свой PIN-код в подложной программе, после чего считать необходимые данные из закрытой области памяти токена. Некоторые приложения кэшируют значение PIN-кода в рамках одного сеанса работы, что так же несёт в себе определённый риск.

Схема №3. Шифрование мастер-ключа с использованием защищенного внешнего носителя (токен, смарт-карта).

Вариант четвёртый: смешанный. Возможен вариант, когда для генерации ключа шифрования одновременно используются пароль, ключевой файл на внешнем носителе и данные в защищённой памяти токена (схему №4).

Такой способ довольно сложен в повседневном использовании, поскольку требует от пользователя дополнительных действий. Многокомпонентная система также значительно сильнее подвержена рискам утраты доступа: достаточно потерять один из компонентов, и доступ без использования заранее созданной резервной копии уже не возможен.

Схема №4. Шифрование мастер-ключа с использованием нескольких компонентов

Вариант пятый: оптимальный. Отдельного рассмотрения заслуживает ещё один подход к организации безопасного хранения мастер-ключа, лишённый основных недостатков вышеперечисленных вариантов этот способ представляется наиболее оптимальным.

Современные токены (рис. №1) позволяют не только хранить в закрытой памяти данные, но также выполняют аппаратно целый ряд криптографических преобразований. Так, например, смарт-карты, а также USB-ключи, являющиеся полнофункциональными смарт-картами, а не их аналогами, реализуют асимметричные алгоритмы шифрования. Примечательно, что при этом пара открытый – закрытый ключ генерируется также аппаратно. Важно, что закрытый ключ на смарт-картах хранится как «write-only», т.е. он используется операционной системой смарт-карты для криптографических преобразований, но не может быть прочитан или скопирован пользователем. Фактически, пользователь сам не знает свой закрытый ключ – он только им обладает.

Данные, которые необходимо расшифровать, передаются операционной системе смарт-карты, аппаратно ей расшифровываются с помощью закрытого ключа и передаются обратно в расшифрованном виде. Все операции с закрытым ключом возможны только после ввода пользователем PIN-кода от смарт-карты.

Такой подход успешно используется во многих современных информационных системах для аутентификации пользователя. Применим он и для аутентификации пользователя при доступе к зашифрованной информации.

Мастер-ключ шифруется с помощью открытого ключа пользователя. Для получения доступа к данным пользователь предъявляет свою смарт-карту (или USB-ключ, являющийся полнофункциональной смарт-картой) и вводит пин-код от неё. Затем мастер-ключ аппаратно расшифровывается с помощью закрытого ключа, хранящегося на смарт-карте, и пользователь получает доступ к данным.

Такой подход (схему №5) сочетает в себе безопасность и удобство использования.

Схема №5. Шифрование мастер-ключа с использованием асимметричного алгоритма шифрования

В вариантах 1, 2, 3 и 4 очень важным является выбор способа генерации ключа шифрования на основе пароля и/или данных с внешнего носителя. Уровень безопасности (в криптографическом смысле), обеспечиваемый этим способом, должен быть не ниже, чем уровень безопасности остальных компонентов системы. Скажем, вариант, когда мастер-ключ просто хранится на внешнем носителе в инвертированном виде, крайне уязвим и небезопасен.

Современные токены поддерживают асимметричные алгоритмы с длиной ключа 1024 или 2048 бит, обеспечивая тем самым соответствие надёжности алгоритма шифрования мастер-ключа и надёжности алгоритма шифрования самих данных.

Аппаратное ограничение на количество неправильных попыток ввода PIN-кода нивелирует риск его подбора и позволяет использовать достаточно простой для запоминания PIN-код.

Использование одного устройства с несложным PIN-кодом повышает удобство без ущерба для безопасности.

Создать дубликат смарт-карты не может даже сам пользователь, т.к. нет возможности скопировать закрытый ключ. Это также позволяет без опасения использовать смарт-карту совместно с любыми другими программами.

Есть и ещё один критерий выбора, который зачастую остаётся без внимания, но при этом относится к разряду «критических». Речь идёт о качестве технической поддержки.

Не вызывает сомнения, что защищаемая информация является ценной. Быть может, её утрата принесёт меньший вред, чем публичное раскрытие, но определённое неудобство будет доставлено в любом случае.

Оплачивая продукт, Вы, в том числе, платите и за то, что он будет нормально функционировать, а в случае сбоя Вам оперативно помогут разобраться в проблеме и устранить её.

Основная сложность заключается в том, что заранее оценить качество техподдержки довольно сложно. Ведь существенную роль служба техподдержки начинает играть на поздних стадиях внедрения, на этапе опытной эксплуатации и после завершения внедрения, в процессе сопровождения системы.

Критериями качества технической поддержки можно считать: время реакции на запрос, полноту ответов и компетентность специалистов. Рассмотрим их чуть более подробно.

Зачастую эквивалентом качества работы службы технической поддержки считается скорость реакции на запрос. Тем не менее, оперативные, но неправильные рекомендации могут принести значительно больший вред, чем простое их отсутствие.

Списки частых вопросов (FAQ) могут стать источником дополнительной информации не только о самом продукте, но и о компетентности специалистов, работающих в компании. Например, отсутствие такого раздела наводит на мысли о непопулярности данного продукта или об отсутствии в организации отдельных специалистов, занимающихся техподдержкой, способных написать базу знаний по обращениям пользователей. Забавно, но на некоторых сайтах в ответах на частые вопросы встречаются ошибки, в том числе и в написании названия самого продукта.

В любом случае, прежде всего нужно трезво оценивать угрозы и критичность данных, а средства обеспечения безопасности желательно выбирать руководствуясь тем, насколько успешно они справляются со своей основной задачей – обеспечением защиты от несанкционированного доступа.

2.2. Алгоритм RSA

Вычислительные машины и электронные средства связи проникли практически во все сферы человеческой деятельности. Немыслима без них и современная криптография. Шифрование и дешифрование текстов можно представлять себе как процессы переработки целых чисел при помощи ЭВМ, а способы, которыми выполняются эти операции, как некоторые функции, определённые на множестве целых чисел. Всё это делает естественным появление в криптографии методов теории чисел. Кроме того, стойкость ряда современных криптосистем обосновывается только сложностью некоторых теоретико-числовых задач.

Но возможности ЭВМ имеют определённые границы. Приходится разбивать длинную цифровую последовательность на блоки ограниченной длины и шифровать каждый такой блок отдельно. Мы будем считать в дальнейшем, что все шифруемые целые числа неотрицательны и по величине меньше некоторого заданного (скажем, техническими ограничениями) числа m. Таким же условиям будут удовлетворять и числа, получаемые в процессе шифрования. Это позволяет считать и те, и другие числа элементами кольца

вычетов Z/mZ. Шифрующая функция при этом может рассматриваться как взаимнооднозначное отображение колец вычетов

$$f : \mathrm{Z}/m\mathrm{Z} \to \mathrm{Z}/m\mathrm{Z}$$

а число $f(x)$ представляет собой сообщение x в зашифрованном виде.

Простейший шифр такого рода - шифр замены, соответствует отображению $f : x \to x + k \pmod m$ при некотором фиксированном целом k. Подобный шифр использовал еще Юлий Цезарь. Конечно, не каждое отображение f подходит для целей надежного сокрытия информации.

Самым распространенным алгоритмом ассиметричного шифрования является алгоритм RSA. Он был предложен тремя исседователями-математиками Рональдом Ривестом (R.Rivest), Ади Шамиром (A.Shamir) и Леонардом Адльманом (L.Adleman) в 1977-78 годах. Разработчикам данного алгоритма удалось эффективно воплотить идею односторонних функций с секретом. Стойкость RSA базируется на сложности факторизации больших целых чисел. Современное состояние алгоритмов факторизации (разложения на множители) позволяет решать эту задачу для чисел длиной до 430 бит; исходя из этого, ключ длиной в 512 бит считается надежным для защиты данных на срок до 10 лет, а в 1024 бита – безусловно надежным. Несмотря на то, что отсутствует математически доказанное сведение задачи раскрытия RSA к задаче разложения на множители, система выдержала испытание практикой и является признанным стандартом de-facto в промышленной криптографии, а также официальным стандартом ряда международных организаций. С другой стороны, свободное распространение программного обеспечения, основанного на RSA, ограничено тем, что алгоритм RSA защищен в США рядом патентов. RSA можно применять как для шифрования/расшифровывания, так и для генерации/проверки электронно-цифровой подписи.

Генерация ключа. Первым этапом любого асимметричного алгоритма является создание пары ключей: открытого и закрытого и распространение открытого ключа "по всему миру". Для алгоритма RSA этап создания ключей состоит из следующих операций :

Выбираются два простых числа p и q. Вычисляется их произведение n=(p*q). Выбирается произвольное число e (e<n), такое, что

НОД(e,(p-1)(q-1))=1, то есть e должно быть взаимно простым с числом (p-1)(q-1).

Методом Евклида решается в целых числах уравнение e*d+(p-1)(q-1)*y=1. Здесь неизвестными являются переменные d и y – метод Евклида как раз и находит множество пар (d,y), каждая из которых является решением уравнения в целых числах. Два числа (e,n) – публикуются как открытый ключ.

Число d хранится в строжайшем секрете – это и есть закрытый ключ, который позволит читать все послания, зашифрованные с помощью пары чисел (e,n).

Шифрование/расшифровывание. Отправитель разбивает свое сообщение на блоки, равные k=[log2(n)] бит, где квадратные скобки обозначают взятие целой части от дробного числа.

Подобный блок, как Вы знаете, может быть интерпретирован как число из диапазона $(0;2^k-1)$. Для каждого такого числа (m_i) вычисляется выражение $c_i=((m_i)^e) \bmod n$. Блоки c_i и есть зашифрованное сообщение, и их можно спокойно передавать по открытому каналу, поскольку операция возведения в степень по модулю простого числа, является необратимой математической задачей. Обратная ей задача носит название "логарифмирование в конечном поле" и является на несколько порядков более сложной задачей. То есть даже если злоумышленник знает числа e и n, то по c_i прочесть исходные сообщения m_i он не может никак, кроме как полным перебором m_i.

А вот на приемной стороне процесс дешифрования все же возможен, и поможет нам в этом хранимое в секрете число d. Достаточно давно была доказана теорема Эйлера, частный случай которой утверждает, что если число n представимо в виде двух простых чисел p и q, то для любого x имеет место равенство $(x^{(p-1)(q-1)}) \bmod n=1$. Для дешифрования RSA-сообщений воспользуемся этой формулой.

Возведем обе ее части в степень (-y) : $(x^{(-y)(p-1)(q-1)}) \bmod n = 1^{(-y)} = 1$.

Теперь умножим обе ее части на x : $(x^{(-y)(p-1)(q-1)+1}) \bmod n = 1*x = x$.

А теперь вспомним как создавались открытый и закрытый ключи с помощью алгоритма Евклида подбиралось такое d, что e*d+(p-1)(q-1)*y=1, то есть e*d=(-y)(p-1)(q-1)+1. А следовательно в последнем выражении предыдущего абзаца можем заменить показатель степени на число (e*d).

Получаем $(x^{e*d})\text{mod } n = x$. То есть для того чтобы прочесть сообщение $c_i=((m_i)^e)\text{mod } n$ достаточно возвести его в степень d по модулю m:

$$((c_i)^d)\text{mod } n = ((m_i)^{e*d})\text{mod } n = m_i.$$

На самом деле операции возведения в степень больших чисел достаточно трудоемки для современных процессоров, даже если они производятся по оптимизированным по времени алгоритмам. Поэтому обычно весь текст сообщения кодируется обычным блочным шифром (намного более быстрым), но с использованием ключа сеанса, а вот сам ключ сеанса шифруется как раз асимметричным алгоритмом с помощью открытого ключа получателя и помещается в начало файла.

2.3. Система шифрования RSA

В системах с открытым ключом используются два ключа - открытый и закрытый, связанные определенным математическим образом друг с другом. Открытый ключ передаётся по открытому (то есть незащищённому, доступному для наблюдения) каналу и используется для шифрования сообщения и для проверки ЭЦП. Для расшифровки сообщения и для генерации ЭЦП используется секретный ключ [3].

Данная схема решает проблему симметричных схем, связанную с начальной передачей ключа другой стороне. Если в симметричных схемах злоумышленник перехватит ключ, то он сможет как «слушать», так и вносить правки в передаваемую информацию. В асимметричных системах другой стороне передается открытый ключ, который позволяет шифровать, но не расшифровывать информацию. Таким образом решается проблема симметричных систем, связанная с синхронизацией ключей [5].

Первыми исследователями, которые изобрели и раскрыли понятие шифрования с открытым кодом, были Уитфилд Диффи и Мартин Хеллман из Стэнфордского университета, и Ральф Меркле из Калифорнийского университета в Беркли. В 1976 году их работа «Новые направления в современной криптографии» открыла новую область в криптографии, теперь известную как криптография с открытым ключом.

Если необходимо наладить канал связи в обе стороны, то первые две операции необходимо проделать на обеих сторонах, таким образом, каждый будет знать свои закрытый, открытый ключи и открытый ключ собеседника. Закрытый ключ каждой стороны не передается по незащищенному каналу, тем самым оставаясь в секретности.

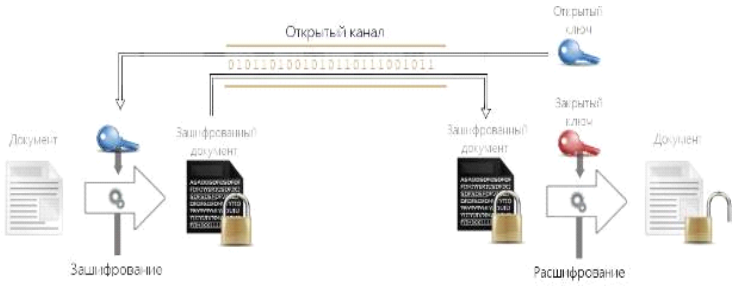


Рис.2.1. Ассиметричное шифрование

В асимметричных системах необходимо применять длинные ключи (512 битов и больше). Длинный ключ резко увеличивает время шифрования. Кроме того, генерация ключей весьма длительна. Зато распределять ключи можно по незащищенным каналам. В симметричных алгоритмах используют более короткие ключи, т.е. шифрование происходит быстрее. Но в таких системах сложно распределение ключей.

Поэтому при проектировании защищенной системы часто применяют и симметричные, и асимметричные алгоритмы. Так как система с открытыми ключами позволяет распределять ключи и в симметричных системах, можно объединить в системе передачи защищенной информации асимметричный и симметричный алгоритмы шифрования. С помощью первого рассылать ключи, вторым же - собственно шифровать передаваемую информацию [6, с. 53].

Обмен информацией можно осуществлять следующим образом:

– получатель вычисляет открытый и секретный ключи, секретный ключ хранит в тайне, открытый же делает доступным;

– отправитель, используя открытый ключ получателя, зашифровывает сеансовый ключ, который пересылается получателю по незащищенному каналу;

– получатель получает сеансовый ключ и расшифровывает его, используя свой секретный ключ;

– отправитель зашифровывает сообщение сеансовым ключом и пересылает получателю;

– получатель получает сообщение и расшифровывает его.

Пусть m и e натуральные числа. Функция f реализующая схему RSA, устроена следующим образом

$$f : x \to x^e \pmod m. \tag{2.1}$$

Для расшифровки сообщения $a = f(x)$ достаточно решить сравнение

$$x^e = a \pmod m. \tag{2.2}$$

При некоторых условиях на m и e это сравнение имеет единственное решение x.

Для того, чтобы описать эти условия и объяснить, как можно найти решение, нам потребуется одна теоретико-числовая функция, так называемая функция Эйлера. Эта функция натурального аргумента m обозначается $\varphi(m)$ и равняется количеству целых чисел на отрезке от 1 до m, взаимно простых с m. Так $\varphi(1) = 1$ и $\varphi(p^r) = p^{r-1}(p-1)$ для любого простого числа p и натурального r. Кроме того, $\varphi(ab) = \varphi(a)\varphi(b)$ для любых натуральных взаимно простых a и b. Эти свойства позволяют легко вычислить значение $\varphi(m)$, если известно разложение числа m на простые сомножители.

Если показатель степени e в сравнении (2.2) взаимно прост с $\varphi(m)$, то сравнение (2.2) имеет единственное решение. Для того, чтобы найти его, определим целое число d, удовлетворяющее условиям

$$de \equiv 1 \pmod{\varphi(m)}, \quad 1 \le d < \varphi(m). \tag{2.3}$$

Такое число существует, поскольку $(e, \varphi(m)) = 1$, и притом единственно. Здесь и далее символом (a, b) будет обозначаться наибольший общий делитель чисел a и b. Классическая теорема Эйлера, утверждает, что для каждого числа x, взаимно простого с m, выполняется сравнение $x^{\varphi(m)} \equiv 1 \pmod m$ и, следовательно.

$$a^d \equiv x^{de} \equiv x \pmod m. \tag{2.4}$$

Таким образом, в предположении $(a, m) = 1$, единственное решение сравнения (2.2) может быть найдено в виде

$$x \equiv a^d \pmod{m}. \qquad (2.5)$$

Если дополнительно предположить, что число m состоит из различных простых сомножителей, то сравнение (2.5) будет выполняться и без предположения $(a, m) = 1$. Действительно, обозначим $r = (a, m)$ и $s = m/r$. Тогда $\varphi(m)$ делится на $\varphi(s)$, а из (2.2) следует, что $(x, s) = 1$. Подобно (2.4), теперь легко находим $x \equiv a^d \pmod{s}$. А кроме того, имеем $x \equiv 0 \equiv a^d \pmod{r}$. Получившиеся сравнения в силу $(r, s) = 1$ дают нам (2.5).

Функция (2.1), принятая в системе RSA, может быть вычислена достаточно быстро. Обратная к $f(x)$ функция $f^{-1} : x \to x^d \pmod{m}$ вычисляется по тем же правилам, что и $f(x)$, лишь с заменой показателя степени e на d. Таким образом, для функции (2.1) будут выполнены указанные выше свойства 1) и 2).

Для вычисления функции (2.1) достаточно знать лишь числа e и m. Именно они составляют открытый ключ для шифрования. А вот для вычисления обратной функции требуется знать число d. оно и является «секретом», о котором речь идёт в пункте в). Казалось бы. ничего не стоит. зная число m. разложить его на простые сомножители, вычислить затем с помощью известных правил значение $\varphi(m)$ и, наконец, с помощью (2.3) определить нужное число d. Все шаги этого вычисления могут быть реализованы достаточно быстро, за исключением первого. Именно разложение числа m на простые множители и составляет наиболее трудоемкую часть вычислений. В теории чисел несмотря на многолетнюю её историю и на очень интенсивные поиски в течение последних 20 лет, эффективный алгоритм разложения натуральных чисел на множители так и не найден. Конечно, можно, перебирая все простые числа до $\sqrt{m}$, и. деля на них m, найти требуемое разложение. Но, учитывая, что количество простых в этом промежутке, асимптотически равно $2\sqrt{m} \cdot (\ln m)^{-1}$, находим, что при m, записываемом 100 десятичными цифрами, найдётся не менее $4 \cdot 10^{42}$ простых чисел, на которые придётся делить m при

разложении его на множители. Очень грубые прикидки показывают, что компьютеру, выполняющему миллион делений в секунду, для разложения числа $m > 10^{99}$ таким способом на простые сомножители потребуется не менее, чем 10^{35} лет. Известны и более эффективные способы разложения целых чисел на множители, чем простой перебор простых делителей, но и они работают очень медленно.

Авторы схемы RSA предложили выбирать число m в виде произведения двух простых множителей p и q, примерно одинаковых по величине. Так как

$$\varphi(m) = \varphi(pq) = (p-1)(q-1), \tag{2.6}$$

то единственное условие на выбор показателя степени e в отображении (1) есть

$$(e,\ p-1) = (e,\ q-1) = 1. \tag{2.7}$$

Итак, лицо, заинтересованное в организации шифрованной переписки с помощью схемы RSA, выбирает два достаточно больших простых числа p и q. Перемножая их, оно находит число $m = pq$. Затем выбирается число e, удовлетворяющее условиям (2.7), вычисляется с помощью (2.6) число $\varphi(m)$ и с помощью (2.3) - число d. Числа m и e публикуются, число d остается секретным. Теперь любой может отправлять зашифрованные с помощью (2.1) сообщения организатору этой системы, а организатор легко сможет расшифровывать их с помощью (2.5).

Для иллюстрации своего метода Ривест, Шамир и Адлеман зашифровали таким способом некоторую английскую фразу. Сначала она стандартным образом (a=01, b=02, ... z=26, пробел=00) была записана в виде целого числа x, а затем зашифрована с помощью отображения (2.1) при

m=1143816257578888676693257799761466120102182967212423625625618429357069352457338978305971235639587050589890751475992900268795435 41

и $e = 9007$. Эти два числа были опубликованы, причем дополнительно сообщалось, что $m = pq$. где p и q - простые числа, записываемые соответственно 64 и 65 десятичными знаками. Первому, кто расшифрует соответствующее сообщение

$$f(x) = 968696137546220614771409222543558829057599911245743$$
$$198746951209308162982251457083569314766228839896280133919$$
$$90551829945157815154$$

Числа p и q оказались равными

$$p = 34905295108476509491478496199038981334177646384933878843990820577,$$

$$q = 32769132993266709549961988190834461413177642967992942539798288533.$$

Входные данные для шифрования файлов – два случайных числа, а также файл, который предварительно должен быть открыт (или текст, введенный в предназначенном для этого текстовом поле). На основе введенных пользователем чисел определяются два ближайших к ним простых числа. Число n – результат их перемножения, – будет входить в пары, являющиеся открытым и закрытым ключами. При этом пара чисел e и n является открытым ключом, а d и n – секретным.

Процедура расшифровки ранее зашифрованных открытым ключом файлов включает указание в соответствующих текстовых полях чисел, входящих в пару, составляющую секретный ключ.

2.4. Разработка программного обеспечения алгоритма RSA при шифровании потоков данных

2.4.1. Описание состава программных средств

Криптографический алгоритм RSA был реализован с помощью интегрированного пакета Delphi 7.0. Выбор данного языка программирования обоснован тем, что он предоставляет такие возможности, как объектно-ориентированный подход к программированию, основанный на формах, интеграция с программированием для Windows и компонентная технология. Среда визуального программирования Delphi 7.0 позволяет, с помощью компонентного подхода к созданию приложений, быстро и качественно

"собрать" интерфейс программы и большую часть времени использовать именно на реализацию составленного алгоритма [20].

Компилятор в машинный код обеспечивает высокую производительность, необходимую для построения приложений. Этот компилятор в настоящее время является самым быстрым в мире. Он предлагает легкость разработки и быстрое время проверки готового программного блока, а также обеспечивает качество кода. Кроме того, Delphi обеспечивает быструю разработку без необходимости писать вставки на C или ручного написания кода (хотя это возможно).

2.4.2. Описание модулей программы

Программа включает в себя два программных модуля.

Первый модуль содержит главную кнопочную форму (рис.2.2).

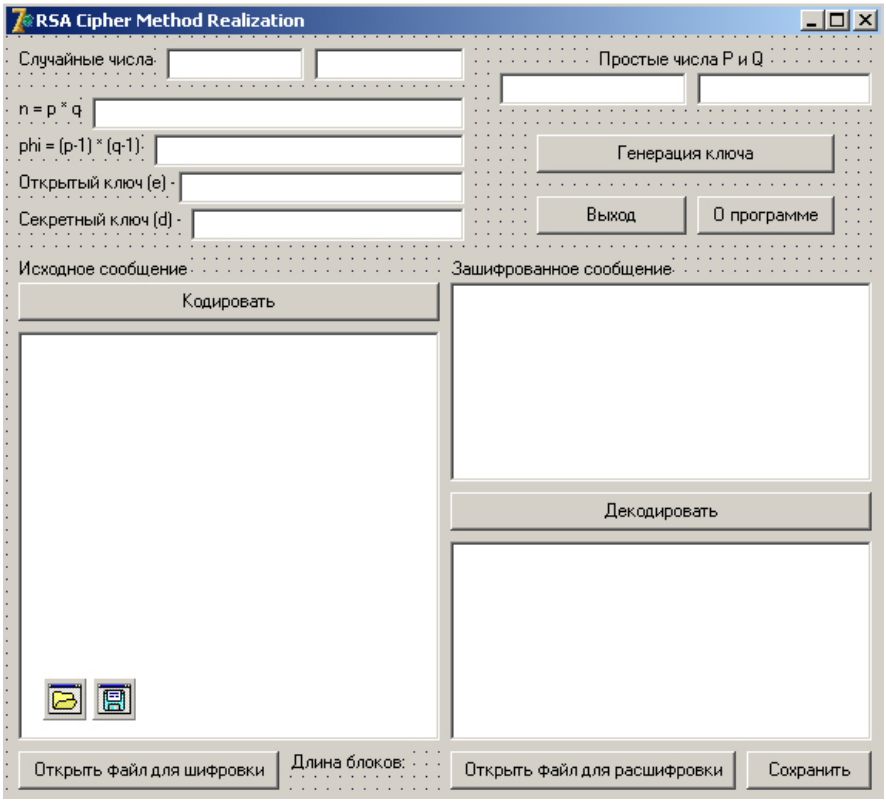


Рис.2.2. Главная кнопочная форма

Заголовок Caption главной формы изменен на «RSA Cipher Method Realization». Компонент bi_Maximize свойства BorderIcons установлен в значение False, а свойство BorderStyle изменено на bsSingle.

Форма Form1 содержит следующие компоненты: btn_About, btn_CryptFileOpen, btn_Decrypt, btn_DecryptFileOpen, btn_Encrypt, btn_Exit, btn_KeyGenerate, btn_SaveToFile, ed_n, ed_number1, ed_number2, ed_p, ed_phi, ed_PrivateKey, ed_PublicKey, ed_q, lbl_BlockLength, lbl_phi, lbl_PrivateKey, lbl_PublicKey, lbl_ScrambledMessage, lbl_SimpleNumbers, lbl_SourceMessage, mem_Cipher, mem_Decrypt, mem_Encrypt, OpenDialog1, SaveDialog1.

Первоначальные свойства компонентов изменены следующим образом:

btn_About:

Caption – «О программе»;

btn_CryptFileOpen:

Caption – «Открыть файл для шифровки»;

btn_Decrypt:

Caption – «Декодировать»;

btn_DecryptFileOpen:

Caption – «Открыть файл для расшифровки»;

btn_Encrypt:

Caption – «Кодировать»;

btn_Exit:

Caption – «Выход»;

btn_KeyGenerate:

Caption – «Генерация ключа»;

btn_SaveToFile:

Caption – «Сохранить»;

Enabled – False;

Свойство Enabled всех компонентов Edit, кроме ed_number1 и ed_number2, изменены на False, а свойство Text – на пустую строку. На этапе шифрования они служат лишь для вывода информации о ключах. Компоненты

ed_number1 и ed_number2 служат для ввода двух случайных чисел, на основе которых находятся два ближайших простых числа и генерируются открытый и секретный ключи.

Компоненты Label служат для пояснения назначения каждого из расположенных на форме компонентов. Их свойств Caption изменено следующим образом: lbl_BlockLength – «Длина блоков», lbl_phi – «phi = (p-1) * (q-1)», lbl_PrivateKey – «Секретный ключ (d)», lbl_PublicKey – «Открытый ключ (e)», lbl_ScrambledMessage – «Зашифрованное сообщение», lbl_SimpleNumbers – «Случайные числа», lbl_SourceMessage – «Исходное сообщение».

Свойство Enabled компонентов mem_Cipher и mem_Decrypt было изменено на False.

Свойство Filter компонентов OpenDialog1 и OpenDialog2 было изменено следующим образом: «Текстовые файлы (*.txt)|*.txt|Все файлы (*.*)|*.*», а свойство FilterIndex обоих компонентов было установлено в значение 1, что в соответствии со свойством Filter обозначает открытие текстовых файлов по умолчанию.

Второй модуль содержит информацию о программном продукте, созданном в ходе написания выпускной квалификационной работы.

Заголовок Caption формы второго модуля изменен на «О программе…», а свойство Visible установлено в значение False. Все компоненты списка свойства BorderIcons установлены в значение False.

На форме расположены компоненты btn_OK со свойством Caption «ОК» (служит для закрытия окна «О программе»), а также восемь компонентов Label, содержащих информацию о программном продукте. Их свойства Caption изменены соответствующим образом.

2.4.3. Состав проекта

Таблица 2.1

Состав проекта

Наименование	*Обозначение*	*Примечание*
RSA.exe	Исполнимый файл	-
RSA.dof	Файл параметров проекта	Содержит текущие установки проекта.
RSA.dpr	Файл проекта	Связывает все файлы, составляющие приложение
RSA.cfg	Файл конфигурации проекта	Содержит установки конфигурации проекта
RSA.res	Файл ресурсов	Содержит пиктограммы, графические изображения и другие ресурсы, используемые в проекте
Unit1.pas Unit2.pas	Файлы программных модулей для форм	Определяет функциональность формы
Unit1.dfm Unit2.dfm	Файл формы	Содержит список свойств всех компонентов, включенных в форму
Unit1.dcu Unt2.dcu	Объектный файл для Unit1.pas, Unit2.pas	Откомпилированная версия Unit.pas

2.4.4. Описание программы

Программная реализация криптографического алгоритма RSA разработана под операционную систему Windows 98/Me/2000/XP.

Программа написана на объектно-ориентированном языке программирования высокого уровня Delphi 7.0.

Программа представляет собой приложение, способное зашифровывать загружаемые пользователем файлы, генерируя открытый и секретный ключи на

основе введенных случайных чисел, а также расшифровывать ранее зашифрованные файлы.

Программа состоит из двух модулей. Модуль Unit1 содержит главную форму. На форме расположены основные компоненты, позволяющие производить все необходимые операции по шифрованию и расшифровке файлов.

Модуль содержит следующие процедуры. Кнопка btn_About запускает процедуру «procedure TForm1.btn_AboutClick (Sender: TObject);». Данная процедура делает видимой форму Form2, на которой расположена краткая информация о программе.

Кнопка btn_CryptFileOpen активирует процедуру «procedure TForm1.btn_CryptFileOpenClick (Sender: TObject);», которая запускает диалоговое окно открытия файла для шифрования. В случае открытия текстового файла его содержимое будет выведено в текстовом поле mem_Encrypt. Кроме открытия файла, данная процедура делает доступными кнопки кодирования (btn_Encrypt) и генерации ключей (btn_KeyGenerate), очищает содержимое полей mem_Cipher и mem_Decrypt, если в них уже находился какой-либо текст, и очищает поля для ввода чисел и вывода сгенерированных ключей, вызывая процедуру «procedure Clear (Sender: TObject)».

Кнопка btn_Decrypt вызывает процедуру «procedure TForm1.btn_DecryptClick (Sender: TObject)». Данная процедура запускает процесс расшифровки файла, который должен быть предварительно открыт. Расшифровка требует ввода пары чисел, составляющих секретный ключ. Для этого предназначены поля ввода ed_n и ed_PrivateKey, которые становятся доступными после открытия файла для расшифровки. Если не введено хотя бы одно число из этой пары, программа выдает сообщение об ошибке.

Кнопка btn_DecryptFileOpen запускает процедуру открытия файла для расшифровки «procedure TForm1.btn_DecryptFileOpenClick (Sender: TObject);». Данная процедура выводит диалоговое окно открытия файла. Кроме того, она

выполняет очистку текстовых полей mem_Encrypt и mem_Decrypt, делает недоступными кнопки кодирования (btn_Encrypt) и генерации ключей (btn_KeyGenerate), свойство кнопки дешифрования (btn_Decrypt) устанавливает в значение True и очищает поля для ввода чисел и ключей процедурой «procedure Clear (Sender: TObject);». Свойство PasswordChar компонентов ed_n и ed_PrivateKey, служащих для ввода чисел, составляющих секретный ключ, устанавливается в «#». Это делается затем, чтобы невозможно было подсмотреть секретный ключ во время его ввода пользователем. Кнопка btn_Encrypt активирует процедуру «procedure TForm1.btn_EncryptClick (Sender: TObject);». Эта процедура запускает процесс шифрования файла сгенерированными ключами. Генерация ключей производится на основе двух введенных пользователем простых чисел. Если введенные числа не позволяют рассчитать секретный ключ, выводится сообщение об ошибке.

Кнопка btn_Exit активирует процедуру «procedure TForm1.btn_ExitClick (Sender: TObject);», которая производит завершение работы приложения. Завершить работу программы можно также щелкнув по кнопке закрытия окна в заголовке главной формы.

Кнопка btn_KeyGenerate запускает процедуру «procedure TForm1.btn_KeyGenerateClick (Sender: TObject);», которая генерирует секретный и открытый ключи на основе введенных пользователем в поля ввода ed_number1 и ed_number2 случайных чисел. Если числа не введены, программа выводит сообщение об ошибке. Сгенерированные ключи отображаются в соответствующих компонентах Edit. Открытый ключ e отображается компонентом ed_PublicKey, закрытый – ed_PrivateKey. Число n из пары, составляющей ключ, находится в поле ed_n. Остальные поля предназначены для вывода дополнительной информации.

Кнопка btn_SaveToFile запускает процедуру «procedure TForm1.btn_SaveToFileClick (Sender: TObject);». Данная процедура открывает диалоговое окно сохранения файла. Кнопка btn_SaveToFile становится доступной только в том случае, если текстовое поле mem_Cipher содержит

какую-либо информацию, то есть если в этом поле содержится зашифрованная информация, которую можно сохранить в файл.

Помимо вышеперечисленных процедур, модуль Unit1 включает также следующие процедуры и функции:

«procedure Clear (Sender: TObject);» - производит очистку полей ввода чисел и ключей, а также дополнительной информации о ходе их генерации. Обнуляет ключи.

«function MemodN (m, e, n: int64): longword;» - реализует процесс кодирования числовых блоков по формуле $C(i) = M(i)^e \bmod n$ и раскодирование по формуле $M(i) = C(i)^d \bmod n$.

«function CodeAlgoritm (CodeText: String; se, sn: LongWord): string;» - разбивает сообщения на блоки, преобразует их в числовое представление и передает работу функции MemodN.

Модуль Unit2 содержит кнопку btn_OK, а также восемь компонентов Label, содержащих справочную информацию о приложении. Кнопка btn_OK устанавливает свойство Visible формы Form2 в значение False, тем самым закрывая окно «О программе». Для полноценного функционирования программы необходим персональный компьютер с центральным процессором класса Pentium 600 MMX, имеющий не менее 32 Мб оперативной памяти. При увеличении чисел, на основе которых генерируются ключи, может потребоваться процессор, обладающий большей производительностью, поскольку время генерации ключей обратно пропорционально производительности процессора. Мелким и средним предприятиям и частным лицам рекомендуется использовать ключи длиной от 56 до 96 бит, так как им не требуется обеспечение секретности на несколько лет вперед.

Для запуска программы необходимо вставить носитель информации в соответствующее устройство чтения (накопитель на гибких магнитных дисках), скопировать исполняемый файл «RSA.exe» в отдельную папку на жестком диске и произвести двойной щелчок левой кнопкой мыши по исполняемому файлу. После запуска приложения на экран выводится главное окно,

представленное на рис.2.3. Для генерации ключей необходимо ввести два случайных числа в поля, расположенные возле пояснительной надписи «Случайные числа», и нажать кнопку «Генерация ключа». В том случае, если хотя бы одно число не будет введено, программа выдаст сообщение об ошибке и ключи сгенерированы не будут.

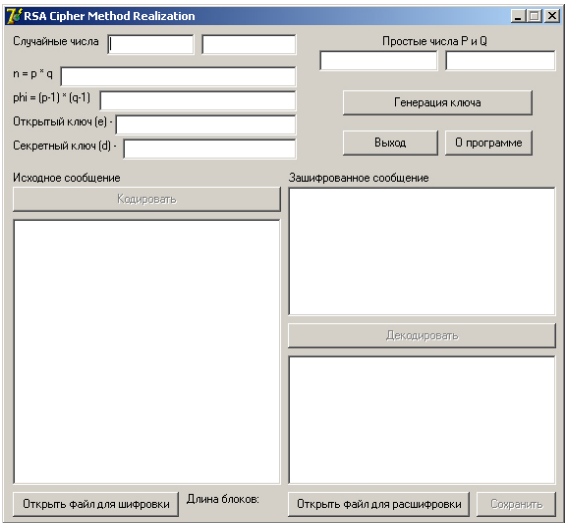


Рис.2.3. Главное окно приложения

После успешной генерации ключей они будут выведены в соответствующие поля. При этом пара чисел e и n будет являться открытым ключом, а пара d и n – секретным. Пользователь может распространить открытый ключ, разослав его другим пользователям, от которых он желает получать зашифрованные сообщения и файлы. Секретный ключ d необходимо сохранить в тайне – он будет использоваться исключительно для расшифровки.

Для того, чтобы зашифровать файл, необходимо нажать на кнопку «Открыть файл для шифровки» и в появившемся диалоговом окне «Открыть» (рис.2.4) выбрать необходимый файл. В случае необходимости шифрования текстовых сообщений можно воспользоваться текстовым полем «Исходное сообщение», набрав в нем нужный текст.

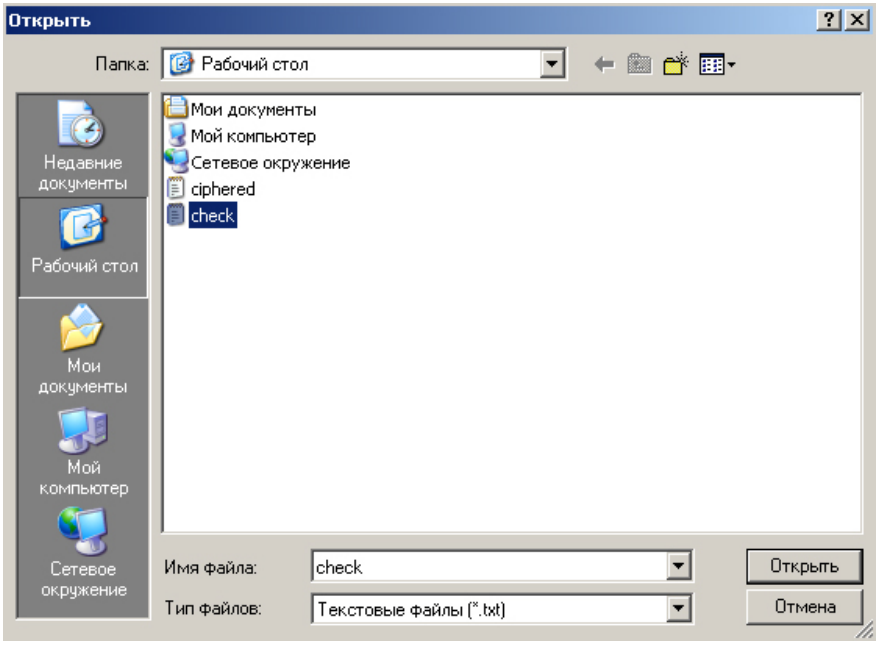


Рис.2.4. Диалоговое окно открытия файла

Для шифрования введенного текста или открытого файла требуется нажать кнопку «Кодировать». Результат работы алгоритма шифрования будет выведен в текстовом поле «Зашифрованное сообщение». Если процесс генерации ключей не был завершен успешно, то при попытке шифрования будет выведено сообщение об ошибке и процесс кодирования прервется.

После того, как информация будет зашифрована, станет доступной кнопка «Сохранить», которая позволяет сохранить зашифрованную информацию в отдельный файл. При нажатии этой кнопки открывается диалоговое окно «Сохранить как» (рис.2.5). От пользователя требуется ввести имя файла и путь, по которому он будет сохранен на носителе информации. В том случае, если такой файл уже существует, будет выдано соответствующее сообщение и предложение заменить уже существующий файл новым. Для расшифровки файла, который был ранее зашифрован, необходимо нажать на кнопку «Открыть файл для расшифровки» и в диалоговом окне открытия файла (рис.2.4) выбрать зашифрованный файл. Содержимое текстовых файлов в данном случае будет выведено в верхней части поля «Зашифрованное сообщение».

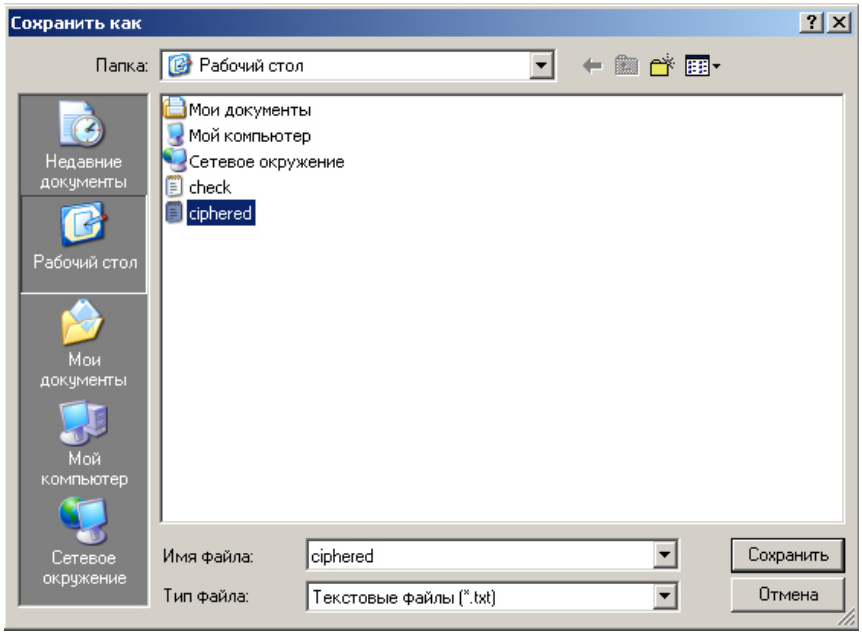


Рис.2.5. Диалоговое окно сохранения файла

Процедура расшифровки открытого ранее файла требует ввода числа n и секретного ключа d, которые должны были быть сохранены после генерации ключей. Каждый вводимый при этом символ будет отображаться на экране в виде символа «#», поскольку в этом случае сводится к минимуму угроза подглядывания секретного ключа посторонним лицом при вводе.

Если не было введено хотя бы одно число из пары, составляющей секретный ключ, программа выдаст сообщение об ошибке, и процесс расшифровки прервется.

При нажатии на кнопку «О программе» на экран будет выведено одноименное окно, на котором представлена краткая информация о программе.

Для завершения работы приложения необходимо либо нажать кнопку «Выход», либо щелкнуть мышью по кнопке закрытия главного окна в его заголовке.

Во время работы программы возможно появление следующих сообщений.

Если пользователем не введено хотя бы одно из случайных чисел, необходимых для генерации ключей, будет выведено сообщение об ошибке, изображенное на рис.2.6.

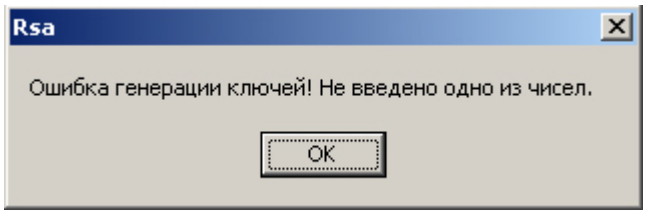


Рис.2.6. Сообщение об ошибке в процессе генерации ключей

Если пользователем введены числа, по которым нельзя рассчитать секретный ключ d, будет выведено сообщение, проиллюстрированное на рис.2.7.

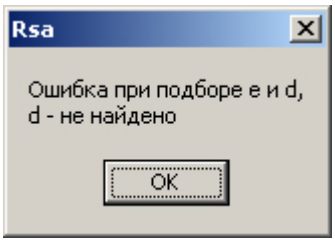


Рис.2.7. Сообщение об ошибке при расчете секретного ключа

Если пользователь не ввел хотя бы одно число из пары, составляющей закрытый ключ, при расшифровке файла, будет выведено сообщение об ошибке, представленное на рис.2.8.

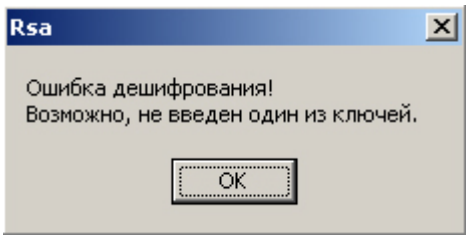


Рис.2.8. Сообщение об ошибке в процессе дешифрования

2.5. Практическая реализация алгоритма

Представленный выше алгоритм шифрования был реализован с помощью интегрированного пакета фирмы Borland Delphi 5.0. Выбор данного языка программирования обоснован тем что, он предоставляет такие возможности, как объектно-ориентированный подход к программированию,

основанный на формах, интеграция с программированием для Windows и компонентная технология. Среду визуального программирования Delphi 5 позволяет с помощью компонентного подхода к созданию приложений, быстро и качественно "собрать" интерфейс программы и большую часть времени использовать именно на реализацию составленного алгоритма.

Программа построена по технологии клиент/сервер, т.е. клиент передает по сети данные из стандартного потока ввода (с клавиатуры), предварительно зашифровав, сервер, получая поток данных, автоматически его расшифровывает.

Программный продукт состоит из двух приложений. Первое приложение представляет собой программу генерации ключей. Она выводит все простые числа заданного диапазона, из которых потом выбираются числа p и q. Там же находятся открытый и закрытый ключи, которые сохраняются на диске. Второе приложение, основная программа, производит соединение между двумя компьютерами и, отправляя сообщение, шифрует его. Это приложение клиент. Приложение сервер получает сообщение и расшифровывает его. Так же во второй программе содержится небольшая база данных абонентов, хранящая в себе имена абонентов, IP адреса и их открытые ключи.

В программном продукте были реализованы основные алгоритмы схемы RSA. Функция ModDegree производит вычисление $a = x^e \pmod m$. Функция Prost находит все простые числа заданного диапазона. Функция Evklid реализует алгоритм Евклида и находит закрытый ключ d. Функция HOD производит вычисление наибольшего общего делителя и находит открытый ключ e.

Результатом работы созданной программы являются зашифрованные и расшифрованные сообщения.

Для тестирования программы использовался пример приведенный в [11] $p = 19$, $q = 23$, $m = 437$ и $\varphi(m) = 396$. Также $e = 13$ и $d = 61$.

Использованный алгоритм RSA имеет ряд преимуществ:

1) алгоритм RSA является ассиметричным, т.е. он основывается на распространении открытых ключей в сети. Это позволяет нескольким пользователям обмениваться информацией, посылаемой по незащищенным каналам связи;

2) пользователь сам может менять как числа p, q, так и открытый и закрытый ключ по своему усмотрению, только потом он должен распространить открытый ключ в сети. Это позволяет добиваться пользователю нужной ему криптостойкости.

При всех этих преимуществах данный алгоритм имеет существенный недостаток – невысокая скорость работы. Алгоритм RSA работает более чем в тысячу раз медленнее симметричного алгоритма DES.

Из всего вышесказанного можно заключить, что данный алгоритм шифрования, хотя довольно медленный, но он ассиметричный и позволяет добиваться нужной криптостойкости, что делает его незаменимым при работе в незащищенных каналах связи.

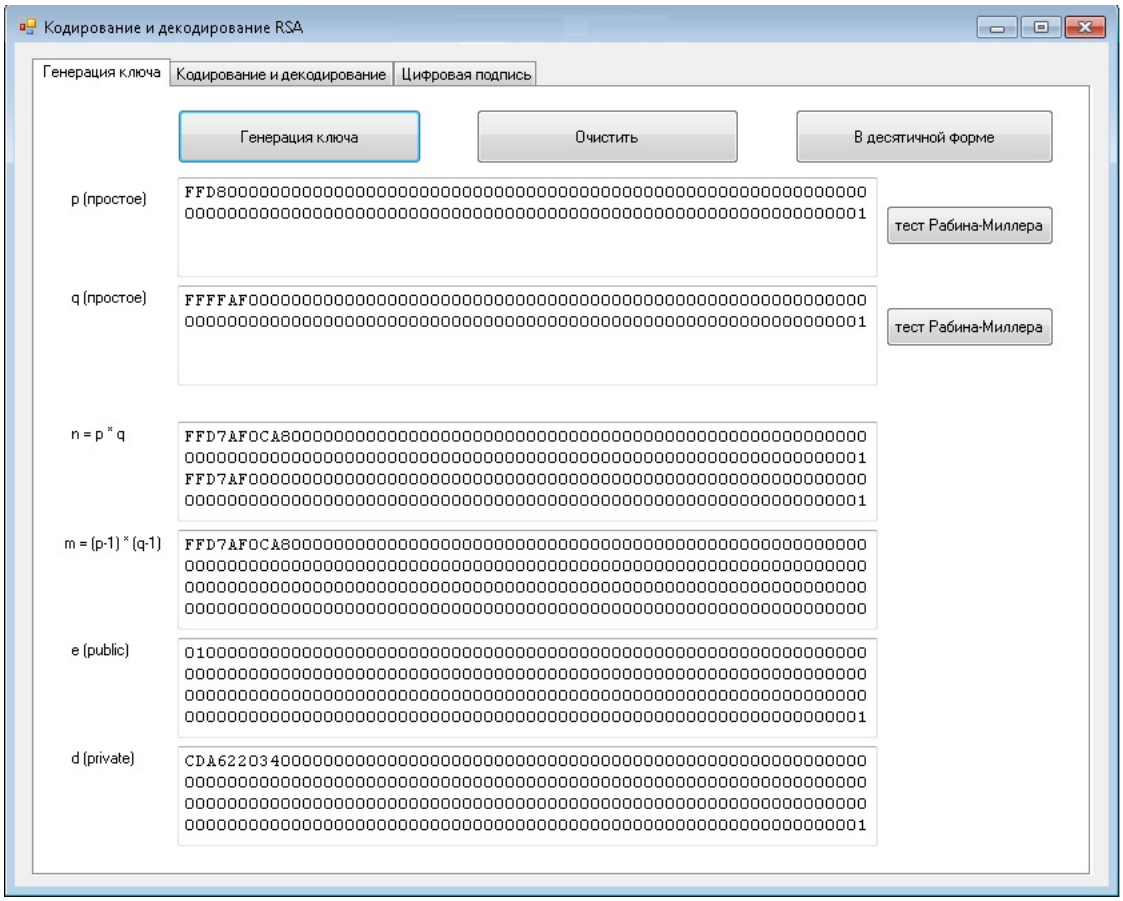
Исходя из проработанных данных, по построенному алгоритму и созданному программному продукту сделаны следующие выводы:

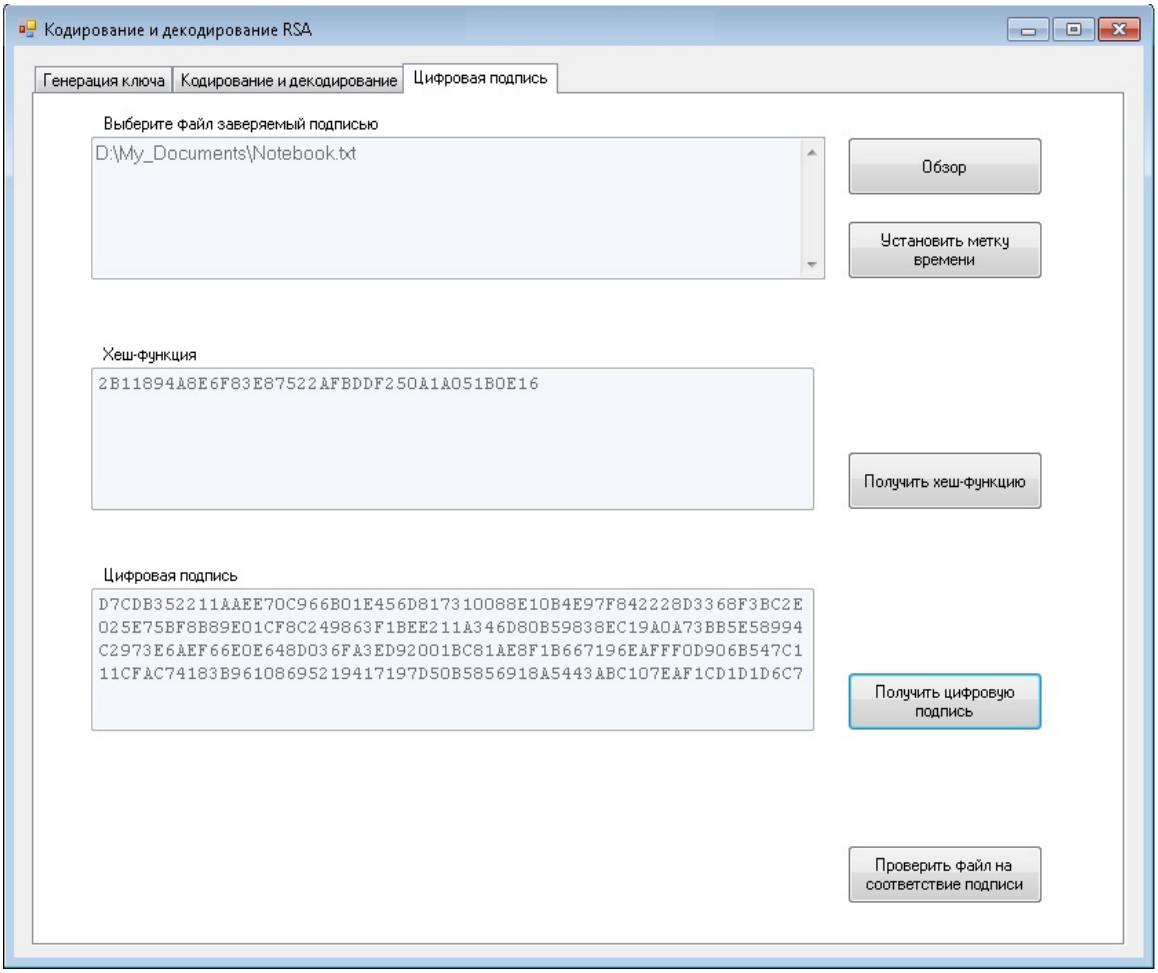
1) построенный алгоритм, а соответственно и созданный на его базе программный продукт, полностью реализует базовые механизмы схемы RSA и, таким образом удовлетворяют основным поставленным задачам;

2) данный программный продукт построен по технологии клиент/сервер и предназначен сохранять конфиденциальность передачи информации в сети.

Таким образом, по выводам о построенном алгоритме и созданном программном продукте можно заключить, что он подходит для решения проблем шифрования информации в системах управления, связанных с передачей данных по сети.

Получены следующие результаты:


Кодирование и декодирование RSA
Генерация ключа
Кодирование и декодирование
Цифровая подпись
Генерация ключа
Очистить
В десятичной форме
p (простое)
FFD8000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000001
тест Рабина-Миллера
q (простое)
FFFFAF0000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000001
тест Рабина-Миллера
n = p * q
FFD7AF0CA8000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000001
FFD7AF0000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000001
m = (p-1) * (q-1)
FFD7AF0CA8000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
e (public)
0100000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000001
d (private)
CDA6220340000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000001


Кодирование и декодирование RSA
Генерация ключа
Кодирование и декодирование
Цифровая подпись
Выберите файл заверяемый подписью
D:\My_Documents\Notebook.txt
Обзор
Установить метку времени
Хеш-функция
2B11894A8E6F83E87522AFBDDF250A1A051B0E16
Получить хеш-функцию
Цифровая подпись
D7CDB352211AAEE70C966B01E456D81731D088E1DB4E97F842228D3368F3BC2E
025E75BF8B89E01CF8C249863F1BEE211A346D80B59838EC19A0A73BB5E58994
C2973E6AEF66E0E648D036FA3ED92001BC81AE8F1B667196EAFFF0D906B547C1
11CFAC74183B96108695219417197D50B5856918A5443ABC107EAF1CD1D1D6C7
Получить цифровую подпись
Проверить файл на соответствие подписи

ГЛАВА III. БЕЗОПАСНОСТЬ ЖИЗНЕДЕЯТЕЛЬНОСТИ

Жизнедеятельность – это способ существования или повседневная деятельность человека. В процессе своей жизнедеятельности любой человек постоянно взаимодействует со средой обитания. Последняя – это окружающая человека среда в процессе его деятельности, обусловленная совокупностью физических, химических, биологических, психофизиологических и социально-экономических факторов, способных оказать прямое или косвенное, немедленное или отдаленное воздействие на деятельность человека, его здоровье и потомство. Основными средами обитания человека являются производственная среда, городская среда или среда населенных мест, бытовая или жилая среда и природная среда (ПС).

Оптимальное взаимодействие человека со средой обитания возможно, если будут обеспечены комфортность среды, минимизация негативных воздействий и устойчивое развитие системы “человек – среда обитания – машина – чрезвычайная ситуация”. Изучением элементов, составляющих эту систему, и явлений, происходящих в ней занимается безопасность жизнедеятельности (БЖД) – наука о комфортном и безопасном взаимодействии человека со средой обитания. Ее основная задача состоит в сохранении работоспособности и здоровья человека, выборе параметров состояния среды обитания и применении мер защиты от негативных факторов естественного и антропогенного происхождения. Основной целью изучения БЖД является приобретение теоретических знаний и практических навыков, необходимых для:

1) создания оптимального состояния среды обитания в зонах трудовой деятельности и отдыха человека;

2) идентификации (распознавание и количественная оценка) опасных и вредных факторов среды обитания естественного и антропогенного происхождения;

3) разработки и реализации мер защиты человека и среды обитания от негативных воздействий (опасностей);

4) проектирования и эксплуатации техники, технологических процессов и объектов народного хозяйства (ОНХ) в соответствии с требованиями по безопасности и экологичности;

5) обеспечения устойчивости функционирования ОНХ и ТС в штатных и чрезвычайных ситуациях;

6) прогнозирования развития и оценки последствий ЧС;

7) принятия решений по защите производственного персонала и населения от возможных последствий аварий, катастроф, стихийных бедствий и применения современных средств поражения, а также принятия мер по ликвидации их последствий.

Существуют три основных метода:

А – метод использующий пространственное и (или) временное разделение гомосферы (пространство, где находится человек в процессе рассматриваемой деятельности) и ноносферы (пространство, в котором постоянно существуют или периодически возникают опасности). Это достигается при механизации или автоматизации производственных процессов, дистанционном управлении оборудованием, использовании манипуляторов и роботов различных поколений.

Б – метод направленный на нормализацию ноносферы путём исключения опасностей и на приведение характеристик ноносферы в соответствии с характеристиками человека. Это совокупность мероприятий, защищающих человека от шума, вибраций, газа, пыли, опасности травмирования и т. д. С помощью СКЗ.

В – метод направленный на адаптацию человека к соответствующей среде и повышение его защищённости (с помощью СИЗ). Он реализуется путём профотбора, обучения, инструктирования, психологического воздействия и т. д. В нашем примере применяются методы Б и В, а также частично – А (дистанционное управление, освещение).

Средства коллективной защиты работающих от воздействия механических воздействий:

- оградительные устройства (кожухи, двери, щиты, планки и др.)
- предохранительные устройства
- тормозные устройства
- устройства автоматического контроля и сигнализации
- устройства дистанционного управления

Средства индивидуальной защиты от механических факторов - рабочая одежда, очки, рукавицы, каска.

Защита от шума: СИЗ - ушные пробки, наушники, шлемы

СКЗ - звукоизоляция, звукопоглащение, звукоглушение

Защита от вибрации.

СКЗ - виброгашении, виброизоляции

Защита от запыленности и загазованности

СКЗ - вентиляция и кондиционирование

СИЗ - респираторы , маски и противогаз

Обеспечение электробезопасности в ЭУ и на рабочем месте

Конструкцией ЭУ и ЭО все электротехнические изделия по способу защиты человека от поражения электротоком подразделены на 5 классов защиты :

0 ; 0I ; I ; II ; III

Технические способы и средства защиты от случайного прикосновения к токоведущим частям применяются :

- защитные оболочки
- защитные ограждения
- безопасное расположение токоведущих частей
- изоляция токоведущих частей и рабочих мест
- малое напряжение (не более 42 В)
- защитное отключение
- предупредительная сигнализация

- блокировка и знаки безопасности
- механическое запирание приводов включения ЭУ и ЭО

От прикосновения к металическим не токоведущим частям ЭУ и ЭО , которое может оказаться под напряжением в результате повреждения электроизоляции:

- зануление
- защитное заземление
- выравнивание потенциала
- защитное отключение
- изоляция токоведущих частей
- электрическое разделение сети
- малое напряжение
- контроль изоляции
- применение С.И.З.

Технические способы и средства защиты человека от электромагнитного поля:

- уменьшение напряженности плотности потока энергии ЭМП
- экранирование рабочих мест
- удаление рабочих мест от источника ЭМП
- рациональное размещение в цехе оборудования ЭМП
- установление рациональных режимов работы оборудования и обслуживающего персонала
- применение предупредительной сигнализации
- применение С.И.З.

Технические способы и средства защиты зданий и сооружений от разрядов и воздействий атмосферного электричества:

- молниеотводы ЗУ определенных конструкций, к которым присоединяются оборудование и металлические конструкции для ограничения

перенапряжений на них; от электромагнитной индукции и запаса высокого потенциала

- перемычки в местах сближения металлических коммуникаций.

К работе в ЭУ допускаются лица не моложе 18 лет, прошедшие инструктаж, обучение и стажировку безопасным методам труда, проверку знаний, правил ТБ , ТЭ, ПБ , а также должностных инструкций и инструкций по охране труда - в соответствии с занимаемой должностью и присвоением соответствующей группы по электробезопасности и прошедших медосмотр.

Для безопасного проведения работ должны выполняться следующие организационные мероприятия:

- назначение лиц, ответственных за безопасное проведение работ
- выдача наряда или распоряжения
- выдача разрешения на подготовку рабочего места и на допуск
- подготовка рабочего места и допуск
- надзор при выполнении работ
- перевод бригады на другое рабочее место
- оформление перерывов в работе и ее окончания

Для подготовки рабочего места при работе , требующей снятия напряжения, должны быть выполнены в указанном порядке следующие технические мероприятия:

- проведены необходимые отключения и приняты меры, препятствующие ошибочному или самопроизвольному включению коммутационной аппаратуры
- вывешены запрещающие плакаты на приводах ручного и на ключах дистанционного управления коммутационной аппаратурой
- проверено отсутствие напряжения на токоведущих частях ,которые должны быть заземлены для защиты людей от поражения электротоком
- установлено заземление (включены заземляющие ножи, установлены переносные заземления)

- ограждены при необходимости рабочие места или оставшиеся под напряжением токоведущие части и вывешены на ограждениях соответствующие плакаты. В зависимости от местных условий токоведущие части ограждаются до или после их заземления

Через глаза человек получает около 90% всей информации. Качество ее поступления во многом зависит от освещения. При неудовлетворительном освещении человек напрягает зрительный аппарат, что ведет к утомлению зрения и организма в целом. Одновременно человек теряет ориентацию среди оборудования, что повышает опасность его травмирования.

По функциональному назначению освещение подразделяется на:

- рабочее освещение (естественный и искусственный свет);
- аварийное освещение (искусственный свет);
- эвакуационное освещение (искусственный свет);
- дежурное освещение (искусственный свет).

В зависимости от источника света освещение может быть:

- естественным (создается солнечным диском диффузионным светом небосвода);
- искусственным (создается электролампами);
- совмещенным (естественное + искусственное).

Искусственное освещение

Искусственное освещение применяется в темное время суток и в помещениях, где нет естественного освещения. По конструктивному исполнению оно подразделяется на:

- общее (равномерное или локализованное);
- комбинированное (общее + местное).

Одно местное освещение в производственных помещениях не допускается. Источниками искусственного света являются лампы накаливания (ЛН) и газоразрядные лампы (ГРЛ).

Выбор искусственных источников света производят по приложению 6 СНиП 2-4-79 в зависимости от характера зрительной работы и

цветоразличению. При этом в помещениях без или с недостаточным естественным освещением (КЕО < 0.1…0.3%) применяют ультрафиолетовые лампы для компенсации солнечной недостаточности. ЛН и ГРЛ с пускорегулирующим аппаратом заключаются в специальную арматуру, предохраняющую глаза от действий ярких частей лампы, обеспечивающую требуемое распределение светового потока и предохраняющую лампу от перегревания, осевшей пыли и влаги, механических повреждений. Такая арматура с источником света составляет светильник. Уровень освещенности нормируется СНиП 2-4-79 раздельно для различных помещений, мест работы вне зданий, и наружного освещения городов, поселков и пунктов. Для производственных помещений при этом устанавливается рабочая (ГРЛ) минимальная освещенность (E_{min}) в зависимости от точности зрительной работы и системы освещения. Для искусственного освещения также предусмотрено восемь разрядов зрительной работы, но первые пять разрядов разделены на четыре подразряда (а, б, в, г) в зависимости от соотношений “контраст объекта различения с фоном – характеристика фона”. При использовании ЛН рабочую освещенность по СНиП 2-4-79 следует снижать по шкале освещенности на 1 или 2 ступени в зависимости от системы освещенности и разряда зрительных работ. Она не должна превышать 300 лк.

Аварийное освещение. Аварийное освещение необходимо для продолжения работы временном погасании рабочего освещения в помещениях, когда отсутствие искусственного освещения может вызвать тяжелые последствия для людей, технологических процессов, оборудования и предприятия в целом. При аварийном освещении освещенность (ЕА) должна быть не ниже 5% от рабочей общей освещенности, но не менее 2 лк внутри здания (но и не более 30 лк) и не менее 1 лк для площадок предприятия. При освещенности в здании более 30 лк (ГРЛ) и более 10 лк (ЛН) требуется обязательное обоснование аварийного освещения.

Рассчитать методом светового потока потребное количество светильников с ЛН и ГЛ для общего освещения производственного помещения

по исходным данным, выбрать экономически целесообразную осветительную установку и расположить светильники на плане помещения. При этом высота светильника от потолка h_c – 0,4 м; высота рабочей поверхности от пола h_p – 0,8м; коэффициент отражения света от потолка $\rho_п$ = 50%, от стен $\rho_{ст}$ = 30%, и от рабочей поверхности $\rho_р$ = 10%.

Исходные данные:

Помещение – экспериментальная лаборатория

Разряд и подразряд зрительных работ – V б.

Минимальная освещенность - E_{min}=150 лк

Размеры помещения – 24 х 12 х 4,2 м

1. Определяем высоту подвеса светильника над рабочей поверхностью, [м] $h = H - h_P - h_C = 4{,}2 - 0{,}8 - 0{,}4 = 3м$, где Н - высота помещения

2. Вычислим освещаемую площадь помещения, [м^2]

$S = A \cdot B = 24 \cdot 12 = 288$, где А и В длина и ширина помещения

3. Для расчета освещения методом светового потока вычисляем индекс помещения

$$i = \frac{S}{h \cdot (A + B)} = \frac{288}{3 \cdot (24 + 12)} = 0{,}83$$

4. С учетом коэффициентов отражения $\rho_п$, $\rho_с$, $\rho_р$ и определенного выше условного номера группы светильника, находим коэффициент светового потока η (%). для ЛН – η = 60, для ГЛ – η = 73

5. По таблице 4.4 (ЛН) и 4.25 (ДРЛ) находим световой поток заданной лампы, $Ф_л$, [лм]

Для ЛН Б-100 $Ф_л$=1540 лм, а для ГЛ ДРЛ125 $Ф_л$=6000 лм .

Определяем потребное количество светильников, [шт]

Для ЛН: $$N_C = \frac{100 \cdot E_{\min} \cdot S \cdot K_з \cdot Z}{n_i \cdot Ф_л \cdot \eta \cdot K_\gamma} = \frac{100 \cdot 100 \cdot 288 \cdot 1.3 \cdot 2}{1 \cdot 1540 \cdot 60 \cdot 0.9} \approx 90;$$

Для ГЛ: $$N_C = \frac{100 \cdot E_{\min} \cdot S \cdot K_з \cdot Z}{n_i \cdot Ф_л \cdot \eta \cdot K_\gamma} = \frac{100 \cdot 150 \cdot 288 \cdot 1.5 \cdot 1.5}{1 \cdot 6000 \cdot 73 \cdot 1} \approx 21;$$

где $К_{\gamma}$=0.8…0.9 - коэффициент затенения для помещений с фиксированным положением работающего, $К_з$ – коэффициент запаса устанавливаемый по таблице 3 СНиП II-4-79 ; Z – коэффициент неравномерности освещения (по СниП II-4-79 для зрительных работ IV-VII разрядов $Z_{ЛН}$ = 2, $Z_{ЛЛ}$ = 1.5); n_i – число ламп в светильнике (1);

6. Определяем экономическую эффективность проектируемых осветительных установок с ЛН и ГЛ. Для этого определяем суммарные затраты $С_{\Sigma}$ (капитальные + основные эксплуатационные затраты), руб., на эти установки по формуле: $С_{\Sigma}$ =Су*$Р_{\Sigma}$ + Ск*$Р_{\Sigma}$*Т*Кисп, где Су – стоимость установки 1 кВт осветительного оборудования, руб.; $Р_{\Sigma}$ - расчетная суммарная мощность осветительной установки, кВт, равная произведению величин Nc и заданной мощности соответственно для ЛН и ГЛ, деленное на 1000; Ск стоимость 1 кВт*ч, руб.; Т – время работы установки в течение года (365*24=8760), ч; Кисп– среднее значение использования осветительной установки в течение года (принимают равным 0,6).

В упрощенном современном виде формула принимает вид:

Для ЛН $С_{\Sigma ЛН} = 255Р_{\Sigma} \cdot Ки$;

Для ГЛ $С_{\Sigma ГЛ} = 405Р_{\Sigma} . Ки$,

где Ки коэффициент индексации принимаем равным 10 с учетом деноминации (по данным СМИ в 1996 г. он был равен 10000).

$Р_{\Sigma ЛН}$= 90 · 100/1000=9 (кВт),

$Р_{\Sigma ГЛ}$= 21 · 125/1000=2,63 (кВт),

$С_{\Sigma ЛН}$ = 255 · 9 · 10= 22950 (руб.),

$С_{\Sigma ГЛ}$ = 405 · 2,63 · 10=10631,25 (руб.)

$C_{\sum ГЛ} < C_{\sum ЛН,}$ следовательно экономически более эффективна осветительная установка с ГЛ, поэтому ее и принимаем за основу.

На третьем этапе разрабатываем рациональную схему равномерного размещения светильников в производственном помещении.

По таблице 9.5 найдем тип кривой силы света (КСС) светильника.

КСС –Д-2

Найдем значение λ в зависимости от КСС по таблице 1.1 $\lambda = 1$

Расстояние, м, между светильниками и рядами этих светильников определим по формуле:

$$L = \lambda \cdot h = 3 \cdot 1{,}2 = 3 \text{ м}$$

Оптимальное расстояние lк, м, от крайнего ряда светильников или от крайнего светильника до стен устанавливается 0,5 L:

Рабочие места у стен отсутствуют (в помещении размером 24×12 м^2 рабочие места можно разместить и не устанавливая их у стен), рассчитаем минимальное расстояние рядов от стен:

$$lк = 0{,}5 \cdot 3 = 1{,}5 \text{ м.}$$

Определим количество суммарную длину ряда светильников, [м]:

$$l_{\Sigma} = N_c \cdot l_c = 21 \cdot 3 = 63 \text{м},$$

Т. к. суммарная длинна ряда светильников выше чем длинна помещения (24 м), то светильники необходимо располагать в несколько рядов. Определим количество рядов светильников:

$$n_h = \frac{l_{\Sigma}}{A} = \frac{63}{21} = 3$$

Получаем 3 ряда с учетом отступа от стен. Определим кол-во светильников в ряду:

$$n_c = \frac{21}{3} = 7$$

Определим фактическую освещенность, лк, по формуле:

$$E_\phi = \frac{N \cdot n_i \cdot Фл \cdot \eta \cdot K\gamma}{100 \cdot S \cdot Кз \cdot Z} = \frac{21 \cdot 1 \cdot 6000 \cdot 73 \cdot 1}{100 \cdot 288 \cdot 1.5 \cdot 1.5} = 153.2$$

Еф > Emin, 153,2 > 150

Определим потребное количество светильников, шт., для аварийного освещения по формуле (*Emin = 0,5 лк*.).

$$N_C = \frac{100 \cdot E_{\min} \cdot S \cdot K_з \cdot Z}{n_i \cdot Ф_л \cdot \eta \cdot K_\gamma} = \frac{100 \cdot 0.5 \cdot 288 \cdot 1.5 \cdot 1.5}{1 \cdot 1540 \cdot 73 \cdot 1} \approx 1 \text{ светильник.}$$

Полученное количество совпадает с расчетным, следовательно система освещения не требует коррекции.

Для увеличения экономической эффективности освещения можно снизить кол-во используемых ламп, применяя светодиодные.

ЗАКЛЮЧЕНИЕ

Одним из основных средств защиты информации в системах управления являются криптографические средства. Они имеют своей задачей защиту информации при передаче по линиям связи, хранении на носителях, а так же препятствуют вводу ложной информации.

Изучены системы шифрования, проанализированы программы шифрования, выбраны наиболее подходящие средства реализации, проведены тестирование разработанного варианта программы шифрования, закреплены практические навыки программирования на языке Delphi.

Практическая реализация криптографических средств защиты может быть программной, т.е. шифрование реализуется специальной программой, и технической, с помощью специальных технических средств, реализующих алгоритм шифрования.

Разработан программный модуль, реализующий криптографический алгоритм RSA, и разработана программу, демонстрирующая работу данного алгоритма. Программа представляет собой приложение, способное зашифровывать загружаемые пользователем файлы для передачи их по каналам связи или последующего хранения на носителях информации. Открытый и секретный ключи генерируются на основе введенных пользователем случайных чисел. Программа позволяет также расшифровывать ранее закодированные файлы при помощи секретного ключа.

Данный программный модуль может применяться любыми технические и технологическими объектами, сохранение конфиденциальности информации которых имеет важное стратегическое значение. Кроме того, его можно использовать в составе других программных комплексов, к примеру для шифрования паролей в целях разграничения доступа к ресурсам.

ЛИТЕРАТУРА

1. Каримов И. А. Ўзбекистон мустақиликка эришиш остонасида. –Тошкент, 2013 й.
2. Каримов И. А. Она юртимиз бахту иқболи ва буюк келажаги йўлида хизмат қилиш-энг олий саодатдир. –Тошкент, 2015 й.
3. Романец Ю.В., Тимофеев П.А., Шаньгин В.Ф. Защита информации в компьютерных системах и сетях - М.: Радио и связь, 2012.
4. Белкин П.Ю., Михальский О.О., Першаков А.С. Программно-аппаратные средства обеспечения информационной безопасности. Защита программ и данных: Учеб. пос. для вузов, М.: Радио и связь, 2009.
5. Анин Б.Ю. Защита компьютерной информации. - СПб.: БХВ - Санкт-Петербург, 2014.
6. Зегжда Д.П.,Ивашко А.М. Основы безопасности информационных систем - М.: «Горячая линия – Телеком», 2013.
7. Анин Б.Ю. Защита компьютерной информации. - СПб.: БХВ - Санкт-Петербург, 2015.
8. Молдовян А.А., Молдовян Н.А., Советов Б.Я. Криптография. Спб.: «Лань», 2012.
9. Романец Ю.В., Тимофеев П.А., Шаньгин В.Ф. Защита информации в компьютерных системах и сетях / Под ред. В.Ф. Шаньгина. – 2-е изд., перераб. и доп. – М.: Радио и связь, 2011
10. Алферов А.П., Зубов А.Ю., Кузьмин А.С. Черемушкин А.В. Основы криптографии: Учебное пособие - М.: Гелиос, 2014
11. Баричев С.Г., Гончаров В.В., Серов Р.Е. Основы современной криптографии. М.: «Горячая линия – Телеком», 2015
12. Б. Шнайер, Прикладная криптография - М.: Издательство ТРИУМФ, 2002.
13. Архангельский А. Я. Delphi7. Справочное пособие, – М: ЗАО «Издательство БИНОМ», 2003

14. Партыка Т.Л., Попов И.И.Информационная безопасность. Учебное пособие для студентов учреждений среднего профессионального образования.– М.: ФОРУМ: ИНФРА-М, 2004.
15. Корт С.С. Теоретические основы защиты информации: Учебное пособие. – М.: Гелиос АРВ, 2004
16. Смирнова Г.Н. Проектирование экономических информационных систем: Учебник/ Г.Н.Смирнова, А.А.Сорокин, Ю.Ф.Тельнов; Под ред. Ю.Ф.Тельнова – М.: Финансы и статистика, 2014.
17. Фаронов В.В. Delphi – программирование на языке высокого уровня:Учебник для вузов – СПб.: Питер, 2005
18. http://kiev-security.org.ua
19. http://nsa.by.ru/docs/security/
20. http://www.citforum.ru/internet/securities/
21. www.ziyonet.uz
22. www.google.uz

ПРИЛОЖЕНИЕ

Листинг программы

```
unit unit1;

interface

uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, ExtCtrls, ComCtrls;
type
TForm1 = class(TForm)
ed_number1: TEdit;
btn_KeyGenerate: TButton;
ed_number2: TEdit;
ed_p: TEdit;
ed_q: TEdit;
lbl_Numbers: TLabel;
lbl_SimpleNumbers: TLabel;
ed_n: TEdit;
lbl_n: TLabel;
ed_PublicKey: TEdit;
ed_PrivateKey: TEdit;
lbl_PublicKey: TLabel;
ed_phi: TEdit;
btn_Encrypt: TButton;
btn_Decrypt: TButton;
mem_Encrypt: TMemo;
mem_Cipher: TMemo;
lbl_SourceMessage: TLabel;
lbl_ScrambledMessage: TLabel;
lbl_BlockLength: TLabel;
mem_Decrypt: TMemo;
OpenDialog1: TOpenDialog;
btn_CryptFileOpen: TButton;
btn_SaveToFile: TButton;
btn_DecryptFileOpen: TButton;
btn_Exit: TButton;
lbl_PrivateKey: TLabel;
lbl_phi: TLabel;
SaveDialog1: TSaveDialog;
btn_About: TButton;
procedure btn_KeyGenerateClick(Sender: TObject);
procedure btn_EncryptClick(Sender: TObject);
```

```
procedure btn_DecryptClick(Sender: TObject);
procedure btn_CryptFileOpenClick(Sender: TObject);
procedure btn_SaveToFileClick(Sender: TObject);
procedure btn_DecryptFileOpenClick(Sender: TObject);
procedure btn_ExitClick(Sender: TObject);
procedure mem_EncryptChange(Sender: TObject);
procedure btn_AboutClick(Sender: TObject);

private
{ Private declarations }
public
{ Public declarations }
end;

var
Form1: TForm1;
n,X: longWord;
d,e: LongWord;

implementation

{$R *.dfm}

procedure Clear (Sender: TObject);
begin
Form1.ed_n.Text := '';
Form1.ed_phi.Text := '';
Form1.ed_PublicKey.Text := '';
Form1.ed_PrivateKey.Text := '';
Form1.ed_p.Text := '';
Form1.ed_q.Text := '';
n := 0; X := 0; d := 0; e := 0;
end;

{***}

function MEmodN(m,e,n: int64): longword;
var i: longword;
x: int64;
begin
X := M mod N;
for i := 2 to e do
begin
X := X*M - ((X*M) div N)*N;
end;
```

```
Result := X;
end;

{***}

function CodeAlgoritm(CodeText: String; se, sn: LongWord): string;
var Code: string;
i, SimbCount, Size, CodeWord: LongWord;
begin
SimbCount := Length(CodeText);
CodeWord := 0;
Size := 1;
for i := 1 to SimbCount do
begin
if ord(CodeText[i])<>0 then CodeWord := CodeWord + (ord(CodeText[i])-
1)*Size
else CodeWord := CodeWord;
Size:= Size*254;
end;

CodeWord := MEmodN(CodeWord,se,sn);

Code:= '';
Size:= 254;
for i:= 1 to SimbCount do
begin
Code:= Code+chr((CodeWord mod Size)+1);
CodeWord:= CodeWord div Size;
end;
Result:= Code;
end;

{***}

procedure TForm1.btn_KeyGenerateClick(Sender: TObject);
var nearp: word;
nearq: word;

p,q: word;
i,j, temp: longword;

Flag: boolean;
begin
e := 0; d := 0; x := 0; n := 0; p := 0; q := 0;
if (ed_number1.Text = '') or (ed_Number2.Text = '') then
```

```
begin
ShowMessage ('Ошибка генерации ключей!' + #13 + ‘Не введено одно из чисел.’);
exit;
end;
Form1.Enabled:= False;

nearp:= StrToInt(ed_number1.Text);

for i := nearp downto 1 do
begin
Flag := False;
for j := 2 to (round(sqrt(i))+1) do
begin
if ((i mod j)=0) and (i<>j) then
begin
Flag := True;
Break;
end;
end;

if not Flag then
begin
P := i;
Break;
end;
end;

nearq := StrToInt(ed_number2.Text);

for i := nearq downto 1 do
begin
Flag := False;
for j := 2 to (round(sqrt(i))+1) do
begin
if ((i mod j)=0) and (i<>j) then
begin
Flag := True;
Break;
end;
end;

if not Flag then
begin
Q := i;
```

```
Break;
end;
end;

ed_p.Text := IntToStr(p);
ed_q.Text := IntToStr(q);

n := p*q;

x := (p-1)*(q-1);

ed_n.Text := IntToStr(n);
ed_phi.Text := IntToStr(x);
Application.ProcessMessages;

For i := Round(X/4) to X-1 do
begin
Flag := False;
For j := i downto 2 do
begin
if ((i mod j)=0) and ((X mod j)=0) then
begin
Flag := True;
Break;
end;
end;

if not Flag then
begin
E := i;
Break;
end;
end;
ed_PublicKey.Text := IntToStr(E);
Application.ProcessMessages;

For i := 2 to X do
begin
if (E*I mod X = 1) and (E<>I) then
begin
D := I;
Break;
end;
end;
```

```
ed_PrivateKey.Text := IntToStr(D);
Form1.Enabled := True;

if d = 0 then
begin
ShowMessage('Ошибка при подборе e и d,'+#13+ 'd – не найдено');
N := 1;
Exit;
end;
Temp := X-6 ;
Temp := MEmodN(Temp,e,N);
Temp := MEmodN(Temp,d,N);
if Temp <> X-6 then
begin
ShowMessage('Ошибка при подборе e и d,'+#13+ 'вероятно переполнение разрядной сетки');
N := 1;
end;
end;

{***}

procedure TForm1.btn_EncryptClick(Sender: TObject);
var CodeText: string;
SimbCount,Size:LongWord;
begin
if N < 256 then
begin
ShowMessage ('Ne kodiruem! Error!');
Exit;
end;

SimbCount := 0;
Size := 255;

while Size < N do
begin
Size := Size*255;
Inc(SimbCount);
end;

CodeText := mem_Encrypt.Text;

while CodeText[length(CodeText)] in [#13,#10] do Delete(CodeText, length(CodeText),1);
```

```
while (length(CodeText) mod SimbCount)<>0 do
 CodeText := CodeText + ' ';

mem_Cipher.Text:= '';

while length(CodeText)>0 do
begin
mem_Cipher.Text:=
mem_Cipher.Text+CodeAlgoritm( Copy(Codetext,1,SimbCount)+#0 , e, n);
Delete(CodeText, 1, SimbCount);
end;
lbl_BlockLength.Caption := 'Длина блоков: '+IntToStr(SimbCount+1);
btn_SaveToFile.Enabled := true;
btn_Decrypt.Enabled := true;
end;

{***}

procedure TForm1.btn_DecryptClick(Sender: TObject);
var CodeText: string;
SimbCount,Size:LongWord;
Begin
if (ed_n.Text='') or (ed_PrivateKey.Text = '') then
begin
ShowMessage ('Ошибка дешифрования!' + #13 + 'Возможно, не введен один из ключей.');
Exit;
end;
n := StrToInt(ed_n.Text);
d := StrToInt(ed_PrivateKey.Text);
if N < 256 then
begin
ShowMessage('Ошибка! Кодировка невозможна');
Exit;
end;

SimbCount := 1;
Size := 255;

while Size < N do
begin
Size := Size*255;
Inc(SimbCount);
end;
```

```
CodeText :=mem_Cipher.Text;
while CodeText[length(CodeText)] in [#13,#10] do Delete(CodeText, length(CodeText),1);
while (length(CodeText) mod SimbCount)<>0 do
CodeText:= CodeText + ' ';


mem_Decrypt.Text := '';
CodeText := mem_Cipher.Text;
while length(CodeText) > 0 do
begin
mem_Decrypt.Text := mem_Decrypt.Text+CodeAlgoritm( Copy(Codetext,1,SimbCount) , d, n);
Delete(CodeText, 1, SimbCount);
end;

CodeText := mem_Decrypt.Text;
while pos(#1,CodeText)<>0 do Delete(CodeText,pos(#1,CodeText),1);
mem_Decrypt.Text := CodeText;
lbl_BlockLength.Caption := 'Длина блоков: '+IntToStr(SimbCount);
btn_SaveToFile.Enabled := true;
end;

{***}

procedure TForm1.btn_CryptFileOpenClick(Sender: TObject);
begin
if OpenDialog1.Execute then { Display Open dialog box }
mem_Encrypt.Lines.LoadFromFile(OpenDialog1.FileName);
btn_Encrypt.Enabled := true;
mem_Decrypt.Text := '';
mem_Cipher.Text := '';
btn_Decrypt.Enabled := false;
btn_KeyGenerate.Enabled := true;
ed_n.Enabled := false;
ed_PrivateKey.Enabled := false;
ed_n.PasswordChar := #0;
ed_phi.PasswordChar := #0;
ed_PublicKey.PasswordChar := #0;
ed_PrivateKey.PasswordChar := #0;
ed_p.PasswordChar := #0;
ed_q.PasswordChar := #0;
Clear(Form1);
end;
```

```
{***}

procedure TForm1.btn_SaveToFileClick(Sender: TObject);
begin
if SaveDialog1.Execute then
if btn_Encrypt.Enabled = false then
mem_Decrypt.Lines.SaveToFile(SaveDialog1.FileName)
else
mem_Cipher.Lines.SaveToFile(SaveDialog1.FileName);
end;

procedure TForm1.btn_DecryptFileOpenClick(Sender: TObject);
begin
if OpenDialog1.Execute then
mem_Cipher.Lines.LoadFromFile(OpenDialog1.FileName);
mem_Encrypt.Text := '';
mem_Decrypt.Text := '';
btn_Decrypt.Enabled := true;
btn_Encrypt.Enabled := false;
btn_KeyGenerate.Enabled := false;
ed_n.Enabled := true;
ed_PrivateKey.Enabled := true;
ed_n.PasswordChar := '#';
ed_phi.PasswordChar := '#';
ed_PublicKey.PasswordChar := '#';
ed_PrivateKey.PasswordChar := '#';
ed_p.PasswordChar := '#';
ed_q.PasswordChar := '#';
Clear(Form1);
end;

procedure TForm1.btn_ExitClick(Sender: TObject);
begin
Application.Terminate;
end;

procedure TForm1.mem_EncryptChange(Sender: TObject);
begin
if mem_Encrypt.Text <> '' then btn_Encrypt.Enabled := true
else
btn_Encrypt.Enabled := false;
end;

procedure TForm1.btn_AboutClick(Sender: TObject);
```

```
begin
Form2.Visible := true;
end;

end.



unit Unit2;

interface

uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls;

type
TForm2 = class(TForm)
btn_OK: TButton;
lbl_Head: TLabel;
lbl_Faculty: TLabel;
lbl_Topic_1: TLabel;
lbl_Topic_2: TLabel;
lbl_Topic_3: TLabel;
lbl_Student: TLabel;
lbl_Group: TLabel;
lbl_Smolensk: TLabel;
procedure btn_OKClick(Sender: TObject);

private
{ Private declarations }
public
{ Public declarations }
end;

var
Form2: TForm2;
implementation
{$R *.dfm}
procedure TForm2.btn_OKClick(Sender: TObject);
begin
Form2.Visible := False;
end;
end.
```