Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Владимирский государственный университет имени Александра
Григорьевича и Николая Григорьевича Столетовых»

А.О. Кучерик, А.Ю. Лексин, Д.Н. Бухаров, А.Ю. Шагурина

КУРС ЛЕКЦИЙ

ПО ДИСЦИПЛИНЕ «ЗАЩИТА ИНФОРМАЦИИ»

Владимир 2017

УДК 004.65
ББК 73.0
Составители: А.О. Кучерик, А.Ю. Лексин, Д.Н. Бухаров, А.Ю. Шагурина
Рецензент доктор физико-математических наук, профессор Владимирского государственного университета имени Александра Григорьевича и Николая Григорьевича Столетовых – Л.В. Фуров.
Печатается по решению редакционного совета ВлГУ
Курс лекций по дисциплине «Защита информации» / Владим. гос. уни-т имени Александра Григорьевича и Николая Григорьевича Столетовых; А.О. Кучерик, А.Ю. Лексин, Д.Н. Бухаров, А.Ю. Шагурина. – Владимир: Изд-во ВлГУ, 2017. – 104 с.
Рассмотрены основы разделов безопасности информационных систем, защиты информации и криптографии, криптоанализа и защиты информации в интернет сетях.
Предназначены для проведения лекционных занятий по направлениям 02.03.02 «Фундаментальная информатика и информационные технологии» (бакалавриат), 02.03.03 «Математическое обеспечение и администрирование информационных систем» (бакалавриат), 01.03.02 «Прикладная математика и информатика» (бакалавриат), 02.03.01 Математика и компьютерные науки (бакалавриат).
Рекомендовано для формирования профессиональных компетенций в соответствии с ФГОС 3-го поколения.
Ил. 38. Табл. 17. Библиогр.: 11 назв.
УДК 004.65
ББК 73.0

Оглавление

Введение

Информация давно перестала быть просто необходимым для производства материальных ценностей вспомогательным ресурсом – она приобрела ощутимый стоимостный вес, который четко определяется реальной прибылью, получаемой при её использовании, или размерами ущерба, наносимого владельцу информации. Создание технологий и индустрии сбора, переработки, анализа информации и её доставки конечному пользователю порождает ряд сложных проблем. Одной из таких проблем является надежное обеспечение сохранности и установленного статуса информации (актуальности, полноты, непротиворечивости, конфиденциальности), циркулирующей и обрабатываемой в информационно-вычислительных системах и сетях, а также безопасность самих систем и технологий.

Современное развитие информационных технологий и, в частности, технологий *Internet*/Intranet, приводит к необходимости защиты информации, передаваемой в рамках распределенной корпоративной сети, использующей сети открытого доступа. При использовании своих собственных закрытых физических каналов доступа эта проблема так остро не стоит, так как в эту *сеть* закрыт *доступ* посторонним. Однако выделенные каналы может себе позволить далеко не любая компания. Поэтому приходится довольствоваться тем, что есть в распоряжении компании. А есть чаще всего *Internet*. Поэтому приходиться изобретать способы защиты конфиденциальных данных, передаваемых по фактически незащищенной сети.

1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

1.1. Понятия экономической и информационной безопасности. Ключевые вопросы информационной безопасности

Безопасность информационных технологий (ИТ) и систем (ИС) является одной из важнейших составляющих проблемы обеспечения экономической безопасности организации. Переход к новым формам государственного и хозяйственного управления экономикой в России в условиях дефицита и противоречивости правовой базы породил целый комплекс проблем в области защиты данных, информации, знаний и самих ИКТ. Это и своеобразие становления рыночных отношений, и отсутствие обоснованных концепций реформ, и отставание в области применения современных информационных технологий в управлении и производстве. Обострение этих проблем выдвинули на первый план вопросы обеспечения национальной, социальной и корпоративной безопасности, в том числе и в информационной сфере.

При анализе проблематики, связанной с информационной безопасностью (ИБ), необходимо учитывать специфику данного аспекта безопасности, состоящую в том, что информационная *безопасность* есть составная часть разработки, внедрения и эксплуатации информационных систем и технологии – области, развивающейся беспрецедентно высокими темпами.

К сожалению, современная технология программирования не позволяет создавать полностью безошибочные и безопасные программы. Поэтому следует исходить из того, что необходимо создавать надежные системы ИБ с привлечением не стопроцентно надежных программных компонентов (программ).

В принципе, это возможно, но требует соблюдения определенных принципов архитектурного построения программных комплексов и контроля состояния защищенности программно-аппаратного обеспечения, телекоммуникационных устройств и сетей на всем протяжении жизненного *цикла* ИС. [6]

Защищаемая информация – информация, являющаяся предметом собственности и подлежащая защите в соответствии с требованиями правовых документов или требованиями, устанавливаемыми собственниками информации. Собственниками информации могут быть: государство, юридическое лицо, группа физических лиц, отдельное физическое лицо [9].

В последнее время, все большие объемы информации, в том числе и критически важной для отдельных людей, организаций или государств,

хранятся, обрабатываются и передаются с использованием автоматизированных систем (АС) обработки информации.

Рассматривая вопросы безопасности АС, можно говорить о наличии некоторых «желательных» состояний системы, через которые и описывается ее «защищенность» или «безопасность». Безопасность является таким же свойством системы, как надежность или производительность, и в последнее время ей уделяется все большее внимание.

Чтобы указать на причины выхода системы из безопасного состояния, вводятся понятия «угроза» и «уязвимость».

Угроза (безопасности информации) – совокупность условий и факторов, создающих потенциальную или реально существующую опасность нарушения безопасности информации.

Источник угрозы безопасности информации – субъект (физическое лицо, материальный объект или физическое явление), являющийся непосредственной причиной возникновения угрозы безопасности информации. По типу источника угрозы делят на связанные и не- связанные с деятельностью человека. Примерами могут служить удаление пользователем файла с важной информацией и пожар в здании, соответственно. Угрозы, связанные с деятельностью человека, разделяют на угрозы случайного и преднамеренного характера. В последнем случае источник угрозы называют нарушителем или злоумышленником.

Уязвимость (информационной системы) – свойство информационной системы, обуславливающее возможность реализации угроз безопасности обрабатываемой в ней информации.

Если говорить об информационных ресурсах, то реализация угрозы может привести к таким последствиям как получение информации людьми, которым она не предназначена, уничтожение или изменение информации, недоступность ресурсов для пользователей. Таким образом, мы подошли к определению трех основных угроз безопасности.

1. Угроза конфиденциальности (угроза раскрытия) – это угроза, в результате реализации которой, конфиденциальная или секретная информация становится доступной лицу, группе лиц или какой-либо организации, которой она не предназначалась.

2. Угроза целостности – угроза, в результате реализации которой информация становится измененной или уничтоженной.

Необходимо отметить, что и в нормальном режиме работы АС данные могут изменяться и удаляться. Являются ли эти действия легальными или нет, должно определяться политикой безопасности.

Политика безопасности – совокупность документированных правил, процедур, практических приемов или руководящих принципов в

области безопасности информации, которыми руководствуется организация в своей деятельности.

3. Угроза отказа в обслуживании (угроза доступности) – угроза, реализация которой приведет к отказу в обслуживании клиентов АС, несанкционированному использованию ресурсов злоумышленниками по-своему усмотрению.

Безопасность информации – это состояние защищенности информации, при котором обеспечены ее конфиденциальность, доступность и целостность.

Основные составляющие информационной безопасности

- **целостность** – это, в первую очередь, актуальность и непротиворечивость информации, её защищенность от разрушения и несанкционированного изменения: данные и информация, на основе которой принимаются решения, должны быть достоверными, точными и защищенными от возможных непреднамеренных и злоумышленных искажений;
- **конфиденциальность** – засекреченная информация должна быть доступна только тому, кому она предназначена: такую информацию невозможно получить, прочитать, изменить, передать, если на это нет соответствующих прав доступа;
- **доступность** (готовность) – это возможность за приемлемое время получить требуемую информационную услугу: данные, информация и соответствующие службы, автоматизированные сервисы, средства взаимодействия и связи должны быть доступны и готовы к работе всегда, когда в них возникает необходимость.

Защита информации может быть определена как деятельность, направленная на предотвращение утечки защищаемой информации, несанкционированных и непреднамеренных воздействий на защищаемую информацию.

Выделяются следующие направления защиты информации:

- **правовая защита информации** – защита информации правовыми методами, включающая в себя разработку законодательных и нормативных правовых документов (актов), регулирующих отношения субъектов по защите информации, применение этих документов (актов), а также надзор и контроль за их исполнением;

- **техническая защита информации** – защита информации, заключающаяся в обеспечении некриптографическими методами безопасности информации (данных), подлежащей (подлежащих) защите в соответствии с действующим законодательством, с применением технических, программных и программно-технических средств;

- **криптографическая защита информации** – защита информации с помощью ее криптографического преобразования;

- **физическая защита информации** – защита информации путем применения организационных мероприятий и совокупности средств, создающих препятствия для проникновения или доступа неуполномоченных физических лиц к объекту защиты.

Защита информации осуществляется с использованием способов и средств защиты. **Способ защиты информации** – порядок и правила применения определенных принципов и средств защиты информации.

Средство защиты информации – техническое, программное, программно-техническое средство, вещество и (или) материал, предназначенные или используемые для защиты информации.

Отдельно выделяют:

- средства контроля эффективности защиты информации;
- средства физической защиты информации;
- криптографические средства защиты информации. [9]

Современное развитие информационных технологий и, в частности, технологий Internet/Intranet, приводит к необходимости всесторонней защиты информационных технологий и систем, данных и информации, передаваемой в рамках распределенной корпоративной сети, использующей внутренние и внешние сети открытого доступа.

Оценка реальной ситуации сводится в большинстве случаев к ответу на следующие **ключевые вопросы**, составляющие **системную основу обеспечения информационной безопасности:**

надо ли защищаться и что следует защищать?

от кого надо защищаться?

от чего надо защищаться?

как надо защищаться?

что обеспечит эффективность защиты?

во что обойдется разработка, внедрение, эксплуатация, сопровождение и развитие систем защиты? [6]

Надо ли защищаться и что следует защищать?

Ответов на этот вопрос неоднозначен – многое зависит от структуры, области деятельности и целей компании. Для одних первоочередной задачей является предотвращение утечки информации (маркетинговых планов, перспективных разработок, величина и распределение прибыли и т.д.) к конкурентам. Другие могут пренебречь конфиденциальностью своей информации и сосредоточить свое внимание на ее целостности (например, для научно-исследовательских организаций, имеющих открытые Web-серверы). Для провайдера Internet-услуг, оператора связи или общедоступного справочного сервера на первое место поднимается задача обеспечения максимальной доступности и безотказной работы корпоративных информационных систем – первейшей задачей является именно обеспечение безотказной работы всех (или наиболее важных)

узлов своей информационной системы. Расставить такого рода приоритеты и определить необходимость и объекты защиты можно только в результате анализа деятельности компании.

При интеграции индивидуальных и корпоративных информационных систем и ресурсов в единую информационную инфраструктуру определяющим фактором является обеспечение должного уровня информационной безопасности для каждого субъекта, принявшего решение войти в это пространство. В едином информационном пространстве должны быть созданы все необходимые предпосылки для установления подлинности пользователя (субъекта), подлинности содержания и подлинности сообщения (т.е. созданы механизмы и инструмент аутентификации).

Таким образом, должна быть создана система информационной безопасности, которая включает необходимый комплекс мероприятий и технических решений по защите:

- от нарушения функционирования информационного пространства путем исключения воздействия на информационные каналы и ресурсы;
- от несанкционированного доступа к информации путем обнаружения и ликвидации попыток использования ресурсов информационного пространства, приводящих к нарушению его целостности;
- от разрушения встраиваемых средств защиты с возможностью доказательства неправомочности действий пользователей и обслуживающего персонала;
- от внедрения программных "вирусов" и "закладок" в программные продукты и технические средства.

Особо следует отметить задачи обеспечения безопасности разрабатываемых и модифицируемых систем в интегрированной информационной среде, т. к. в процессе модификации неизбежно возникновение дополнительных ситуаций незащищенности системы. Для решения этой проблемы наряду с общими методами и технологиями следует отметить введение ряда требований к разработчикам, создания регламентов внесения изменений в системы, а также использования специализированных средств. [6]

От кого надо защищаться?

В абсолютном большинстве случаев ответом на этот вопрос является фраза: "Как от кого - конечно, от хакеров!". Исследования показали, что, по мнению большинства российских предпринимателей, основная опасность исходит от внешних злоумышленников, которые проникают в компьютерные системы банков и корпораций, перехватывают управление бизнес-процессами, "взламывают" сайты, запускают "троянских коней".

Такая опасность существует и нельзя её недооценивать. В системах информационной защиты обязательно должны быть соответствующие модули защиты от внешних угроз подобного рода.

Но эта опасность часто преувеличена. До 75-85% всех компьютерных угроз и преступлений связаны с внутренними нарушениями, т.е. осуществляются действующими или уволенными сотрудниками компании. По исследованиям 2013 года в 82% случаев источником реальных атак были сотрудники компаний. Для сравнения: хакеры, атакующие корпоративные сети извне, оказывались источником атак в 73% случаев.

Однако самая большая опасность может исходить не просто от уволенных или обиженных рядовых сотрудников (например, операторов различных информационных подсистем), а от тех, кто облечён большими полномочиями и имеет доступ к широкому спектру самой различной информации. Обычно это сотрудники ИТ-отделов (аналитики, разработчики, системные администраторы), которые знают пароли ко всем системам, используемым в организации. Их квалификация, знания и опыт, используемые во вред, могут привести к очень большим проблемам. Кроме того, таких злоумышленников очень трудно обнаружить, поскольку они обладают достаточными знаниями о системе защиты ИС компании, чтобы обойти используемые защитные механизмы и при этом остаться "невидимыми". [2]

От чего надо защищаться?

Во-первых, это вирусы (Virus, Worm) и всевозможные виды практически бесполезной информации, рассылаемой абонентам электронной почты (Spam). По различным данным в 2013 году вирусным и спамовым атакам было подвержено 85-90 % компаний во всем мире. Далее следует назвать программы типа "троянский конь" (Trojan Horse), которые могут быть незаметно для владельца установлены на его компьютер и так же незаметно функционировать на нем. Следующим распространенным типом атак являются действия, направленные на выведение из строя того или иного узла сети. Эти атаки получили название "отказа в обслуживании" (Denial of Service – DoS), на сегодняшний день известно более сотни различных вариантов этих действий. Выше отмечалось, что выведение из строя узла сети на несколько часов может привести к очень серьезным последствиям. Например, выведение из строя сервера транзакционной системы крупной корпорации или банка приведет к невозможности осуществления платежей и, как следствие, к большим прямым и косвенным финансовым и рейтинговым потерям.

Укажем ещё один существенный источник угроз, который с точки зрения размера ущерба может быть отнесён к одному из самых распространённых в России – непреднамеренные ошибки пользователей

ИС, операторов, системных администраторов и других лиц, обслуживающих информационные системы. Иногда такие ошибки являются угрозами (неправильно введенные данные, ошибка в программе), а иногда они создают уязвимости, которыми могут воспользоваться злоумышленники – таковы обычно ошибки администрирования и предоставления доступа. [8]

Как надо защищаться?

Наиболее простой способ – купить новейшие рекламируемые средства защиты, установить у себя в организации, не утруждая себя обоснованием её полезности и эффективности. Если компания богата, то она может позволить себе этот путь. Однако истинный руководитель должен системно оценивать ситуацию и правильно расходовать средства.

Во всем мире сейчас принято строить комплексную систему защиты информации и информационных систем в несколько этапов – на основе формирования концепции и программы информационной безопасности, имея в виду в первую очередь взаимосвязь её основных понятий.

Первый этап – информационное обследование предприятия – самый важный. Именно на этом этапе определяется, от чего, в первую очередь, необходимо защищаться компании. Вначале строится так называемая модель нарушителя, которая описывает вероятный облик злоумышленника, т. е. его квалификацию, имеющиеся средства для реализации тех или иных атак, обычное время действия и т. п. На этом этапе можно получить ответ на два вопроса, которые были заданы выше: "Зачем и от кого надо защищаться?" На этом же этапе выявляются и анализируются уязвимые места и возможные пути реализации угроз безопасности, оценивается вероятность атак и ущерб от их осуществления.

По результатам этапа вырабатываются рекомендации по устранению выявленных угроз, правильному выбору и применению средств защиты. На этом этапе может быть рекомендовано не приобретать достаточно дорогие средства защиты, а воспользоваться имеющимися в распоряжении. Например, в случае использования в небольшой компании мощного маршрутизатора можно рекомендовать воспользоваться встроенными в него защитными функциями, а не приобретать более дорогой межсетевой экран (Firewall).

Наряду с анализом существующих в компании конкретных средств защиты следует разработать общую и частные политики в области информационной безопасности и совокупности организационно-распорядительных мер и документов, а также методологий и технических решений, являющихся основой для создания инфраструктуры информационной безопасности.

Эти документы, основанные на международном законодательстве и законах Российской федерации и нормативных актах, дают необходимую

правовую базу службам безопасности и отделам защиты информации для проведения всего спектра защитных мероприятий, взаимодействия с внешними организациями, привлечения к ответственности нарушителей и т. п.

Следующим этапом построения комплексной системы информационной безопасности служит приобретение, установка и настройка рекомендованных на предыдущем этапе средств и механизмов защиты информации. К таким средствам можно отнести системы защиты информации от несанкционированного доступа, системы криптографической защиты, межсетевые экраны, средства анализа защищенности и другие.

Для правильного и эффективного применения установленных средств защиты необходим квалифицированный персонал.

С течением времени имеющиеся средства защиты устаревают, выходят новые версии систем обеспечения информационной безопасности, постоянно расширяется список найденных слабых мест и атак, меняется технология обработки информации, изменяются программные и аппаратные средства, приходит и уходит персонал компании. Поэтому необходимо периодически пересматривать разработанные организационно-распорядительные документы, проводить обследование ИС или ее подсистем, обучать новый персонал, обновлять средства защиты. [6]

1.2. ИДЕНТИФИКАЦИЯ, АУТЕНТИФИКАЦИЯ, УПРАВЛЕНИЕ ДОСТУПОМ КАК ЗАЩИТА ОТ НЕСАНКЦИОНИРОВАННОГО ДОСТУПА

Защита информации от несанкционированного доступа (НСД) – защита информации, направленная на предотвращение получения защищаемой информации заинтересованными субъектами с нарушением установленных нормативными и правовыми документами (актами) или обладателями информации прав или правил разграничения доступа к защищаемой информации.

Для защиты от НСД, как правило, используется идентификация, аутентификация и управление доступом. В дополнение к перечисленным, могут применяться и другие методы.

Идентификация – присвоение пользователям идентификаторов (уникальных имен или меток) под которыми система «знает» пользователя.

Аутентификация – установление подлинности – проверка принадлежности пользователю предъявленного им идентификатора.

Управление доступом – метод защиты информации путем регулирования использования всех ресурсов системы на основе прав доступа.

Обычно выделяют **3 группы методов** аутентификации.

1. Аутентификация **по наличию у пользователя уникального объекта заданного типа**. Иногда этот класс методов аутентификации называют по-английски “I have” («у меня есть»). В качестве примера можно привести аутентификацию с помощью смарт-карт или электронных USB-ключей.

2. Аутентификация, основанная на том, что пользователю известна некоторая **конфиденциальная информация** – “I know” («я знаю»). Например, аутентификация по паролю. Более подробно парольные системы рассматриваются далее в этом разделе.

3. Аутентификация пользователя по его **собственным уникальным характеристикам** – “I am” («я есть»). Эти методы также называются биометрическими. Биометрические методы аутентификации делят на **статические** и **динамические**.

Примеры аутентификации по статическим признакам – это проверка отпечатка пальца, рисунка радужной оболочки глаз, геометрии кисти руки, сравнение с фотографией и т. д. Достоинством этих методов является достаточно высокая точность. Но надо отметить, что подобные методы, как правило, требуют наличия специализированного оборудования (например, специальные сканеры) и имеют ограниченную область применения (например, при аутентификации по отпечатку пальца, из-за грязи на руке человек может не пройти аутентификацию, т. е. подобные методы неприменимы на стройках и на многих производствах).

Примеры динамической аутентификации – аутентификация по голосу (при произнесении заранее определенной фразы или произвольного текста), аутентификация по «клавиатурному почерку» (проверяются особенности работы пользователя на клавиатуре, такие как время задержки при нажатии клавиш в различных сочетаниях) и т. д.

Нередко используются комбинированные схемы аутентификации, объединяющие методы разных классов.

Аутентификация может быть **односторонней**, когда одна сторона аутентифицирует другую (например, сервер проверяет подлинность клиентов), и **двусторонней**, когда стороны проводят взаимную проверку подлинности.

Также аутентификация может быть **непосредственной**, когда в процедуре аутентификации участвуют только две стороны, или с участием доверенной стороны. В последнем случае в процессе аутентификации участвуют не только стороны, проверяющие подлинность друг друга, но и другая или другие, вспомогательные. Эту третью сторону иногда называют

сервером аутентификации (англ. «authentication server») или арбитром (англ. «arbitrator»).

Наиболее распространенными на данный момент являются парольные системы аутентификации. Определим ряд понятий, использующихся при описании подобных систем. [9]

Парольные системы аутентификации

Идентификатор пользователя – уникальная информация, позволяющая различить отдельных пользователей парольной системы (провести идентификацию).

Пароль пользователя – секретная информация, известная только пользователю (и возможно – системе), которая используется для прохождения аутентификации. В зависимости от реализации системы, **пароль** может быть **одноразовым** или **многоразовым**. Системы с одноразовыми паролями являются более надежными. В них исключаются некоторые риски связанные с перехватом паролей – пароль действителен только на одну сессию и, если легальный пользователь его уже задействовал, нарушитель не сможет такой пароль повторно использовать.

Учетная запись пользователя – совокупность идентификатора, пароля и, возможно, дополнительной информации, служащей для описания пользователя.

Учетные записи хранятся в базе данных парольной системы.

Парольная система – это программный или программно-аппаратный комплекс, реализующий функции идентификации и аутентификации пользователей компьютерной системы путем проверки паролей.

Компьютерная атака – целенаправленное несанкционированное воздействие на информацию, на ресурс автоматизированной информационной системы или получение несанкционированного доступа к ним с применением программных или программно-аппаратных средств.

Рассмотрим некоторые **рекомендации** по **администрированию парольной системы**, использующей многоразовые пароли.

1. Задание минимальной длины используемых в системе паролей. Это усложняет атаку путем подбора паролей. Как правило, рекомендуют устанавливать минимальную длину в 6-8 символов.

2. Установка требования использовать в пароле разные группы символов – большие и маленькие буквы, цифры, специальные символы.

3. Периодическая проверка администраторами безопасности качества используемых паролей путем имитации атак, таких как подбор паролей «по словарю».

4. Установление максимального и минимального сроков жизни пароля, использование механизма принудительной смены старых паролей.

5. Ограничение числа неудачных попыток ввода пароля (блокирование учетной записи после заданного числа неудачных попыток войти в систему).

6. Ведение журнала истории паролей, чтобы пользователи, после принудительной смены пароля, не могли вновь выбрать себе старый, возможно скомпрометированный пароль. [9]

1.3. МОДЕЛИ БЕЗОПАСНОСТИ

Важным этапом процесса обеспечения безопасности АС является разработка политики безопасности. Формальное описание политики безопасности производится в рамках **модели безопасности**.

Большинство моделей безопасности оперируют терминами «**сущность**», «**субъект**», «**объект**».

Сущность – любая именованная составляющая защищаемой АС.

Субъект – активная сущность, которая может инициировать запросы ресурсов.

Объект – пассивная сущность, используемая для хранения или получения информации.

Доступ – взаимодействие между субъектом и объектом, перенос информации между ними.

Два фундаментальных типа доступа: **чтение** – операция, результатом которой является перенос информации от объекта к субъекту; **запись** – операция, результатом которой является перенос информации от субъекта к объекту.

Также предполагается существование **монитора безопасности объектов**, т. е. такого субъекта, который будет активизироваться при любом обращении к объектам, может различать (на базе определенных правил) легальные и несанкционированные обращения, и разрешать только легальный доступ.

Выделяются три основных класса моделей политики безопасности: **дискреционные**, **мандатные** и **ролевые**.

Основу **дискреционной** (избирательной) политики безопасности составляет дискреционное управление доступом, которое характеризуется следующими свойствами:

- все субъекты и объекты должны быть идентифицированы;
- права доступа субъекта к объекту системы определяются на основании некоторого внешнего по отношению к системе правила.

Правила дискреционного управления доступом часто задаются матрицей доступов. В подобной матрице строки соответствуют субъектам системы, столбцы – объектам, элементы матрицы описывают права доступа для соответствующей пары «субъект - объект».

Одной из наиболее известных дискреционных моделей является **модель Харрисона-Рузо-Ульмана**, часто называемая **матричной** моделью.

Основу **мандатной** политики безопасности составляет мандатное управление доступом, которое подразумевает, что:

- все субъекты и объекты должны быть идентифицированы;
- задан линейно упорядоченный набор меток секретности;
- каждому объекту системы присвоена метка секретности, определяющая ценность содержащейся в нем информации – его уровень секретности;
- каждому субъекту системы присвоена метка секретности, определяющая уровень доверия к нему – его уровень доступа;
- решение о разрешении доступа субъекта к объекту принимается исходя из типа доступа и сравнения метки субъекта и объекта.

Чаще всего мандатную политику безопасности описывают в терминах модели **Белла-ЛаПадула**.

Управление доступом, основанное на **ролях**, оперирует в терминах «роль», «пользователь», «операция». Вся информация рассматривается как принадлежащая организации (а не пользователю, ее создавшему).

Решения о разрешении или отказе в доступе принимаются на основе информации о той функции (роли), которую пользователь выполняет в организации. Роль можно понимать, как множество действий, которые разрешены пользователю для выполнения его должностных обязанностей. Администратор описывает роли и авторизует пользователей на выполнение данной роли. Таким образом, ролевые модели содержат как признаки мандатных, так и признаки дискреционных моделей. [8]

Процесс построения и оценки системы обеспечения безопасности. Стандарт ISO/IEC 15408

Одним из наиболее распространенных современных стандартов в области информационной безопасности является международный стандарт ISO/IEC 15408. В 2002 году этот стандарт был принят в России как ГОСТ Р ИСО/МЭК 15408-2002 «Информационная технология. Методы обеспечения безопасности. Критерии оценки безопасности информационных технологий», часто называемый в литературе «Общие критерии».

Стандарт разработан таким образом, чтобы удовлетворить потребности трех групп специалистов: разработчиков, экспертов по сертификации и пользователей объекта оценки.

«Общие критерии» предусматривают наличие двух типов требований безопасности – функциональных и доверия. Функциональные требования относятся к сервисам безопасности, таким как управление доступом, аудит и т. д. Требования доверия к безопасности относятся к

технологии разработки, тестированию, анализу уязвимостей, поставке, сопровождению, эксплуатационной документации и т. д.[9]

2. ОСНОВЫ КРИПТОГРАФИИ
2.1. ОСНОВНЫЕ ПОНЯТИЯ. КЛАССИФИКАЦИЯ ШИФРОВ

Исторически криптография (с греческого – «тайнопись») зародилась как способ скрытой передачи сообщений без сокрытия самого факта их передачи. Для этой цели сообщение, написанное с использованием какого-либо общепринятого языка, преобразовывалось под управлением дополнительной информации, называемой ключом. Результат преобразования, называемый криптограммой, содержит исходную информацию в полном объеме, однако последовательность знаков в нем внешне представляется случайной и не позволяет восстановить исходную информацию без знания ключа.

Процедура преобразования называется шифрованием, обратного преобразования – расшифровыванием.

Сейчас криптографией принято называть науку о математических методах обеспечения конфиденциальности и аутентичности (целостности и подлинности) информации. Задачей исследования методов преодоления криптографической защиты занимается криптоанализ. Для обозначения совокупности криптографии и криптоанализа используется термин «криптология».

Попытку раскрытия конкретного шифра с применением методов криптоанализа называют криптографической атакой на этот шифр. Криптографическую атаку, в ходе которой раскрыть шифр удалось, называют взломом или вскрытием. [9]

Выделяют 4 основных и 3 дополнительных метода криптоанализа, предполагая знание криптоаналитиком алгоритма шифра:

Основные методы криптоанализа:

1.Атака на основе шифротекста

2.Атака на основе открытых текстов и соответствующих шифротекстов

3.Атака на основе подобранного открытого текста (возможность выбрать текст для шифрования)

4.Атака на основе адаптивно подобранного открытого текста

Дополнительные методы криптоанализа:

1.Атака на основе подобранного шифротекста

2.Атака на основе подобранного ключа

3.Бандитский криптоанализ [1]

Несмотря на то, что шифры применялись еще до нашей эры, как научное направление современная криптография относительно молода.

Одной из важнейших работ в данной области является статья Клода Шеннона «Теория связи в секретных системах», опубликованная в открытой печати в 1949 году. На рис. 1 изображена предложенная Шенноном схема секретной системы [9].

На стороне отправителя имеются два источника информации – источник сообщений и источник ключей. Источник ключей выбирает из множества всех возможных ключей один ключ K, который будет использоваться в этот раз. Ключ передается отправителю и получателю сообщения таким образом, что его невозможно перехватить.

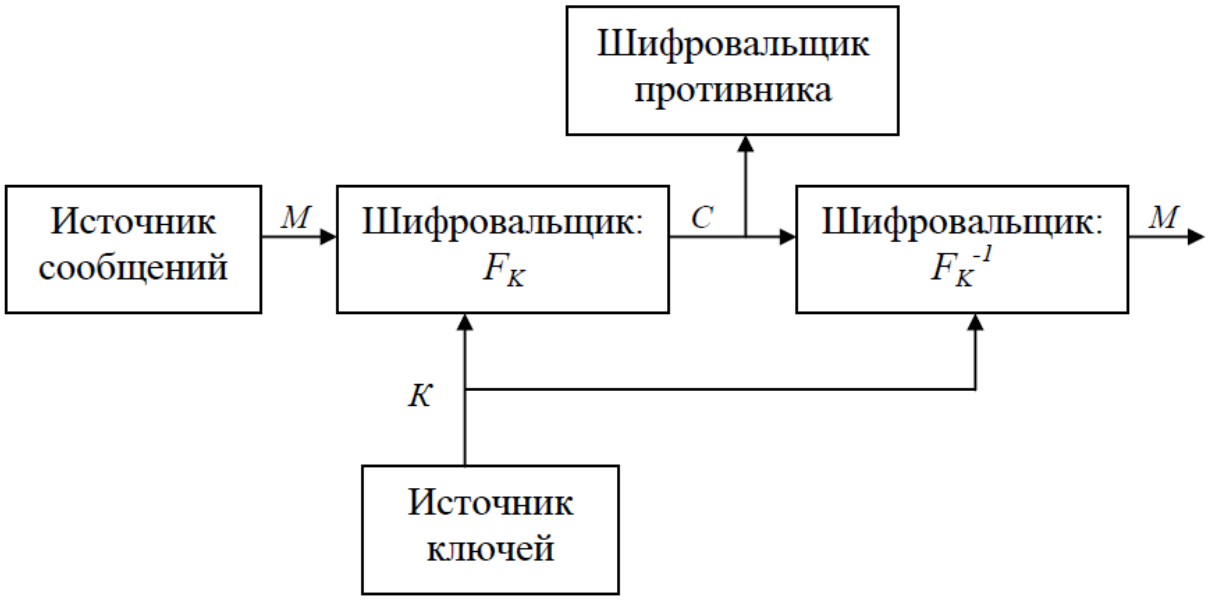

Рис. 1. Схема секретной системы

Отображение FK, примененное шифровальщиком к сообщению M дает криптограмму C:

$C = F_K M$.

В связи с тем, что получатель должен иметь возможность восстановить сообщение M из криптограммы C при известном ключе K, отображение FK должно иметь единственное обратное отображение F^{K-1}, такое что:

$M = F_K^{-1} C$.

Секретная система (или в современной терминологии – шифр) определяется как семейство однозначно обратимых отображений множества возможных сообщений во множество криптограмм.

Выбор ключа K определяет, какой именно элемент – F_K – будет использоваться. Предполагается, что противнику известна используемая система, т. е. семейство отображений $\{F_i | i=1..N\}$ и вероятности выбора различных ключей. Однако он не знает, какой именно ключ выбран, и остальные возможные ключи столь же важны для него, как и истинный.

Процесс расшифровывания сообщения для легального получателя информации состоит в применении криптографического отображения, обратного по отношению к отображению, использованному при шифровании.

Процесс расшифровки для противника представляет собой попытку определить сообщение (или конкретный ключ), имея в распоряжении только криптограмму и априорные вероятности различных ключей и сообщений.

Существуют шифры, для которых любой объем перехваченной информации недостаточен для того, чтобы найти шифрующее отображение. Шифры такого типа называются безусловно стойкими.

Шифры другого типа характеризуются тем, что при определенном объеме перехваченных данных определить ключ (или расшифровать сообщение без знания ключа) становится теоретически возможно.

Шифры такого типа называются условно стойкими. Их стойкость основана на высокой вычислительной сложности «взлома» шифра. Большинство применяемых сейчас шифров относятся к этому типу.

Доказано, что безусловно стойкие шифры существуют. Но для их построения необходимо использовать равновероятный случайный ключ, имеющий длину, равную длине сообщения.

Рассмотрим, какими же свойствами должен обладать хороший шифр.

Во-первых, шифрование и расшифровывание должно осуществляться достаточно быстро в тех условиях, в которых применяется шифр (с использованием ЭВМ, при шифровании вручную и т. п.). Во-вторых, шифр должен надежно защищать сообщение, т. е. быть стойким к раскрытию.

Криптостойкость – стойкость шифра к раскрытию методами криптоанализа. Она определяется вычислительной сложностью алгоритмов, применяемых для атаки на шифр. Вычислительная сложность измеряется временной и емкостной сложностями.

Для определения сложности алгоритма с конкретной задачей связывается число, называемое размером задачи, которое характеризует количество входных данных. Например, для задачи умножения чисел размером может быть длина наибольшего из сомножителей.

Временная сложность (или просто сложность) – это время, затрачиваемое алгоритмом для решения задачи, рассматриваемое как функция от размера задачи.

Нередко сложность измеряют количеством некоторых элементарных операций. Емкостная сложность – объем памяти, необходимой для хранения полученных в ходе работы данных, как функция от размера задачи.

Очень важное требование к стойкому шифру было сформулировано в криптографом Огюстом Керкгоффсом: при оценке надежности шифрования необходимо предполагать, что противник знает все об используемой системе шифрования, кроме применяемых ключей.

Данное правило отражает важный принцип организации защиты информации: защищенность системы не должна зависеть от секретности долговременных элементов (т. е. таких элементов, которые невозможно было бы быстро изменить в случае утечки секретной информации).

Рассмотрим классификации шифров по разным признакам. По типу преобразований шифры можно разделить на следующие группы:

- шифры замены (подстановки);
- шифры перестановки;
- шифры гаммирования;
- шифры на основе аналитических преобразований.

При этом надо учитывать, что некоторые современные шифры совместно используют преобразования различных типов.

Шифры замены (подстановки): преобразование следующую через одну после нее. В случае русского алфавита, «а» меняется на «в», «б» на «г» и т. д. Алфавит «замыкался», поэтому «я» надо было заменять на «б». В качестве ключа в данном случае выступает число, на которое надо «сдвигать» символ алфавита, в нашем примере – 2. К достоинству таких шифров относится простота преобразования. Но они легко взламываются путем сравнения частоты появления различных символов в естественном языке и криптограмме. [9]

Рассмотрим, как используют шифр Цезаря.

Прежде всего, выбирается нормативный алфавит, т.е. набор символов, которые будут использоваться при составлении сообщений, требующих зашифровки. Допустим, это будут прописные буквы русского алфавита (исключая буквы “Ё” и “Ъ”) и пробел. Таким образом, наш нормативный алфавит состоит из 32 символов. Затем выбирается алфавит шифрования и устанавливается взаимно однозначное соответствие между символами нормативного алфавита и символами алфавита шифрования. Алфавит шифрования может состоять из произвольных символов, в том числе и из символов нормативного алфавита.

Чтобы зашифровать исходное сообщение, каждый символ открытого текста заменяется соответствующим ему символом алфавита шифрования (табл. 1).

Таблица 1

Соотношение алфавитов при шифровании

Нормативный алфавит	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	...
Алфавит шифрования	Н	К	А	Л	З	Т	П	И	О	Р	Б	Г	...

Зашифруем, например, слово “звезда”. Если использовать алфавиты, приведенные на таблица 1, то получится следующее:

Исходное сообщение: З В Е З Д А

Шифрованный текст: И А Т И З Н

Алгоритмически метод моноалфавитной подстановки можно представить, как числовые преобразования символов исходного текста. Для этого каждой букве нормативного алфавита ставится в соответствие некоторое число, называемое числовым эквивалентом этой буквы.

Например, для букв русского алфавита и пробела это выглядит так, как показано на таблице 2.

Таблица 2

Буквы алфавита и числовые эквиваленты

Нормативный алфавит	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
Числовые эквиваленты	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Нормативный алфавит	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ы	Ь	Э	Ю	Я	“_”
Числовые эквиваленты	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Нормативный алфавит	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
Числовые эквиваленты	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Нормативный алфавит	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ы	Ь	Э	Ю	Я	“_”
Числовые эквиваленты	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

Моноалфавитные подстановки можно описать выражением:

$E_i=(M_i + S_i) \bmod L$,

где E_i, M_i - числовые эквиваленты символов алфавита шифрования и нормативного алфавита соответственно, S_i - коэффициент сдвига, L - мощность алфавита.

Цезарь использовал величину сдвига S=3, но, конечно, можно использовать любое целое S: $1 \leq S \leq (L-1)$.

Зашифруем, например, текст “ШИФР_ЦЕЗАРЯ”, используя коэффициент сдвига S = 2.

Открытый текст: Ш И Ф Р _ Ц Е З А Р Я

Шифрованный текст: Ы К Ц Т Б Ш З Й В Т А

При использовании многоалфавитных подстановок, учитываются дополнительные параметры (например, положение преобразуемого символа в тексте) и в зависимости от них символ исходного алфавита может заменяться на один из нескольких символов алфавита шифротекста. Например, нечетные символы сообщения заменяются по одному правилу, четные – по другому. [7]

Шифр Виженера состоит из последовательности нескольких шифров Цезаря с различными значениями сдвига. Для зашифровывания может использоваться таблица алфавитов, называемая tabula recta или квадрат (таблица) Виженера. Применительно к латинскому алфавиту таблица Виженера составляется из строк по 26 символов, причём каждая следующая строка сдвигается на несколько позиций. (рис.2) Таким образом, в таблице получается 26 различных шифров Цезаря.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Рис.2. Квадрат Виженера

На каждом этапе шифрования используются различные алфавиты, выбираемые в зависимости от символа ключевого слова. Например, предположим, что исходный текст имеет такой вид:

ATTACKATDAWN

Человек, посылающий сообщение, записывает ключевое слово («LEMON») циклически до тех пор, пока его длина не будет соответствовать длине исходного текста:

LEMONLEMONLE

Первый символ исходного текста A зашифрован последовательностью L, которая является первым символом ключа. Первый символ L шифрованного текста находится на пересечении строки L и столбца A в таблице Виженера. Точно так же для второго символа исходного текста используется второй символ ключа; то есть второй символ шифрованного текста X получается на пересечении строки E и столбца T. Остальная часть исходного текста шифруется подобным способом.

Исходный текст: ATTACKATDAWN
Ключ: LEMONLEMONLE
Зашифрованный текст: LXFOPVEFRNHR

Расшифровывание производится следующим образом: находим в таблице Виженера строку, соответствующую первому символу ключевого слова; в данной строке находим первый символ зашифрованного текста. Столбец, в котором находится данный символ, соответствует первому символу исходного текста. Следующие символы зашифрованного текста расшифровываются подобным образом.

Если n – количество букв в алфавите, m_j – буквы открытого текста, k_j – буквы ключа, то шифрование Виженера можно записать следующим образом: $c_j=m_j+k_j \pmod n$.

И расшифровывание:

$m_j=c_j-k_j \pmod n$.

В компьютере такая операция соответствует сложению кодов ASCII символов сообщения и ключа по некоторому модулю. Кажется, что если таблица будет более сложной, чем циклическое смещение строк, то шифр станет надежнее. Это действительно так, если ее менять чаще, например, от слова к слову. Но составление таких таблиц, представляющих собой латинские квадраты, где любая буква встречается в строке или столбце один раз, трудоемко и его стоит делать лишь на ЭВМ. Для ручного же многоалфавитного шифра полагаются лишь на длину и сложность ключа, используя приведенную таблицу, которую можно не держать в тайне, а это упрощает шифрование и расшифровывание.

Шифры перестановок: шифрование заключается в том, что символы исходного текста переставляются по определенному правилу в пределах блока этого текста. При достаточной длине блока и сложном, неповторяющемся порядке перестановки, можно достичь приемлемой стойкости шифра.

Шифр перестановок – заключается в перестановках структурных элементов шифруемого блока данных – битов, символов, цифр. Шифр простой перестановки является одним из простейших блочных шифров. Исходное сообщение (открытый текст) разбивается на блоки равного размера. Если размер текста не кратен размеру блока, в конец сообщения дописывают пробелы, чтобы число знаков в сообщении было кратно размеру блока. Далее в каждом из блоков (независимо от остальных блоков) символы открытого текста меняются местами. Правило выполнения перестановки и является ключом к шифру. Данный вариант шифра не является стойким, поскольку размер ключа существенно меньше размера текста. Кроме того, в зашифрованном тексте сохраняются статистические зависимости между символами. Например, частота появлений отдельных символов в открытом тексте и шифртексте остается постоянной, что облегчает подбор ключа (т.е. взлом шифра). [7]

Пример использования шифра перестановок

Исходное сообщение "МАМА МЫЛА РАМУ"

Перестановка:

Вместо Подставить

1 2

2 4

3 1

4 3

Размер блока: 4 символа (в перестановке участвуют 4 последовательных символа). Длина текста: 14 символов (размер сообщения не является кратным размеру блока). Разбиение открытого текста на блоки (с добавлением пробелов, для наглядности пробелы изображены символами подчеркивания):

М А М А _ М Ы Л А _ Р А М У _ _

(блок 1) (блок 2) (блок 3) (блок 4)

Перестановка в блоке 1:

Открытый текст: Символ М А М А

Позиция 1 2 3 4

Шифртекст: Символ А А М М

Позиция 2 4 1 3

Перестановка:

А А М М М Л _ Ы _ А А Р У _ М _

(блок 1) (блок 2) (блок 3) (блок 4)

Шифртекст:

"ААММММЛ Ы ААР У М "

Для расшифровки сообщения необходимо применить обратную перестановку.

Прямая перестановка: 1→2, 2→4, 3→1, 4→3

Обратная перестановка: 1→3, 2→1, 3→4, 4→2

Шифрование гаммированием о заключается в том, что символы шифруемого текста складываются с символами некоторой случайной последовательности, называемой гаммой шифра или ключевой гаммой.

$C = M \oplus K$.

Подобное преобразование также называют гаммированием, а ключ K – ключевой гаммой.

Стойкость шифрования определяется длиной (периодом) неповторяющейся части гаммы шифра, а также сложностью предугадывания следующих элементов гаммы по предыдущим.

2.2. СИММЕТРИЧНЫЕ ШИФРЫ

2.2.1. Шифр сеть Фейстеля

Сеть Фейстеля – разновидность блочного шифра с определенной многократно повторяющейся (итерированной) структурой, называющейся ячейкой Фейстеля. При переходе от одной ячейки к другой меняется ключ, причём выбор ключа зависит от конкретного алгоритма. Операции шифрования и расшифрования на каждом этапе очень просты, и при определённой доработке совпадают, требуя только обратного порядка используемых ключей. Шифрование при помощи данной конструкции легко реализуются как на программном уровне, так и на аппаратном. Большинство современных блочных шифров используют сеть Фейстеля в качестве основы.

Шифрование (см рис. 3)

Рассмотрим случай, когда мы хотим зашифровать некоторую информацию, представленную в двоичном виде в компьютерной памяти (например, файл) или электронике, как последовательность нулей и единиц.

- Вся информация разбивается на блоки фиксированной длины. В случае, если длина входного блока меньше, чем размер, который шифруется заданным алгоритмом, то блок удлиняется каким-либо способом. Как правило длина блока является степенью двойки, например: 64 бита, 128 бит. Далее будем рассматривать операции происходящие только с одним блоком, так как с другими в процессе шифрования выполняются те же самые операции.
- Выбранный блок делится на два равных подблока – «левый» (L_0) и «правый» (R_0).
- «Левый блок» L_0 видоизменяется функцией $f(L_0, K_1)$ в зависимости от ключа K_1 , после чего он складывается по модулю 2 с «правым блоком» R_0 .

- Результат сложения присваивается новому левому подблоку L_1, который будет половиной входных данных для следующего раунда, а «левый блок» L_0 присваивается без изменений новому правому подблоку R_1 (см. схему), который будет другой половиной.
- После чего операция повторяется N-1 раз, при этом при переходе от i-го к i+1-му этапу могут меняться ключи (K_i на K_{i+1}) по какому-либо правилу, где N – количество раундов в заданном алгоритме.

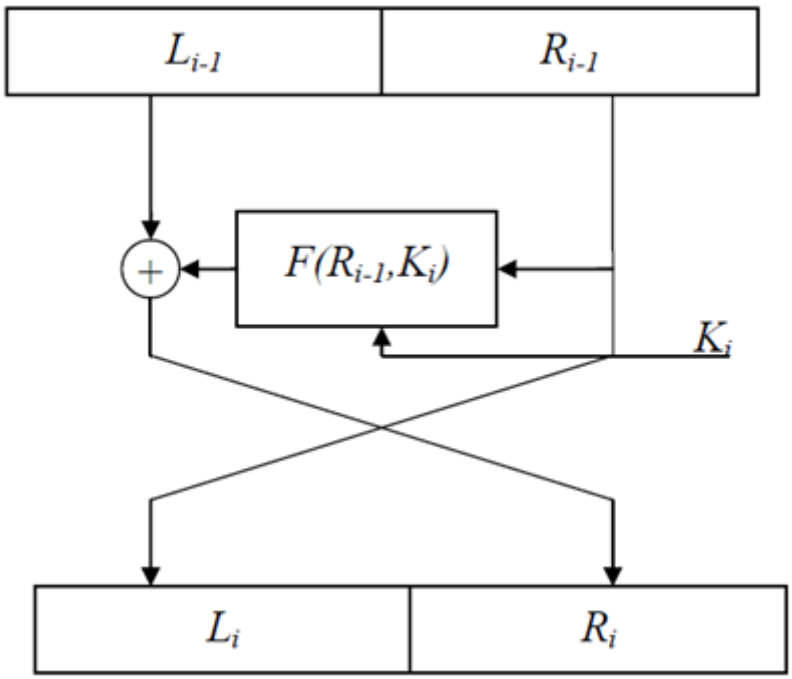

Рис.3. Прямое преобразование Фейстеля

Расшифрование (см рис. 4)

Расшифровка информации происходит так же, как и шифрование, с тем лишь исключением, что ключи идут в обратном порядке, т.е. не от первого к N-ному, а от N-го к первому.

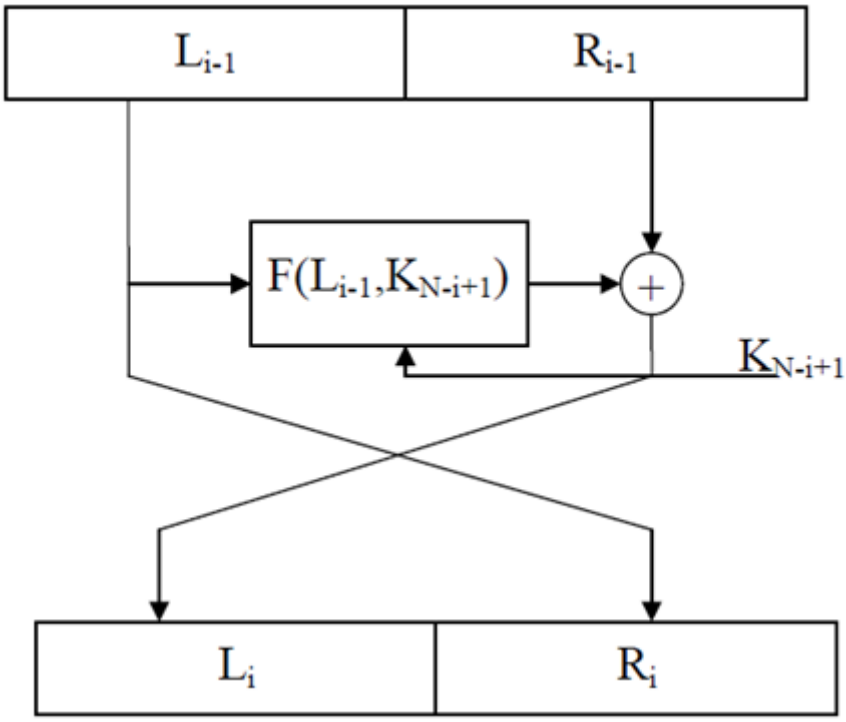

Рис.4. Обратное преобразование Фейстеля

Алгоритмическое описание

- блок открытого текста делится на 2 равные части (L_0 и R_0).
- в каждом раунде вычисляется (i – номер раунда)

$L_i = R_{i-1} \oplus f(L_{i-1}, K_{i-1})$

$R_i = L_{i-1}$,

где f – некоторая функция, а K_i – ключ i-го раунда. Результатом выполнения n раундов является (L_n, R_n) . Но обычно в n-ом раунде перестановка L_n и R_n не производится, что позволяет использовать ту же процедуру и для расшифрования, просто инвертировав порядок использования раундовой ключевой информации:

$L_{i-1} = R_i \oplus f(L_i, K_{i-1})$, $R_{i-1} = L_i$.

Небольшим изменением можно добиться и полной идентичности процедур шифрования и дешифрования. Одно из преимуществ такой модели – обратимость алгоритма независимо от используемой функции f, и она может быть сколь угодно сложной. [7]

Функции, используемые в сетях Фейстеля

В своей работе «Криптография и Компьютерная безопасность» Хорст Фейстель описывает два различных блока преобразований (функций $f((L_i, K_i))$ – блок подстановок(S-Блок) и блок перестановок(P-Блок). Можно показать, что любое двоичное преобразование над двоичным блоком фиксированной длины, сводятся к S-блоку, но на практике в силу сложности строения n-разрядного S-блока при больших n, применяют

более простые конструкции. Термин «блок» в оригинальной статье используется вместо функции вследствие того, что речь идёт о блочном шифре и предполагалось, что S- и P-блоки будут цифровыми микросхемами (цифровыми блоками).

S-Блок(S-box)

Блок подстановок (S-Блок) состоит из дешифратора, преобразующего n-разрядный двоичный сигнал в одноразрядный сигнал по основанию 2^n, системы коммутаторов внутренних соединений (всего соединений $2^n!$) и шифратора, переводящего сигнал из одноразрядного 2^n - ричного в n-разрядный двоичный.

Обычно при программировании генерируют таблицы замены.

Таблица 3

Таблица замены для приведённого 3х-разрядного S-блока

Номер комбинации	**Вход**	**Выход**
0	000	010
1	001	110
2	010	000
3	011	001
4	100	011
5	101	100
6	110	111
7	111	101

P-Блок(P-box)

Блок перестановок изменяет положение цифр и является линейным устройством.

Циклический сдвиг

Циклический сдвиг является частным случаем P-блока. В простейшем случае (сдвиг на 1 бит), крайний бит отщепляется и перемещается на другой конец регистра или шины. В зависимости от того какой бит берётся, правый или левый, сдвиг называется вправо или влево. Сдвиги на большее число бит можно рассматривать, как многократное применение сдвига на 1.

Таблица 4

Циклический сдвиг на m бит для n-разрядного входа (m < n)

Направление сдвига	**порядок следования битов до сдвига**	**порядок следования битов после сдвига**
влево	$b_0, b_1, b_2,\ldots b_{n-1}$	$b_m , b_{m+1} , b_{n-1} ,\ldots, b_0, b_1, b_2,\ldots b_{m-1}$
вправо	$b_0, b_1, b_2,\ldots b_{n-1}$	$b_{n-m} , b_{n-m+1} , b_{n-1} ,\ldots, b_0, b_1, b_2,\ldots b_{n-m-1}$

Сложение по модулю n

Обозначение операции – (A + B) mod n – это остаток от деления суммы A + B на n , где A,B – складываемые числа.

Можно показать, что сложение двух чисел по модулю представляется в двоичной системе счисления, как S-блок, у которого на вход подаётся число A, а в качестве системы коммутации S-блока используется циклический сдвиг влево на B разрядов.

Умножение по модулю n

Обозначение операции – (A * B) mod n – это остаток от деления произведения A * B на n , где A,B – перемножаемые числа.

Достоинства и недостатки

Достоинства:

- Простота аппаратной реализации на современной электронной базе
- Простота программной реализации в силу того, что все значительная часть функций поддерживаются на аппаратном уровне в современных комьютерах (например, сложение по модулю 2, сложение по модулю n, умножение по модулю n, и т. д.)
- Хорошая изученность алгоритмов на основе сетей Фейстеля.

Недостатки:

- За один раунд шифруется только половина входного блока. [10]

Теперь посмотрим работу сети Фейстеля на примере.

Пример 1

Для простоты будем работать с блоком, состоящим всего из двух байт. Делим его пополам и назовем левую часть L, а правую R, задаем функцию F.

XOR (обозначается $\oplus$) – инволютивная операция. Это всего лишь означает, что если поксорить одно число с другим дважды, то мы опять искомое получим. Т.е. A $\oplus$ B $\oplus$ B == A.

Отсюда следует, что можно выстраивать бесконечные цепочки A $\oplus$ B $\oplus$ C $\oplus$ D $\oplus$… и если мы перексорим всё в обратном порядке, то получим A.

Теперь про «черный ящик» или функцию F. Функция F по идее может быть любой, функцией, потому что когда мы будем в обратном порядке всё перексоривать (расшифровывать), то вторым аргументом у нас всегда будет тот же результат этой функции, что был в процессе шифрования. На практике же её создание сродни искусству и не дает 100% гарантии от новых методов криптоанализа.

Рассмотрим на примере трех раундов по шагам

Шифрование

0) У нас есть L и R какие-то числа. Пусть они будут 100 и 200, и F, какая-либо функция, зависящая от L и номера раунда n. Пусть, к примеру, F будет просто складывать их по модулю 256(чтоб не вылезало за байт). т.е. F(L, n) = (L+n) mod 256.

Раунд первый (n =1)

1) Берем R(200) и ксорим его с результатом функции F(L, n), т.е. 200 ⊕ ((100+1) % 256) получаем 173.

2) Ставим 173 на место L, а на место R предыдущее значение L (100), т.е. меняем местами R и результат ксора R с функцией F.

Раунд 2 (n = 2)

1) Теперь L = 173, R = 100. Ксорим 100 с ((173 + 2) % 256), получаем 203

2) Ставим 203 на место L, а 173 на место R.

Раунд 3 (n = 3)

1) L = 203, R = 173. Ксорим 173 с ((203 + 3) % 256), получаем 99

2) Поскольку раунд последний, то меняем только R (чтобы потом перестановку не делать)

После шифрования L = 203, R = 99.

Расшифровка

Идем в обратном порядке, номера раундов идут с 3 и до 1

Раунд 1 (n = 3)

1) L = 203, R = 99. Ксорим 99 с ((203 + 3) % 256) получаем 173. Знакомое число?

2) Ставим 173 на место L, 203 на место R

Раунд 2 (n = 2)

1) L = 173, R = 203. Отсюда 203 ⊕ ((173 + 2) % 256) = 100. Уже почти!

2) Меняем L = 100, R = 173

Раунд 3 (n = 1)

1) L = 100, R = 173. Считаем R(перестановка, как и в случае с шифрованием, не нужна) = 173 ⊕ ((100+1) % 256) = 200

L = 100, R = 200.

Пример 2

Возьмем слово AVADAKEDAVRA и разобьем его на два блока по шесть символов – AVADAK | EDAVRA. За функцию возьмем шифр сдвига на число позиций, определенных раундовым ключом. Пусть секретный ключ K = [1, 2]. В качестве раундовых ключей возьмём K[0] = 1, K[1] = 2. Для сложения по модулю 2 переведем текст в двоичный код согласно коду Бодо.

Код Бодо – цифровой 5-битный код. Код вводился прямо клавиатурой, состоящей из пяти клавиш, нажатие или ненажатие клавиши соответствовало передаче или непередаче одного бита в пятибитном коде.

Существует несколько разновидностей (стандартов) данного кода (CCITT-1, CCITT-2, МТК-2 и др.) В частности МТК-2 представляет собой модификацию международного стандарта CCITT-2 с добавление букв кириллицы.

Таблица 5
Стандарт кода Бодо МТК-2

Управляющие символы				
Двоичный код	Десятичный код	Назначение		
01000	8	Возврат каретки		
00010	2	Перевод строки		
11111	31	Буквы латинские		
11011	27	Цифры		
00100	4	Пробел		
00000	0	Буквы русские		
Буквы, цифры и остальные символы				
Двоичный код	Десятичный код	Латинская буква	Русская буква	Цифры и прочие символы
00011	3	A	А	-
11001	25	B	Б	?
01110	14	C	Ц	:
01001	9	D	Д	Кто там?
00001	1	E	Е	3
01101	13	F	Ф	Э
11010	26	G	Г	Ш
10100	20	H	Х	Щ
00110	6	I	И	8
01011	11	J	Й	Ю
01111	15	K	К	(
10010	18	L	Л	)
11100	28	M	М	.
01100	12	N	Н	,
11000	24	O	О	9
10110	22	P	П	0
10111	23	Q	Я	1
01010	10	R	Р	4
00101	5	S	С	'
10000	16	T	Т	5
00111	7	U	У	7
11110	30	V	Ж	=
10011	19	W	В	2
11101	29	X	Ь	/
10101	21	Y	Ы	6
10001	17	Z	З	+

Вот что получилось:

A	V	A	D	A	K	E	D	A	B	R	A
00011	11110	00011	01001	00011	01111	00001	01001	00011	11110	01010	00011

Теперь прогоним через сеть Фейстеля из двух раундов первый блок:

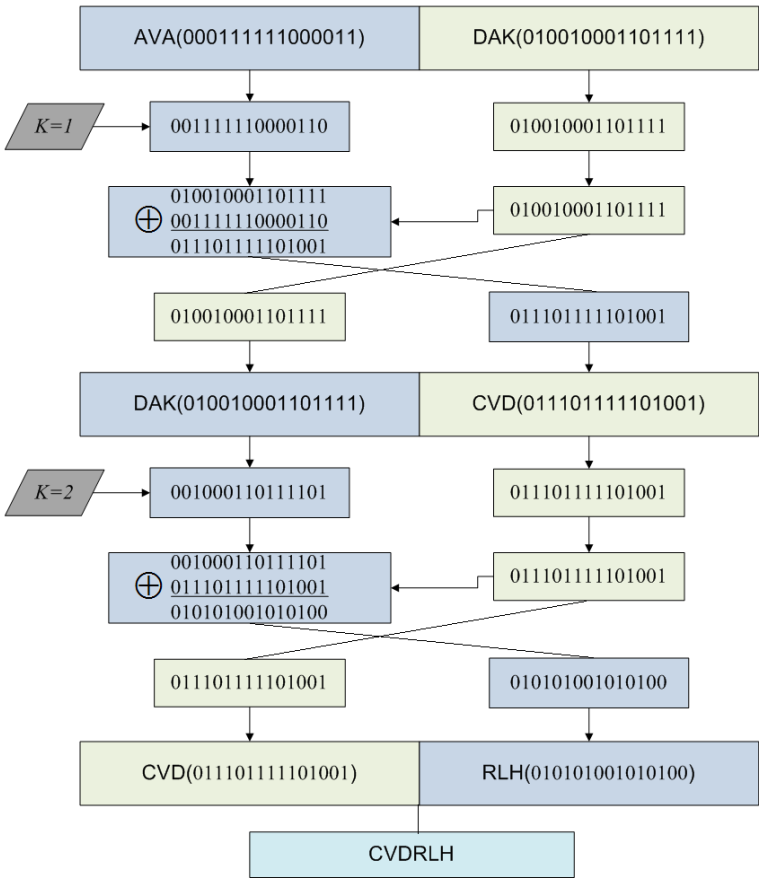

Рис.5. Прогон первого блока

Второй блок шифруется аналогично, и в результате получается шифрограмма MOSSTR.

Расшифрование осуществляется точно так же: шифротекст разбивается на блоки и затем подблоки, левый подблок поступает в функцию, складывается по модулю 2 с правым, и затем подблоки меняются местами. Отличие заключается в том, что раундовые ключи подаются в обратном порядке, то есть в нашем случае в первом раунде применим ключ K = 2, а затем во втором раунде K = 1.

Исследования сети Фейстеля показали, что при независимых раундовых ключах и криптостойкой псевдослучайной функции f трех раундов сети Фейстеля будет достаточно, чтобы шифротекст был

псевдослучайным. Это говорит о том, что шифры, основанные на сети Фейстеля, на данный момент достаточно криптостойки.

2.2.2. Шифр DES

Для шифрования DES принимает 64-битовый открытый текст и порождает 64-битовый зашифрованный текст и наоборот, получив 64 бита зашифрованного текста, он выдает 64 бита расшифрованного. В обоих случаях для шифрования и дешифрования применяется один и тот же 56-битовый ключ.

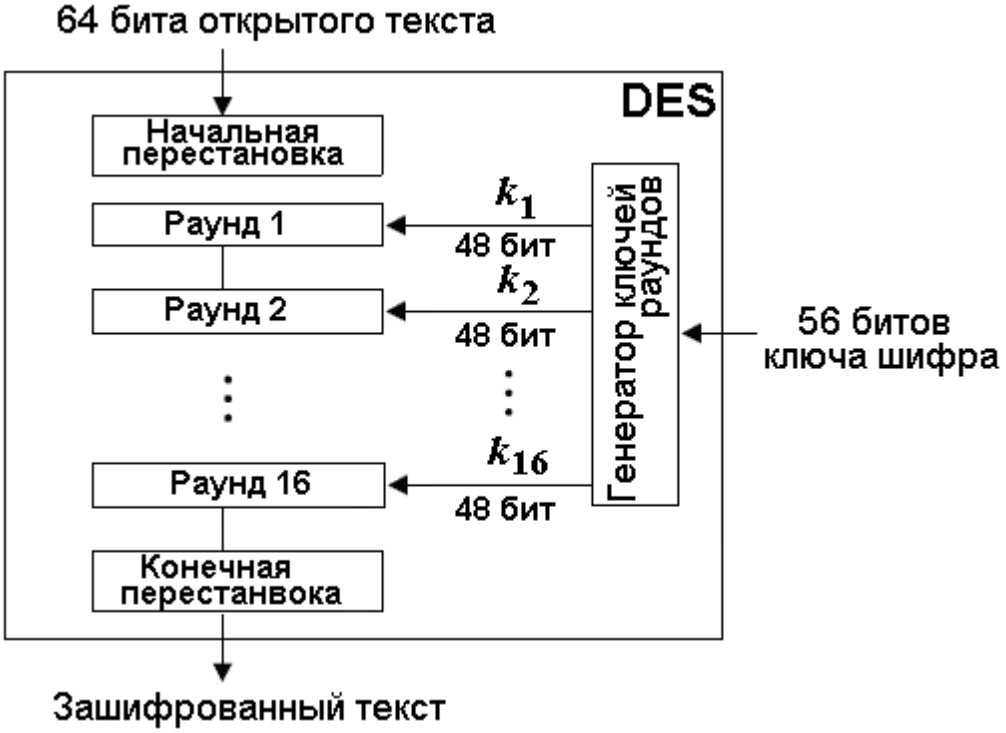

Рис. 6. Структура DES

Шифрование

Процесс шифрования состоит из двух перестановок, которые называют начальной и финальной (конечной) перестановками, и 16 раундов Фейстеля. Каждый раунд использует различные сгенерированные 48-битовые ключи.

Начальная IP и конечная IP^{-1}перестановки

На вход каждой из них поступает 64 бита, которые затем переставляются в соответствии с заданными таблицами. Эти перестановки взаимно обратны. Другими словами, 58-й бит на входе начальной перестановке переходит в 1-ую позицию на выходе из нее. А финальная перестановка 1-ый входной бит переведет в 58-ую позицию на выходе. бит 58 блока становится битом 1, бит 50 – битом 2 и т.д.

Таблица 6

Таблицы начальной и конечной перестановок

Начальная перестановка IP								конечная перестановка IP^{-1}							
58	50	42	34	26	18	10	2	40	8	48	16	56	24	64	32
60	52	44	36	28	20	12	4	39	7	47	15	55	23	63	31
62	54	46	38	30	22	14	6	38	6	46	14	54	22	62	30
64	56	48	40	32	24	16	8	37	5	45	13	53	21	61	29
57	49	41	33	25	17	9	1	36	4	44	12	52	20	60	28
59	51	43	35	27	19	11	3	35	3	43	11	51	19	59	27
61	53	45	37	29	21	13	5	34	2	42	10	50	18	58	26
63	55	47	39	31	23	15	7	33	1	41	9	49	17	57	25

Раунды DES

DES использует 16 раундов. Каждый раунд DES применяет шифр Фейстеля, как это показано на рисунке 7.

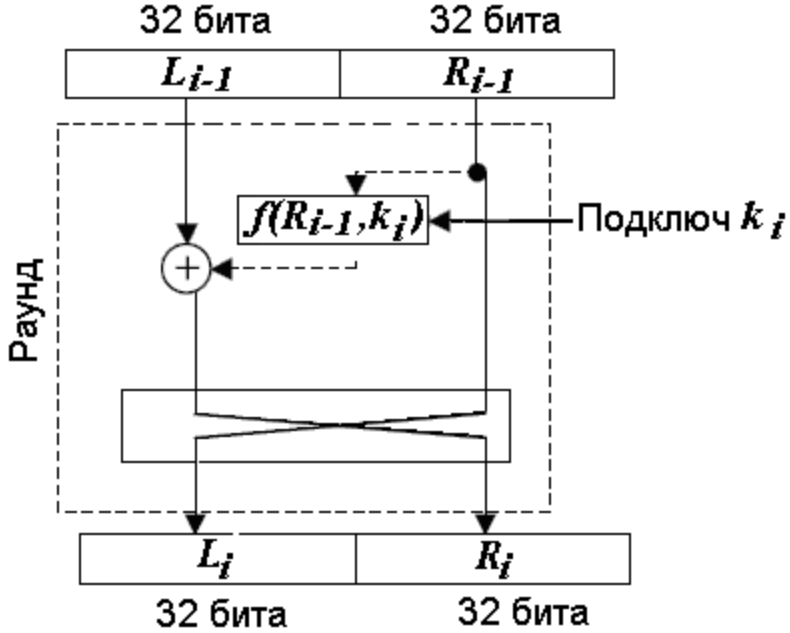

Рис.7. Схема раунда

Раунд принимает полублоки L_{i-1} и R_{i-1} от предыдущего раунда (или начального блока перестановки) и создает полублоки L_i и R_i для входа в следующий раунд (или конечный блок перестановки). Все необратимые элементы сосредоточены в функции f (R_{i-1},k_i) .

Функция DES

Функция DES с помощью 48-битового ключа зашифровывает 32 самых правых бит R_{i-1}, чтобы получить на выходе 32-битовый результат. Эта функция содержит, как это показано на 4 составляющие: операция XOR , P-бокс расширения, группу S -боксов и прямой бокс. (рис. 8)

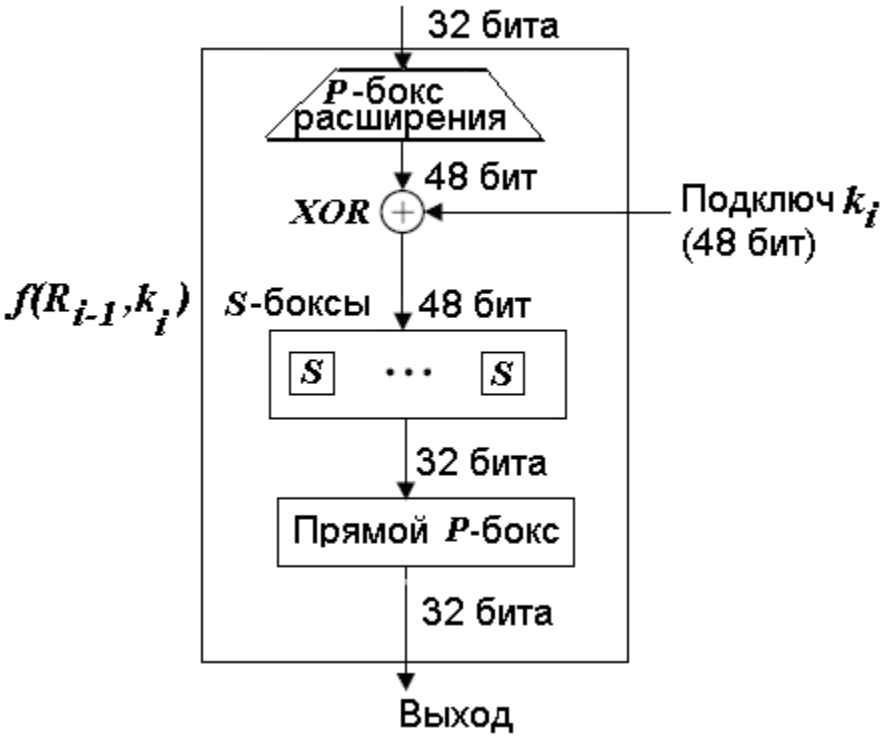

Рис. 8. Схема функции DES

P -бокс расширения служит для расширения 32-битового блока R_{i-1} до 48 битов, чтобы согласовать его размеров с размерами подключа раунда. Блок R_{i-1} делится на 8 секций по 4 бита. Каждая секция расширяется до 6 бит.

Хотя отношения между входом и выходом могут быть определены математически, P-бокс задают таблицей, где (число выходов 48, диапазон значений – от 1 до 32). (табл. 7).

Таблица 7

P -бокс расширения в i -м раунде

P-бокс расширения в *i*-м раунде					
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

После расширения DES использует операцию XOR над расширенной частью правого полублока R_{i-1} и ключом раунда k_i .

После суммирования с битами ключа блок из 48 битов делится на 8 последовательных 6-битових векторов b_1,b_2 ,...,b_8 , каждый из которых заменяется на 4-битовий вектор $b`_j$ с помощью S -боксов.

Таблица 8

Таблица S –бокс

S-боксы																	
		l – номер столбца															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
номер строки m	0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_1																	
	0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_2																	
	0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_3																	
	0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_4																	
	0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_5																	
	0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_6																	
	0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_7																	
	0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11
S_8																	

I вектор b_1 попадает в бокс S_1, II вектор b_2 – в бокс S_2 . S -бокс – это таблица из 4 строк и 16-ти столбцов. Чтобы в S -боксе найти шифрообозначения вектора, надо:

- из 1-го и последнего битов вектора образовать двоичное число,
- перевести его в десятичную систему. Это будет номер строки m (m = 0,1,2,3);
- из 2-го, 3-го, 4-го и 5-го бита вектора образовать двоичное число, перевести его в десятичную систему. Это будет номер столбца l (l =0,...,15);
- Число, стоящее в S -боксе на пересечении m-ой строки и l –го столбца, будет шифрообозначение $b`_j$ вектора b_j ;
- шифрообозначение $b`_j$ перевести в двоичную систему.

Таким образом, после S-боксов мы получаем 8 4-битовых векторов $b_1, b_2, ..., b_8$, которые опять объединяют в 32-битовый блок.

Далее биты блока перетасовываются в прямом P-боксе на основе заданной таблицы 9 (правила пользования таблицей перестановки: например, 7-ой бит входа станет 2-ым битом выхода).[10]

Таблица 9

P -бокс прямой в i -м раунде

P-бокс прямой в *i*-м раунде							
16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

Генерация ключей

DES создает 16 раундовых ключей k_i по 48 битов из ключа k шифра на 56 битов. Однако, чтобы задать ключ шифра надо среди 56 битов ключа дополнительно вписать 8 битов в позиции 8,16,...,64 для проверки четности таким образом, чтобы каждый байт содержал нечетное число единиц. С помощью этой операции выявляют ошибки при обмене и хранении ключей.

Ключевое расписание состоит из этапов:

1). Перестановка сжатия для удаления битов проверки (Функция *PC1*) – из 64-битового ключа удаляют биты 8,16,24, 32,…,64 и переставляет остальные биты согласно таблице (в ходе перестановки сохраняется нумерация битов расширенного ключа).

Таблица 10

Удаление проверочных битов ключа+перестановка

Удаление проверочных битов ключа+перестановка														
57	49	41	33	25	17	9	1	58	50	42	34	26	18	C_0
10	2	59	51	43	35	27	19	11	3	60	52	44	36	
63	55	47	39	31	23	15	7	62	54	46	38	30	22	D_0
14	6	61	53	45	37	29	21	13	5	28	20	12	4	

2) После перестановки 56 битов ключа делятся на два блока C_0 и D_0 по 28 бит каждый. Дале для генерации раундовых ключей из блоков C_0 и D_0 с помощью операции циклического сдвига влево на 1-2 бита строятся блоки C_i и D_i , i=1,2,...,16 . В раундах 1,2,9 и 16 смещение – на 1 бит, в других раундах – на 2 бита. После определения блоков C_i и D_i биты этих блоков объединяются в один ключ на 56 битов.

Таблица 11

Левый циклический сдвиг

Левый циклический сдвиг																
Раунд	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Величина сдвига	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

3). Перестановка сжатия (P-бокс, таблица) (*PC2)* изменяет 56 битов на 48 битов, которые образуют раундовый ключ.

Таблица 12

Перестановка сжатия ключа

Перестановка сжатия ключа							
14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

При расшифровании – раундовые ключи те же, что и при шифровании, но теперь они используются в обратном порядке. [3]

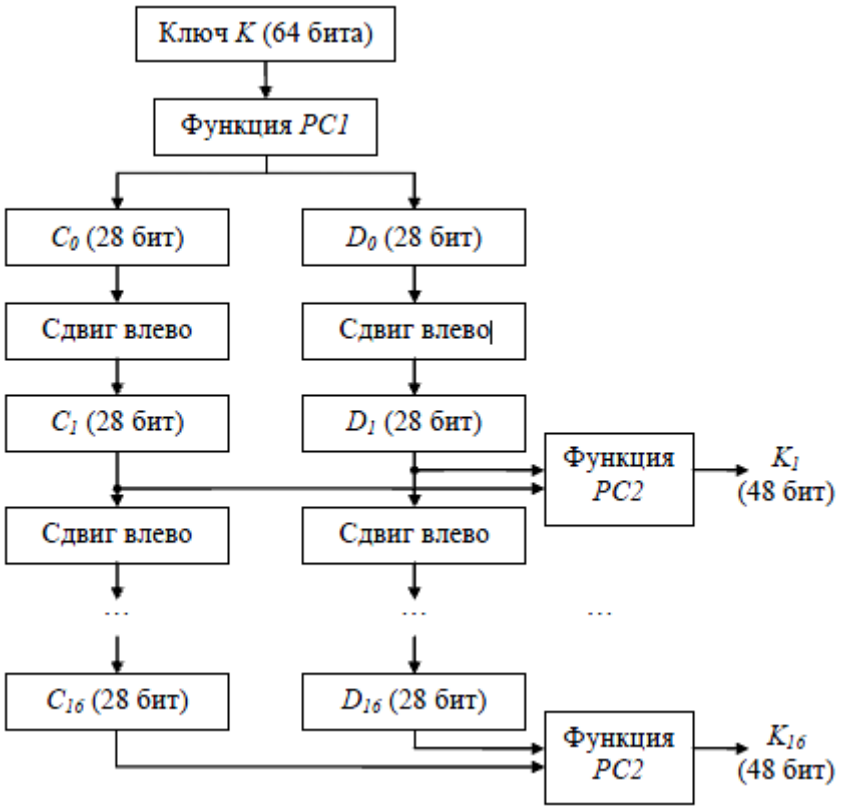

Рис. 9. Вычисление раундовых ключей

Пример.
Шифрование
Открытый текст: 123456ABCD 132536
Ключ: AABB09182736CCDD
После первоначальной перестановки: 14A7D67818CA18D
После разбиения: L_0= 14A7D678; R_0 = 18CA18D

Таблица 13

Пример шифрования DES

Раунд	Левый полублок	Правый полублок	Подключ раунда
Раунд 1	18CA18AD	5A78E394	194CD072DE8C
Раунд 2	5A78E394	4A1210F6	4568581ABCCE
Раунд 3	4A1210F6	B8089591	06EDA4ACF5B5
Раунд 4	B8089591	236779C2	DA2D032B6EE3
Раунд 5	236779C2	A15A4B87	69A629FEC913
Раунд 6	A15A4B87	2E8F9C65	C1948E87475E
Раунд 7	2E8F9C65	A9FC20A3	708AD2DDB3C0
Раунд 8	A9FC20A3	308BEE97	34F822F0C66D
Раунд 9	308BEE97	10AF9D37	84BB4473DCCC
Раунд 10	10AF9D37	6CA6CB20	02765708B5BF
Раунд 11	6CA6CB20	FF3C485F	6D5560AF7CA5
Раунд 12	FF3C485F	22A5963B	C2C1E96A4BF3
Раунд 13	22A5963B	387CCDAA	99C31397C91F
Раунд 14	387CCDAA	BD2DD2AB	251B8BC717D0
Раунд 15	BD2DD2AB	CF26B472	3330C5D9A36D
Раунд 16	19BA9212	CF26B472	181C5D75C66D
После объединения: 19BA9212 CF26B472			

После объединения: 19BA9212 CF26B472

Зашифрованный текст: C0B7A8D05F3A829C

(после конечной перестановки)

Дешифрование

Ключ 1-го раунда расшифрования – такой же как ключ 16-го раунда зашифрования.

Зашифрованный текст: C0B7A8D05F3A829C

После первоначальной перестановки: 19BA9212 CF26B472

После разбиения: L_0= 19BA9212 R_0 = CF26B472.

Таблица 14

Пример шифрования DES

Раунд	Левая	Правая	Ключ раунда
Раунд 1	CF26B472	BD2DD2AB	181C5D75C66D
Раунд 2	BD2DD2AB	387CCDAA	3330C5D9A36D
.....	.....	.....	.....
Раунд 15	5A78E394	18CA182AD	4568581ABCCE
Раунд 16	19BA9212	18CA18AD	194CD072DE8C

После объединения: 14A7D67818CA18D

Исходный текст: 123456ABCD 132536 (после конечной перестановки)

DES стойко выдержал 20 лет массового всемирного криптоанализа – десятилетия криптоанализа не привели к обнаружению серьезных уязвимостей в алгоритме.

Режимы работы DES

Для того, чтобы иметь возможность использовать шифр DES для решения различных криптографических задач, определены 4 режима его работы:

- электронная кодовая книга (англ. «Electronic Code Book» – ECB);
- сцепление блоков шифра (англ. «Cipher Block Chaining» – CBC);
- обратная связь по шифртексту (англ. «Cipher FeedBack» – CFB);
- обратная связь по выходу (англ. «Output FeedBack» – OFB).

При использовании режима ECB, защищаемое сообщение разбивают на 64-битные блоки M_i. Каждый такой блок шифруют независимо от других, с использованием одного и того же ключа шифрования (рис. 10). При расшифровывании криптограммы C_i также преобразуют независимо.

Достоинством данного режима является простота его реализации. Главный недостаток режима ECB заключается в том, что если в исходном сообщении есть повторяющиеся блоки, то и значения соответствующих блоков криптограммы будет совпадать. А это даст криптоаналитику противника дополнительную информацию о содержании сообщения. Поэтому режим ECB рекомендуют использовать для защиты небольших объемов данных (например, криптографических ключей), где вероятность появления совпадающих блоков сообщения невелика.

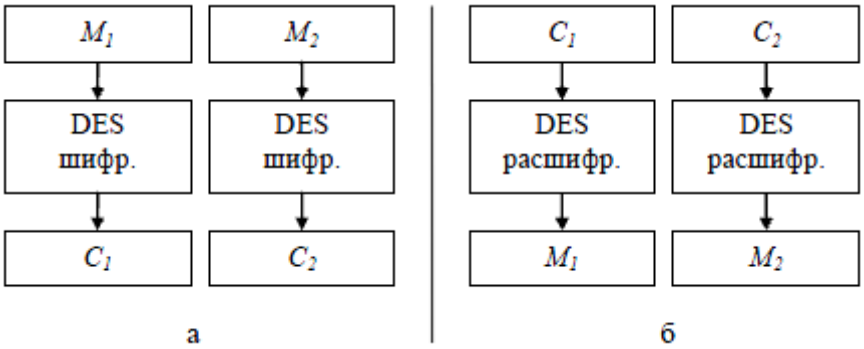

Рис. 10. Шифрование (а) и расшифровывание (б) в режиме ECB

Указанного выше недостатка лишен режим CBC. Исходное сообщение, как и в предыдущем случае, разбивается на блоки M_i по 64 бита. Первый блок складывается побитно по модулю 2 с 64-битным блоком, называемым инициализирующим вектором – IV, который известен обеим сторонам взаимодействия, периодически ими меняется и держится в секрете от других. Блок исходного сообщения M_2 модифицируется с использованием блока криптограммы C_1 и т. д. Аналогичные действия

производятся при расшифровывании. Схема преобразования представлена на рис. 11.

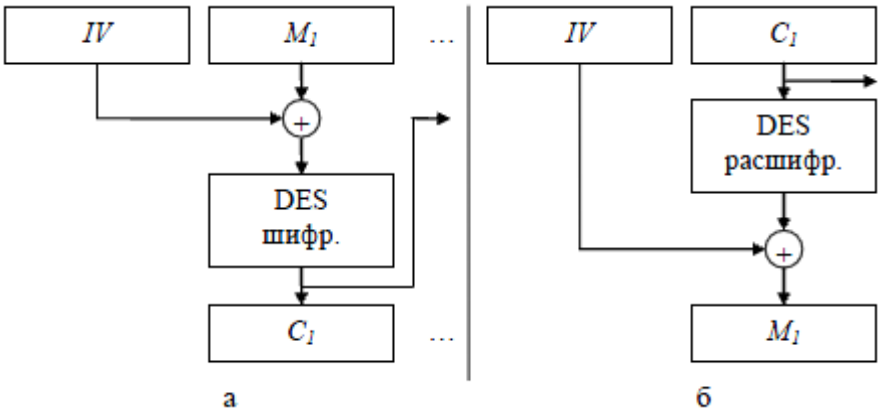

Рис. 11. Шифрование (а) и расшифровывание (б) в режиме CBC

Режим CBC используется для шифрования больших сообщений. Последний блок криптограммы зависит от инициализирующего вектора, каждого бита открытого текста и значения секретного ключа. Поэтому его можно использовать для контроля целостности и аутентификации сообщений, задавая ему фиксированное значение и проверяя его после расшифровки.

Режим CFB используется в тех случаях, когда длина преобразуемого блока отличается от 64 бит. Пусть необходимо зашифровать сообщение, считываемое последовательно блоками по r бит, где $1 \leq r \leq 64$. Для построения преобразования используется сдвиговый регистр I, куда на 1-м шаге преобразования помещается инициализирующий вектор IV. Схема преобразования представлена на рис. 12.

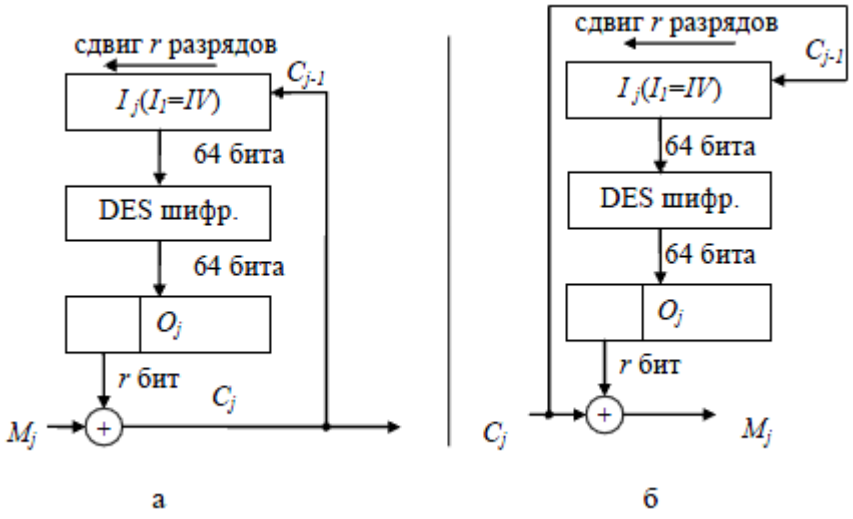

Рис. 12. Шифрование (а) и расшифровывание (б) в режиме CFB

Преобразуемое сообщение M разбито на блоки по r бит, обозначаемые на рисунке M_j. Блок криптограммы C_j будет равен M_j сложенному побитно по модулю 2 с r старшими битами зашифрованного на j-м шаге блока. Шифруемое значение I_j получается путем сдвига предыдущего блока I_{j-1} влево на r позиций и записи блока криптограммы C_{j-1} в младшие позиции.

При расшифровывании сдвиговый регистр также инициализируется значением IV. Для того, чтобы получить ту же последовательность вспомогательных значений Oj, что и при шифровании, здесь также используется DES шифрование (а не расшифровывание, как например, при обратном преобразовании в режиме CBC).

Режим OFB также позволяет шифровать блоки, меньшие по длине, чем 64 бита. Его схема представлена на рис. 13. Аналогично режиму CFB сдвиговый регистр сначала содержит значение инициализирующего вектора. Есть два варианта модификации его значения.

На рисунке представлен вариант с полной заменой на j-м шаге содержимого сдвигового регистра вспомогательным значением O_{j-1}. Второй вариант построения схемы предполагает, как и в случае CFB, сдвиг I_{j-1} влево на r разрядов и запись в младшие разряды сдвигового регистра старших r разрядов O_{j-1}.

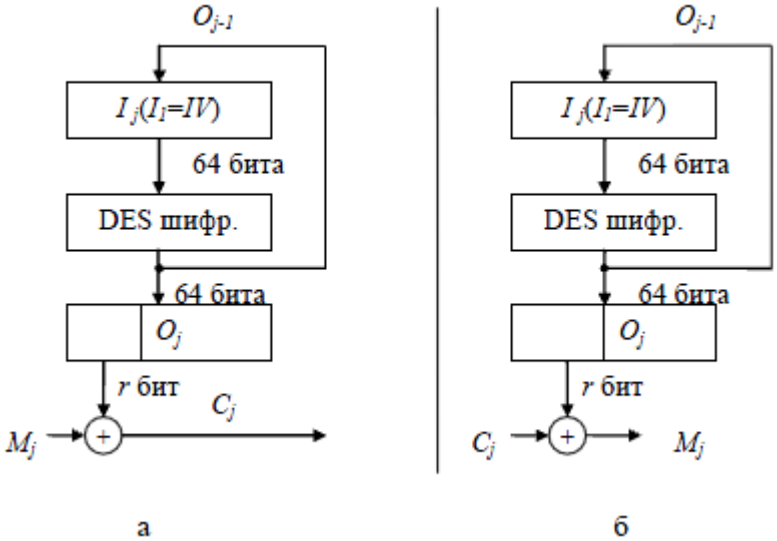

Рис. 13. Шифрование (а) и расшифровывание (б) в режиме OFB

При использовании режима OFB важно для каждого сеанса шифрования данных использовать новое начальное состояние сдвигового регистра (его можно передавать, в том числе, и открытым текстом). Связано это с тем, в режимах OFB и CFB генерируется псевдо-случайная последовательность чисел, которая накладывается на блоки открытого текста. В случае использования режима OFB, если дважды используется

один и тот же инициализирующий вектор и ключ шифрования, то и генерируемые последовательности будут совпадать.

Некоторое время шифр DES считался достаточно безопасным. Но по мере развития вычислительной техники, короткий 56-битный ключ привел к тому, что атака путем полного перебора ключевого множества стала относительно легко реализуемой. Чтобы увеличить стойкость алгоритма и в то же время сохранить существующие наработки (в виде программных и аппаратных реализаций алгоритма), было использовано многократное шифрование.

Более надежной оказалась схема, включающая шифрование, расшифровывание и повторное шифрование на различных ключах.

Данный шифр получил название Triple DES. Варианты схемы его построения приведены на рис. 14.

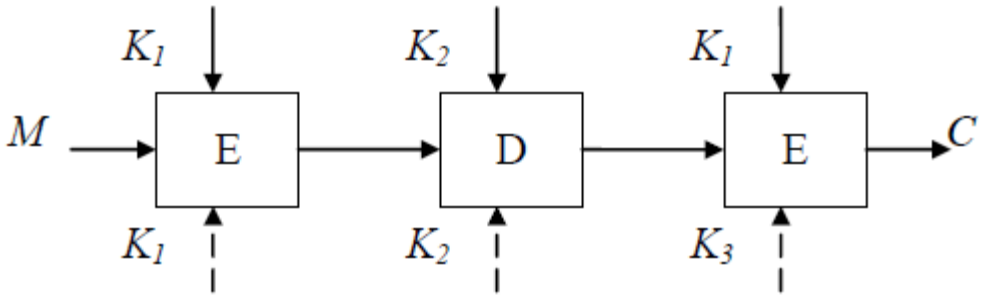

Рис. 14. Шифр Triple DES

В первом случае, выбор ключей производится так, как показано в верхней части рисунка: сначала производится шифрование на ключе K_1, далее – расшифровывание на ключе K2, после – шифрование на ключе K_1. Суммарная длина ключа – 112 бит. Более надежной считается схема с использованием в третьем преобразовании еще одного ключа – K_3 (изображена в нижней части рисунка). Тогда суммарный ключ будет длиной 168 бит. [9]

2.2.3. Шифр ГОСТ 28147-89

Данный алгоритм симметричного шифрования был разработан в СССР и в качестве стандарта утвержден в 1989 году. Он считается достаточно стойким и широко используется в России теми предприятиями и организациями, которым, в силу особенностей сферы их деятельности, необходимо применять сертифицированные средства криптографической защиты данных (это государственные и военные структуры, организации банковской сферы и т. д.).

Этот шифр преобразует сообщение 64-битными блоками, преобразование осуществляется в соответствии со схемой Фейстеля в 32 раунда, размер ключа – 256 бит. Алгоритм предусматривает 4 режима работы:

- шифрование данных в режиме простой замены (аналог режима ECB для шифра DES);
- шифрование данных в режиме гаммирования (аналог режима OFB для шифра DES);
- шифрование данных в режиме гаммирования с обратной связью;
- выработка имитовставки.

Ниже будет рассмотрен режим простой замены, являющийся основой для построения других режимов. Схема раунда шифрования приведена на рис. 15.

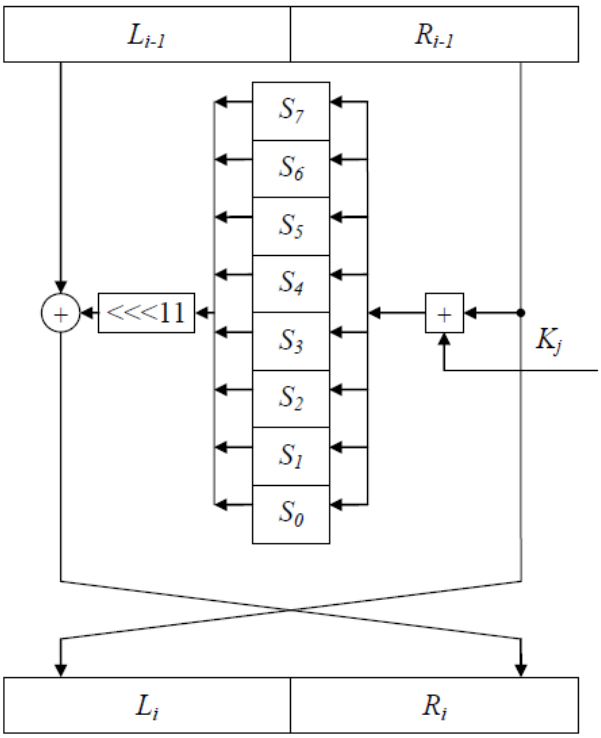

Рис. 15. Схема раунда алгоритма ГОСТ 28147-89

Преобразование производится в следующем порядке. Правый полублок R_{i-1} складывается по модулю 232 (сложение 32-разрядных двоичных значений без переноса старшего разряда) с раундовым ключом K_j. Далее выполняется подстановка, задаваемая таблицами S_0, …, S_7, и полученное значение преобразуется с помощью циклического сдвига влево на 11 позиций. После этого выполняется побитное сложение по модулю 2 с левым полублоком L_{i-1} и перестановка полублоков.

Расписание использования раундовых ключей формируется следующим образом: 256-битный секретный ключ K представляется в виде сцепления 8-ми 32-битных подключей $K=K_7|K_6|K_5|K_4|K_3|K_2|K_1|K_0$.

На первом раунде берется 0-й подключ, на втором – 1-й и т. д., на 9-м раунде опять используется 0-й подключ, на 24-м – 7-й, а вот на 25-м раунде преобразования вновь используется 7-й и далее ключи используются в обратном порядке. Иными словами, номер раундового ключа j зависит от номера раунда i следующим образом:

j = (i -1)mod8 для 1<= i<= 24,

j = (32 =i)mod 8 для 25 <= i <= 32 .

Подстановка проводится после разбиения входного значения на 4-битные подблоки. После того как правый полублок R был сложен с раундовым подключем, он разбивается на 8 частей $R=R_7|R_6|R_5|R_4|R_3|R_2|R_1|R_0$. Таблица подстановок S_i представляет собой вектор с 16-ю 4-битовыми элементами. Из нее берется элемент, номер которого совпадает со значением R_i, пришедшим на вход подстановки.

По сравнению с шифром DES у ГОСТ 28147-89 есть следующие достоинства:

- существенно более длинный ключ (256 бит против 56 у шифра DES), атака на который путем полного перебора ключевого множества на данный момент представляется невыполнимой;
- простое расписание использования ключа, что упрощает реализацию алгоритма и повышает скорость вычислений. [9]

2.2.4. Шифр Blowfish

Шифр Blowfish ориентирован на программную реализацию на 32-разрядных микропроцессорах. Его отличают высокая скорость и криптостойкость.

Также в качестве отличительной особенности можно назвать возможность использовать ключ переменной длины. Шифр блочный, размер входного блока равен 64 битам. Преобразование блока выполняется в 16 раундов (есть версия с 111-ю раундами). Ключ переменной длины, максимально 448 бит.

До начала шифрования или расшифровывания данных производится расширение ключа. В результате, на базе секретного ключа получают расширенный, который представляет собой массив из 18 раундовых ключей K_1, …, K_{18} (размерность K_i □□32 бита) и матрицу подстановок Q с 4-мя строками, 256-ю столбцами и 32-битными элементами (рис. 16).

$$Q = \begin{vmatrix} Q_0^{(1)} & \cdots & Q_{255}^{(1)} \\ Q_0^{(2)} & \cdots & Q_{255}^{(2)} \\ Q_0^{(3)} & \cdots & Q_{255}^{(3)} \\ Q_0^{(4)} & \cdots & Q_{255}^{(4)} \end{vmatrix}$$

Рис. 16. Матрица подстановок

Данная матрица используется для задания нелинейной функции шифрующего преобразования F(X), где X – 32-битный аргумент. X представляется в виде сцепления 4-х 8-битных слов $X = X_3 \mid X_2 \mid X_1 \mid X_0$, а сама функция задается формулой (+ здесь обозначает сложение по модулю 2^{32}, $\oplus$- сложение по модулю 2):

$$F(X) = \left(\left(Q_{X_3}^{(1)} + Q_{X_2}^{(2)}\right) \oplus Q_{X_1}^{(3)}\right) + Q_{X_0}^{(4)}.$$

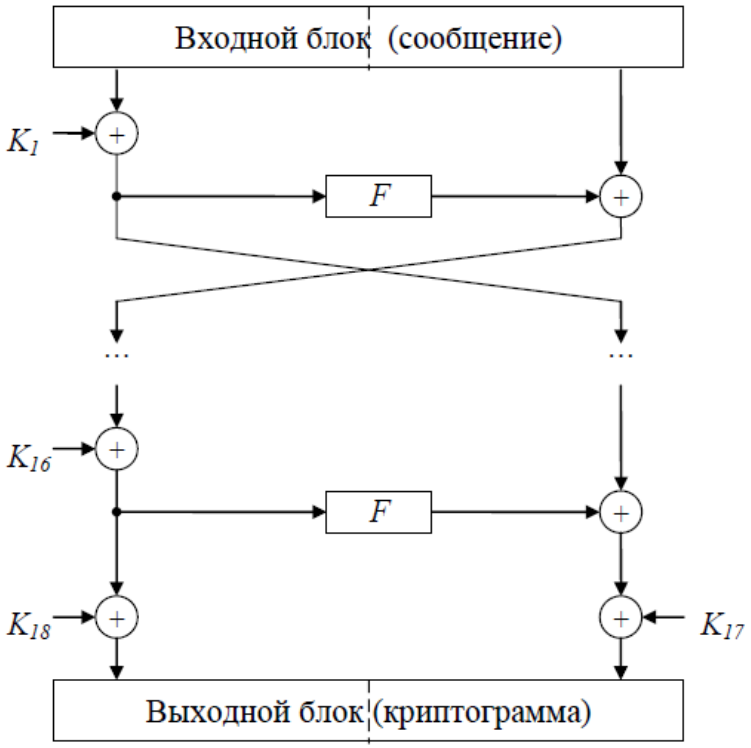

Рис. 17. Шифр Blowfish

Схема шифрующего преобразования приведена на рис. 17.

Расширение секретного ключа *KS* производится следующим образом.

1) Начальный массив раундовых ключей K_i и элементов *Q* инициализируется фиксированными значениями. Например, в шестнадцатеричном представлении K_1=243F6A88 и т. д.

2) K_1 суммируется по модулю 2 с первыми 32-мя битами секретного ключа K_S, K_2 – со следующими 32-мя битами и т. д. Когда ключ K_S заканчивается, его начинают использовать сначала. Суммируются только K_i (без Q).

3) 64-битный блок нулей 0=(0…0) зашифровывают с помощью Blowfish на ключах, полученных на шагах 1) и 2): C_0=Blowfish(0).

4) 4) Раундовые подключи K1 и K2 заменяют полученным на шаге 3) значением C_0.

5) 5) C_0 шифруют на модифицированных ключах C_1= Blowfish(C0).

6) Раундовые ключи K3 и K4 заменяют значением C_1.

7) Процесс продолжается, пока не будут получены сначала все пары раундовых ключей (9 пар), а затем все пары элементов матрицы Q (512 пар).

Таким образом, расширение ключа требует шифрования 521 блока данных. Эта процедура дополнительно осложняет атаку путем перебора ключевого множества, т. к. нарушитель будет вынужден проводить процедуру расширения для каждого возможного ключа. [9]

2.2.5. Алгоритм Rijndael

Алгоритм Rijndael достаточно сложен для описания, поэтому рассмотрим только основные аспекты построения и особенности использования шифра.

Шифр *Rijndael/AES* (то есть рекомендуемый стандартом) характеризуется размером блока 128 бит, длиной ключа 128, 192 или 256 бит и количеством раундов 10, 12 или 14 в зависимости от длины ключа. В принципе, структуру *Rijndael* можно приспособить к любым размерам блока и ключа, кратным 32, а также изменить число раундов.

В отличие от шифров, предлагаемых *DES* и ГОСТ 28147-89, в основе *Rijndael* не лежит сеть Фейстеля. Основу *Rijndael* составляют так называемые *линейно-подстановочные преобразования*. Блок данных, обрабатываемый с использованием *Rijndael*, делится на массивы байтов, и каждая операция шифрования является байт-ориентированной. Каждый раунд состоит из трех различных обратимых преобразований, называемых слоями. Эти слои следующие.

1. Нелинейный слой. На этом слое выполняется замена байтов. Слой реализован с помощью S-блоков, имеющих оптимальную нелинейность, и предотвращает возможность использования дифференциального, линейного и других современных методов криптоанализа.

2. Линейный перемешивающий слой гарантирует высокую степень взаимопроникновения символов блока для маскировки статистических связей. На этом слое в прямоугольном массиве байтов выполняется сдвиг строк массива и перестановка столбцов.

3. Слой сложения по модулю 2 с подключом выполняет непосредственно шифрование.

Шифр начинается и заканчивается сложением с ключом. Это позволяет закрыть вход первого раунда при атаке по известному тексту и сделать криптографически значимым результат последнего раунда.

В алгоритме широко используются табличные вычисления, причем все необходимые таблицы задаются константно, т.е. не зависят ни от ключа, ни от данных.

Необходимо отметить, что в отличие от шифров, построенных по сети Фейштеля, в *Rijndael* функции шифрования и расшифрования различны.

Алгоритм *Rijndael* хорошо выполняется как в программной, так и в аппаратной реализации. *Rijndael* имеет небольшие требования к памяти, что делает его пригодным для систем с ограниченными ресурсами. Надежность шифрования алгоритмом *Rijndael* очень высоко оценивается специалистами. [10]

2.2.6.Шифр AES

AES является упрощенной версией алгоритма Rijndael.

Оригинальный алгоритм Rijndael отличается тем, что поддерживает более широкий набор длин блоков.

AES представляет собой алгоритм шифрования 128-битных блоков данных ключами по 128, 192 и 256 бит.

Алгоритм AES представляет блок данных в виде двумерного байтового массива размером 4x4. Все операции производятся над отдельными байтами массива, а также над независимыми столбцами и строками.

Введем определения.

Байт – последовательность из 8 битов. В контексте данного алгоритма байт рассматривается как элемент поля Галуа. Операции над байтами производятся как над элементами поля Галуа $GF(2^8)$ (конечного поля), то есть байту $\{b_7b_6b_5b_4b_3b_2b_1b_0\}$ соответствует многочлен $\sum_{i=0}^{7} b_i x^i$ в поле $GF(2^8)$ [4].

Слово (word) – последовательность из 4 байтов.

Блок – последовательность из 16 байтов, над которой оперирует алгоритм. Блок служит входным и выходным данными алгоритма. Байты в блоке нумеруются с нуля.

Ключ – последовательность из 16, 24 или 32 байтов, используемая в качестве ключа шифрования. Байты в ключе нумеруются с нуля. Ключ, наряду с блоком, является входным данным алгоритма.

Форма (state) – двумерный массив байтов, состоящий из четырех строк. Байты в форме располагаются в порядке, изображенном на таблице 15. В алгоритме AES форма используется для представления блока.

Таблица 15
Порядок байтов в форме

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

Раунд – итерация цикла преобразований над формой. В зависимости от длины ключа раундов может быть от 10 до 14, как показано на таблице 15.

Ключ раунда (round key) – ключ, применяемый в раунде. Вычисляется для каждого раунда.

Таблица подстановок (S-box) – таблица, задающая биективное отображение байта в байт. Таблица подстановок представлена в таблице 16.

Обратная таблица подстановок – таблица, задающая отображение, обратное задаваемому таблицей подстановок. Обратная таблица подстановок представлена в таблице 17.

Введем обозначения Nb – количество слов (word) в блоке.

Nk – количество слов в ключе. Nk может принимать значения 4, 6, 8.

Nr – количество раундов. Параметр Nr =10, 12, 14

Для шифрования в алгоритме AES применяются следующие процедуры преобразования данных:

1. ExpandKey – Вычисление раундных ключей для всех раундов.
2. SubBytes – Подстановка байтов с помощью таблицы подстановок;
3. ShiftRows – Циклический сдвиг строк в форме на различные величины;
4. MixColumns – Смешивание данных внутри каждого столбца формы;
5. AddRoundKey – Сложение ключа раунда с формой.

Порядок выполнения процедур 2 и 3 можно поменять местами в силу определения этих операций.

Процедуры 4 и 5 тоже можно выполнять в разном порядке, но при этом изменяется количество их вызовов, поскольку MixColumns(AddRoundKey(A, B)) = AddRoundKey(MixColumns(A), MixColumns(B)). Шифрование производится по алгоритму, приведенному на рис. 18.

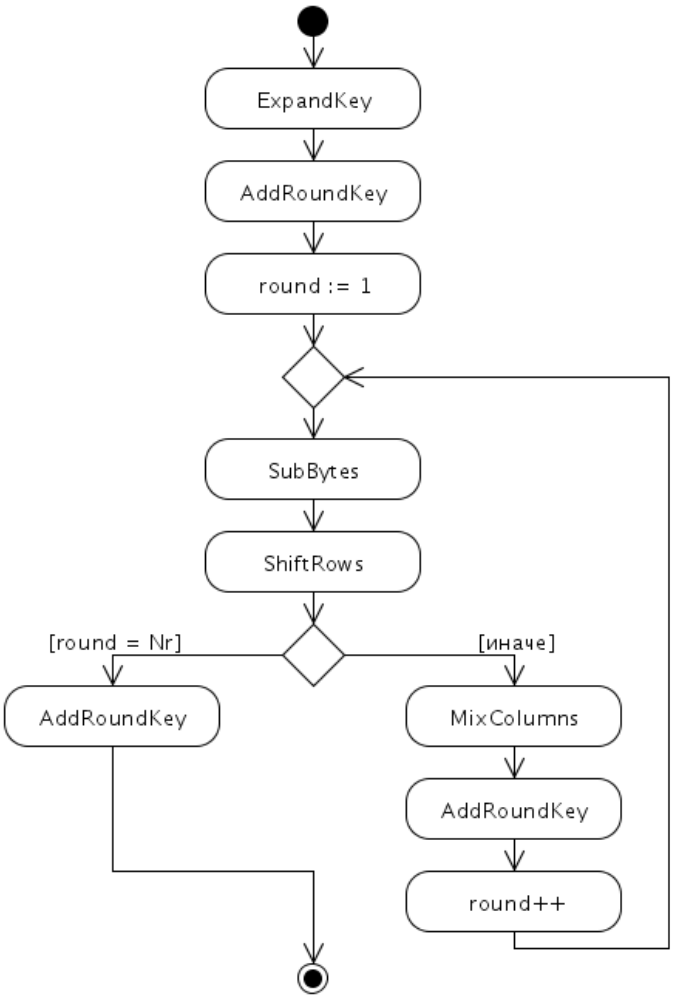

Рис. 18 Алгоритм шифрования AES

Преобразование SubBytes

Преобразование SubBytes заключается в замене каждого байта {xy} формы (где x и y обозначают шестнадцатиричные цифры) на другой в соответствии с таблицей 16. Например, байт {fe} заменится на {bb}.

Таблица 16
Таблица подстановок

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

Преобразование ShiftRows

Преобразование ShiftRows заключается в циклическом сдвиге влево строк формы. Преобразование схематично представлено на рис. 19. Первая строка остается неизменной. Во второй производится сдвиг на 1 байт, то есть первый байт переносится в конец. В третьей – сдвиг на 2 байта, в четвертой – на 3.

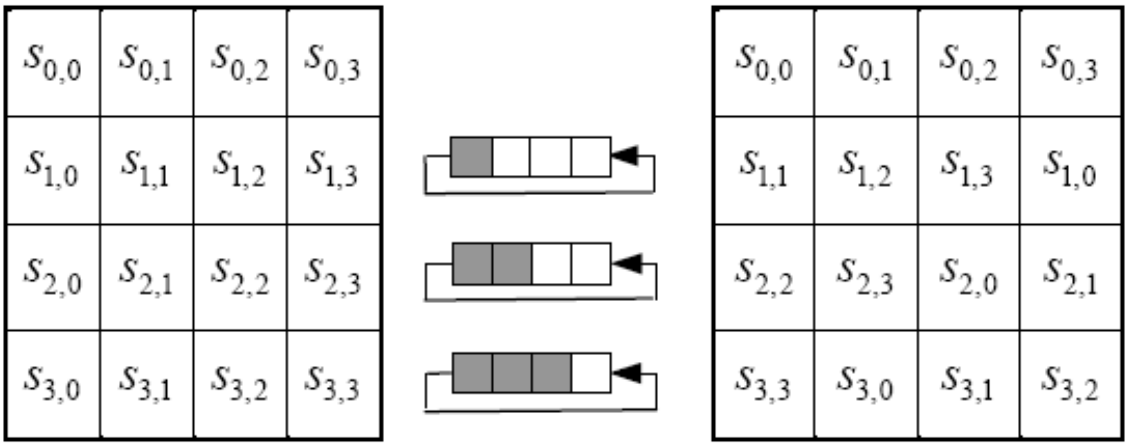

Рис. 19. Преобразование ShiftRows

Преобразование MixColumns

Преобразование MixColumns заключается в умножении квадратной матрицы 4-го порядка на каждый столбец формы из рис. 20.

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

Рис. 20. Умножение матрицы

Умножение производится в поле Галуа $GF(2^8)$.

Над каждым столбцом операция производится отдельно, как показано на рис. 21.

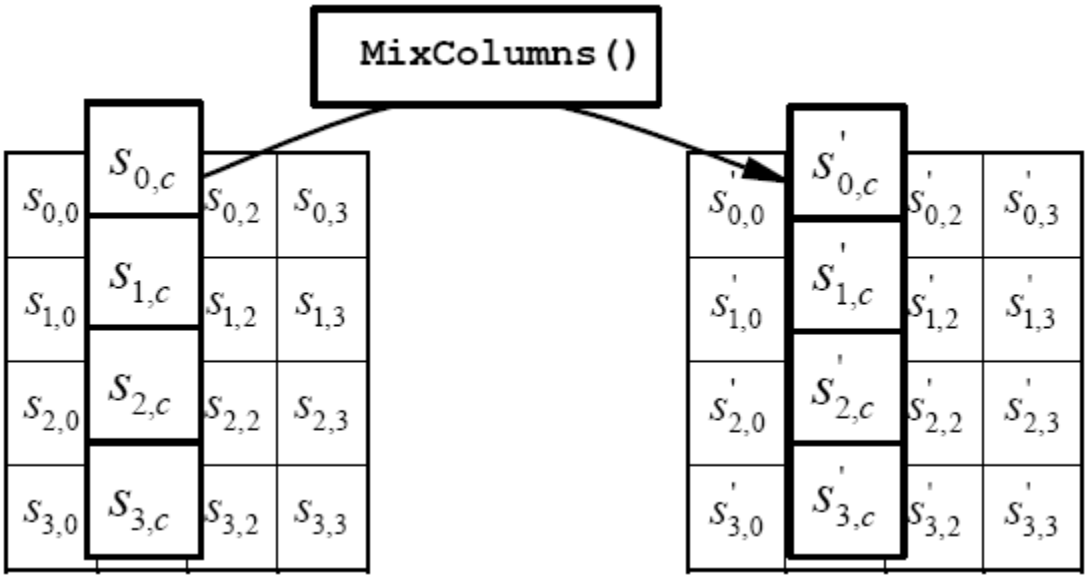

Рис. 21. Преобразование MixColumns

Преобразование AddRoundKey

В преобразовании AddRoundKey 32-битные слова раундного ключа прибавляются к столбцам формы с помощью побитовой операции XOR:

$[s`_{0,c}, s`_{1,c}, s`_{3,c}] = [s_{0,c}, s_{1,c}, s_{3,c}] \oplus [w_{round*Nb+c}]$.

Здесь w_i — это столбцы ключа.

Над каждым столбцом операция производится отдельно, как показано на рис. 22.

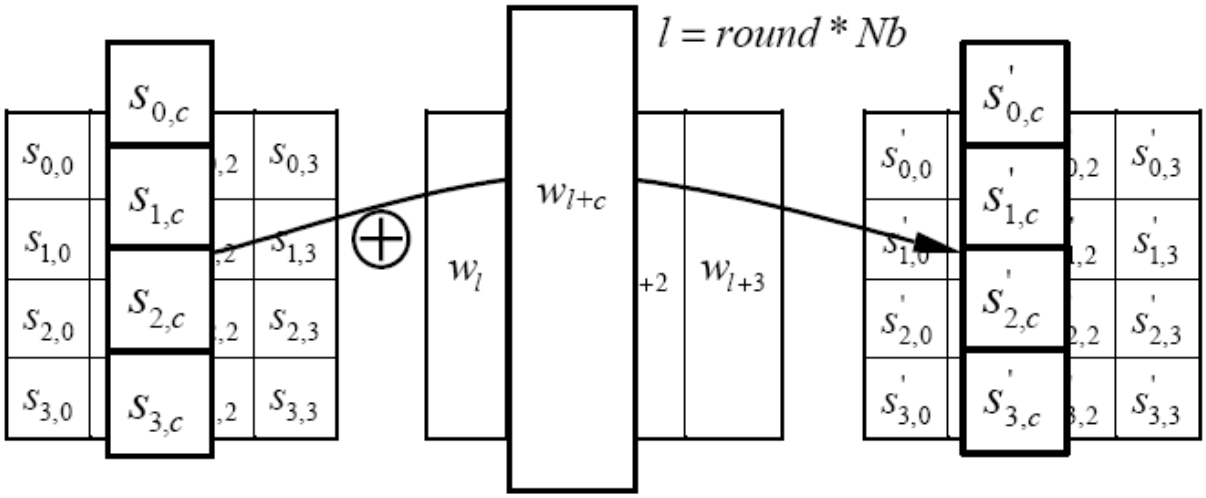

Рис. 22. Преобразование AddRoundKey

Процедура ExpandKey

В алгоритме AES генерируются раундные ключи на основе ключа шифрования с помощью процедуры ExpandKey. Процедура ExpandKey создает Nb * (Nr + 1) слов: алгоритму требуется начальный ключ размером Nb, плюс каждый из Nr раундов требует ключ из Nb слов. Ниже приведен псевдокод процедуры ExpandKey:

```
// Процедура вычисляет ключи раундов.
// key – ключ
// out – результат
// Nk – количество слов в ключе
ExpandKey(byte key[4*Nk], word out[Nb*(Nr+1)], int Nk)
begin
i = 0
while (i < Nk)
out[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
i = i + 1
end while
i = Nk
while (i < Nb * (Nr+1))
word temp = out[i-1]
if (i mod Nk = 0)
temp = SubWord(RotWord(temp)) xor Rcon(i/Nk)
else if ((Nk > 6) and (i mod Nk == 4))
temp = SubWord(temp)
end if
out[i] = out[i-Nk] xor temp
i = i + 1
end while
```

end

Здесь использованы следующие функции:

SubWord осуществляет замену каждого байта в слове в соответствии с таблицей подстановок, представленной на рис. 22.

RotWord осуществляет циклический сдвиг байтов в слове влево, как показано на рис. 22.

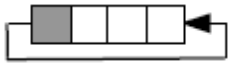

Рис. 22. Процедура RotWord

Rcon(i) формирует слово [02^{i-1}, 00, 00, 00].

Дешифрование

При дешифровании все преобразования производятся в обратном порядке. Используются следующие обратные преобразования вместо соответствующих шифрующих:

• InvSubBytes – Подстановка байтов с помощью обратной таблицы подстановок;

• InvShiftRows – Циклический сдвиг строк в форме на различные величины;

• InvMixColumns – Смешивание данных внутри каждого столбца формы;

Процедуры ExpandKey и AddRoundKey остаются неизменными. Ключи раунда используются в обратном порядке.

Алгоритм дешифрования представлен на рис. 23.

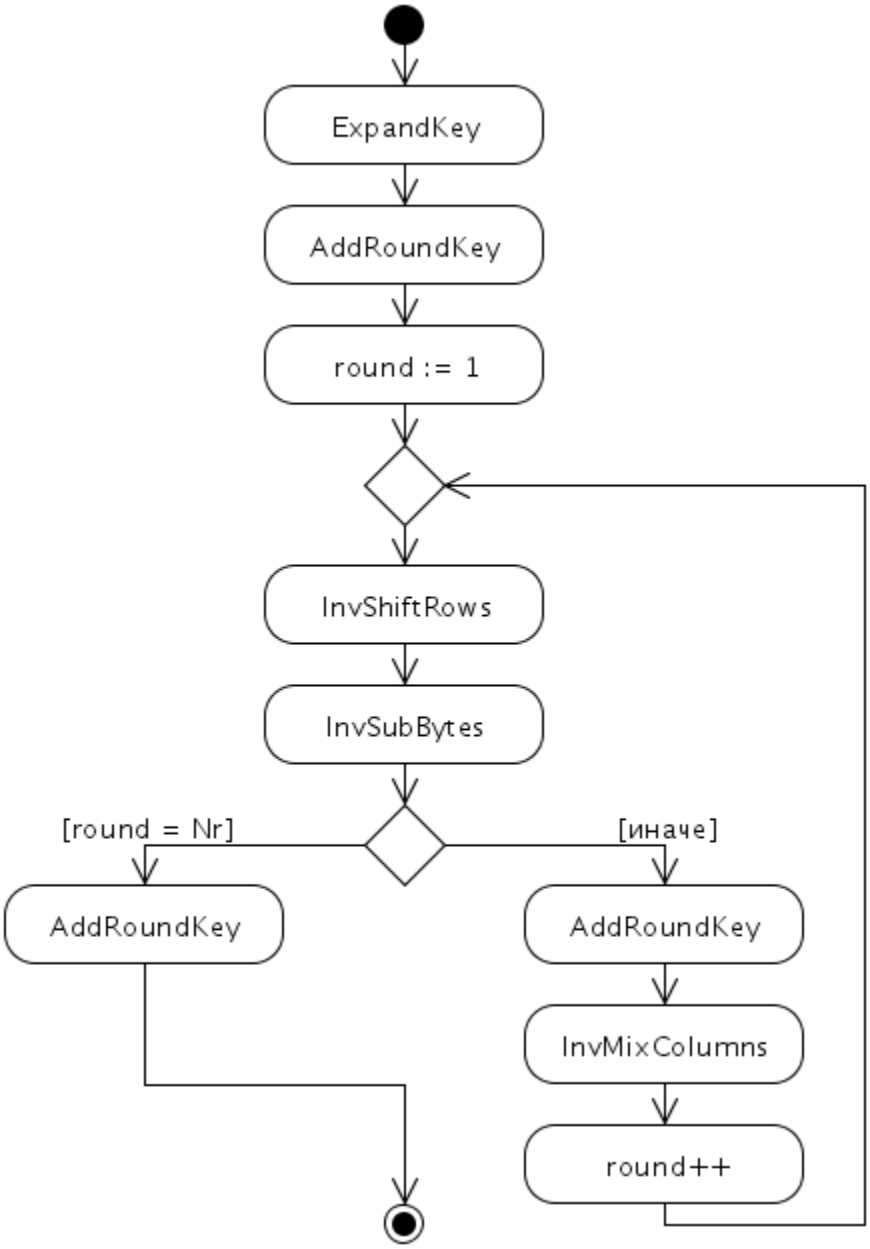

Рис. 23. Алгоритм дешифрования

Преобразование InvShiftRows

Это преобразование обратно преобразованию ShiftRows. Первая строка формы остается неизменной. Вторая строка циклически сдвигается вправо на 1 байт. Третья – на 2, четвертая – на 3. Схематично преобразование показано на рис. 24.

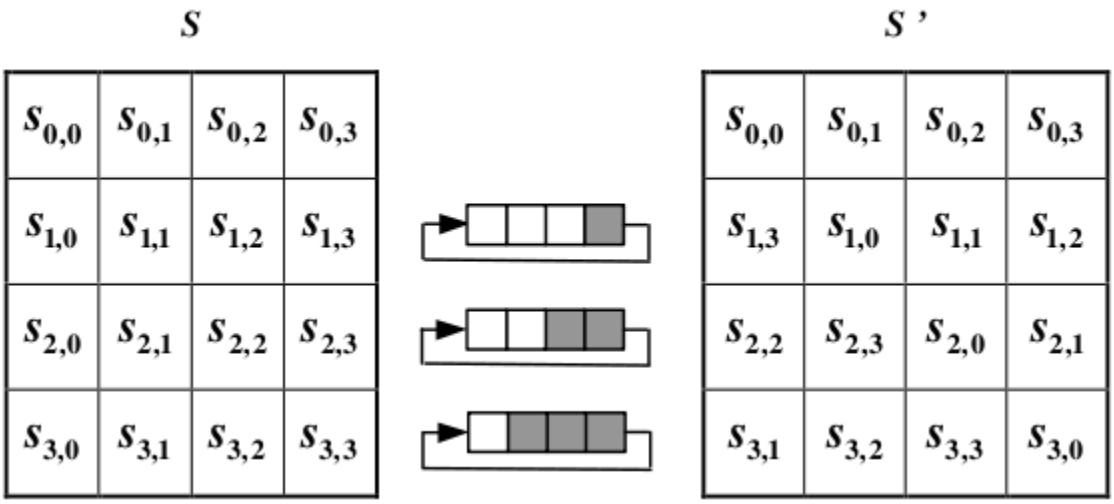

Рис. 24. Процедура InvShiftRows

Преобразование InvSubBytes

Это преобразование обратно преобразованию SubBytes. Подстановка байтов происходит аналогично с помощью обратной таблицы подстановок, представленной в таблице 17.

Талица 17

Обратная таблица подстановок

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

Преобразование InvMixColumns

Это преобразование обратно преобразованию MixColumns. InvMixColumns преобразует в форме каждый столбец отдельно. Преобразование происходит по следующие формуле:

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

Здесь умножение также производится в поле Галуа GF(2^8). [4]

2.2.7. УПРАВЛЕНИЕ КРИПТОГРАФИЧЕСКИМИ КЛЮЧАМИ ДЛЯ СИММЕТРИЧНЫХ ШИФРОВ

Под ключевой информацией понимают совокупность всех ключей, действующих в системе.

Под ключевой информацией понимают совокупность всех ключей, действующих в системе.

Генерация ключей должна производиться таким образом, чтобы предугадать значение ключа (даже зная, как он будет генерироваться) было практически невозможно. В идеальном случае, вероятность выбора конкретного ключа из множества допустимых равна 1/N, где N – мощность ключевого множества (число его элементов).

Для получения ключей используют аппаратные и программные средства генерации случайных значений. Для систем с высокими требованиями к уровню безопасности более предпочтительными считаются аппаратные датчики, основанные на случайных физических процессах. В то же время, из-за дешевизны и возможности неограниченного тиражирования наиболее распространенными являются программные реализации.

Рекомендуется регулярно проводить замену ключей, используемых в системе. В некоторых случаях вместо замены допустимо использовать процедуру модификации. Модификация ключа – генерация нового ключа из предыдущего значения с помощью односторонней функции (т. е. такой функции для которой обратное преобразование вычислить практически невозможно)

При организации хранения ключей симметричного шифрования необходимо обеспечить такие условия работы, чтобы секретные ключи никогда были записаны в явном виде на носителе, к которому может получить доступ нарушитель. Например, это требование можно выполнить, создавая иерархии ключей. Трехуровневая иерархия подразумевает деление ключей на:

- главный ключ;

- ключ шифрования ключей;
- ключ шифрования данных (сеансовый ключ).

Сеансовые ключи – нижний уровень иерархии – используются для шифрования данных и аутентификации сообщений. Для защиты этих ключей при передаче или хранении используются ключи шифрования ключей, которые никогда не должны использоваться как сеансовые. На верхнем уровне иерархии располагается главный ключ (или мастер-ключ). Его применяют для защиты ключей второго уровня.

Для защиты главного ключа в системах, использующих только симметричные шифры, приходится применять не криптографические средства, а например, средства физической защиты данных (ключ записывается на съемный носитель, который после окончания работы изымается из системы и хранится в сейфе, и т. п.). В относительно небольших информационных системах может использоваться двухуровневая иерархия ключей (главный и сеансовые ключи).

При распределении ключей необходимо выполнить следующие требования:

- обеспечить оперативность и точность распределения ключей;
- обеспечить секретность распределения ключей.

Распределение ключей может производиться:

- с использованием одного или нескольких центров распределения ключей (централизованное распределение);
- прямым обменом сеансовыми ключами между пользователями сети (децентрализованное распределение ключей).

Децентрализованное распределение ключей симметричного шифрования требует наличия у каждого пользователя большого количества ключей (для связи с каждым из абонентов системы), которые необходимо сначала безопасно распределить, а потом обеспечивать их секретность в процессе хранения.

Централизованное распределение ключей симметричного шифрования подразумевает, что у каждого пользователя есть только один основной ключ для взаимодействия с центром распределения ключей.

Для обмена данными с другим абонентом, пользователь обращается к серверу ключей, который назначает этому пользователю и соответствующему абоненту сеансовый симметричный ключ. Одной из самых известных систем централизованного распределения ключей является Kerberos.

Протокол Kerberos сейчас является фактическим стандартом системы централизованной аутентификации и распределения ключей симметричного шифрования. Поддерживается операционными системами семейства Unix, Windows (начиная с Windows’2000), есть реализации для Mac OS.

Протокол Kerberos обеспечивает распределение ключей симметричного шифрования и проверку подлинности пользователей, работающих в незащищенной сети. Реализация Kerberos – это программная система, построенная по архитектуре «клиент-сервер».

Клиентская часть устанавливается на все компьютеры защищаемой сети, кроме тех, на которые устанавливаются компоненты сервера

Kerberos. В роли клиентов Kerberos могут, в частности, выступать и сетевые серверы (файловые серверы, серверы печати и т. д.).

Серверная часть Kerberos называется центром распределения ключей (англ. «Key Distribution Center», сокр. KDC) и состоит из двух компонент:

- сервер аутентификации (англ. «Authentication Server», сокр. AS);
- сервер выдачи разрешений (англ. «Ticket Granting Server», сокр. TGS).

Каждому субъекту сети сервер Kerberos назначает разделяемый с ним ключ симметричного шифрования и поддерживает базу данных субъектов и их секретных ключей. Схема функционирования протокола Kerberos представлена на рис. 25.

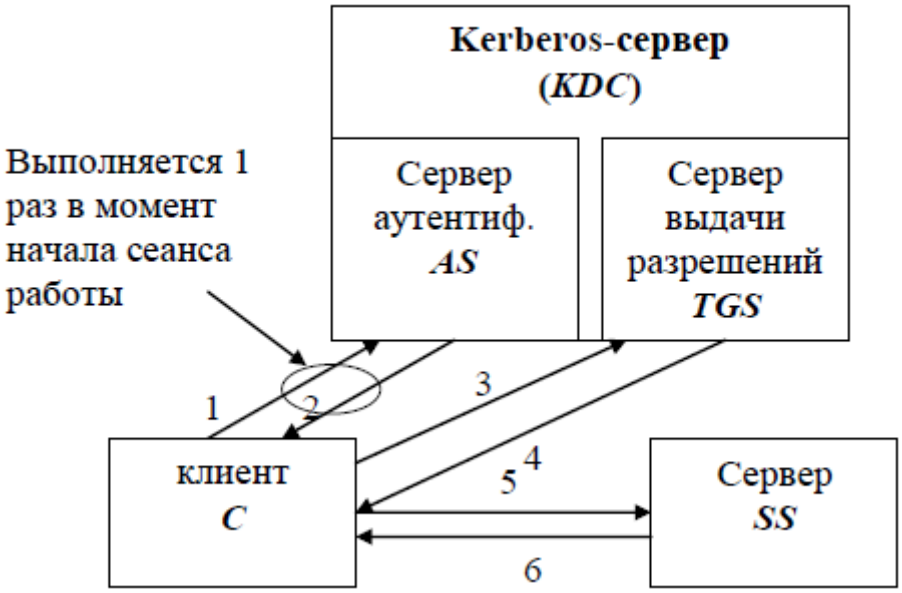

Рис. 25. Протокол Kerberos

При использовании протокола Kerberos компьютерная сеть логически делится на области действия серверов Kerberos. Kerberos-область – это участок сети, пользователи и серверы которого зарегистрированы в базе данных одного сервера Kerberos (или в одной базе, разделяемой несколькими серверами). Одна область может охватывать сегмент локальной сети, всю локальную сеть или объединять несколько связанных локальных сетей. [9] Схема взаимодействия между Kerberos-областями представлена на рис. 26.

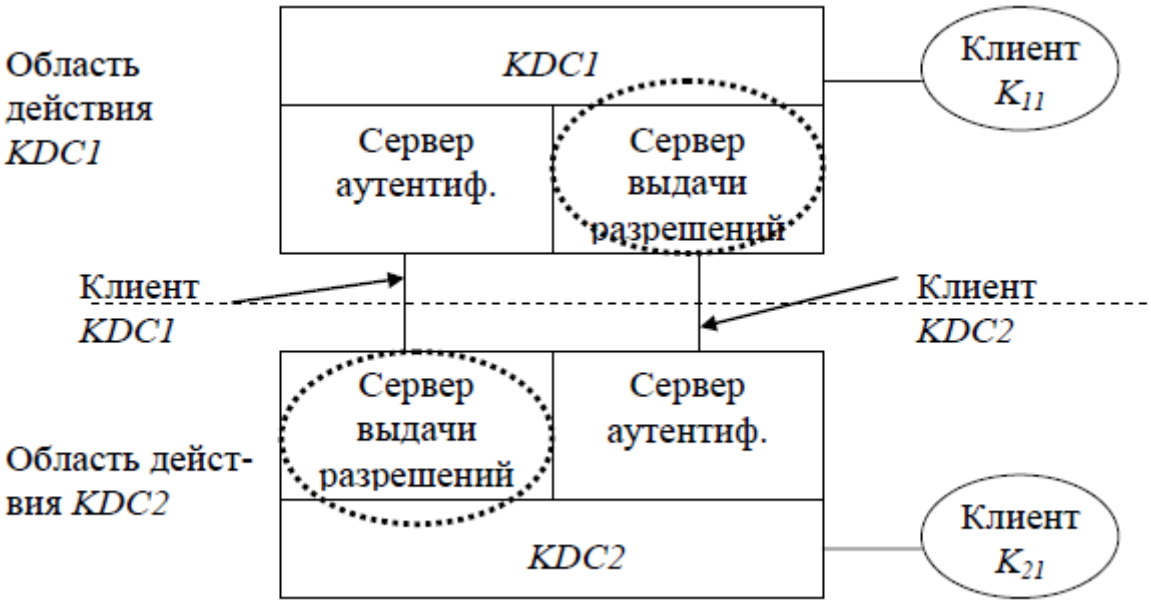

Рис.26. Взаимодействие между Kerberos-областями

2.3. Поточные шифры

Поточный (потоковый) шифр – это симметричный шифр, в котором каждый символ открытого текста преобразуется в символ шифрованного текста в зависимости не только от используемого ключа, но и от его расположения в потоке открытого текста. Поточный шифр реализует другой подход к симметричному шифрованию, нежели блочные шифры.

Специалисты используют термин «потоковый шифр» только в том случае, когда шифруемые символы открытого текста малы (одна буква, бит или байт).

Авторитетный швейцарский криптограф Райнер Рюппель главное отличие потоковых и блоковых шифров сформулировал так: блоковые шифры применяют одно и то же криптографическое преобразование к большим блокам данных открытого текста; потоковые шифры реализуют сменное по времени преобразование отдельных битов (байтов) открытого текста.

Так как посимвольное шифрование при потоковом шифровании данных не приводит к задержкам в криптосистеме, то важнейшее достоинство потоковых шифров – высокая скорость шифрования, соизмеримая со скоростью поступления входной информации. Это позволяет обеспечить шифрование практически в реальном масштабе времени вне зависимости от объема информации и разрядности потока данных.

Классические примеры потоковых шифров – шифр Вернама, или одноразовый блокнот. В Шифре Вернама по модулю m знаки открытого текста, криптограммы и ключа отождествляются с элементами кольца вычетов Z_m, длина n открытого текста и ключа одинакова, а уравнение шифрования имеет вид $y_i = (x_i + k_i) \bmod m$, $i = 1,2,...,n$.

Дешифрование сводится к вычислению $x_i = (y_i - k_i) \bmod m$.

Другое название ключевой последовательности – гамма, шифра – шифр модульного гаммирования, операции наложения гаммы на текст – гаммирование.

Кроме сложения при шифровании возможно вычитание гаммы:

$y_i = x_i - k_i \pmod m$ или $y_i = k_i - x_i \pmod m$.

Если каждый знак ключа $k_1,k_2,...,k_n$ выбирается случайно, то вскрыть шифр невозможно (т.к. все возможные открытые тексты будут равновероятны).

Однократно используемый шифр Вернама по модулю 2 называют одноразовым шифровальным блокнотом. Ключ такого блокнота должен обладать тремя важными свойствами:

1. Быть истинно случайным;
2. Совпадать по размеру с заданным открытым тестом;
3. Использоваться однократно.

Уже отмечалось, что если для гаммы последовательность битов выбирается случайно и длина гаммы равна длине сообщения, то взломать шифр невозможно. Но у данного режима шифрования есть и отрицательная особенность – проблемы с передачей и хранением ключей, ведь ключи, сравнимые по длине с передаваемыми сообщениями, трудно использовать на практике. Поэтому основная идея современных потоковых шифров – реализовать концепцию одноразового блокнота, используя секретный ключ меньшей длины, из которого для гаммы генерируется псевдослучайная числовая последовательность, похожая на случайную.

Самые популярные потоковые шифры можно назвать двоичными аддитивными: в них секретный ключ K используется только для управления генератором псевдослучайной числовой последовательности, порождающим битовую последовательность $k_1,k_2,...,k_n$, называемую ключевым потоком или гаммой. Поток битов $k_1,k_2,...,k_n$ далее суммируется по модулю 2 с потоком битов открытого текста $x_1, x_2,..., x_n$ (операция XOR аппаратно реализована в арифметико-логическом устройстве процессора). В итоге появляется поток битов шифрованного текста $y_1, y_2,..., y_n$, где $y_i = x_i \oplus \gamma_i$. Для дешифрования получатель над битами шифротекста и той же самой гаммы повторно выполняют операцию XOR $x_i = y_i \oplus \gamma_i$. Схема потокового шифрования в режиме гаммирования представлена на рисунке 27. Фактически секретный ключ небольшой размерности исполняет роль

«зародыша», порождающего значительно более длинную ключевую последовательность.

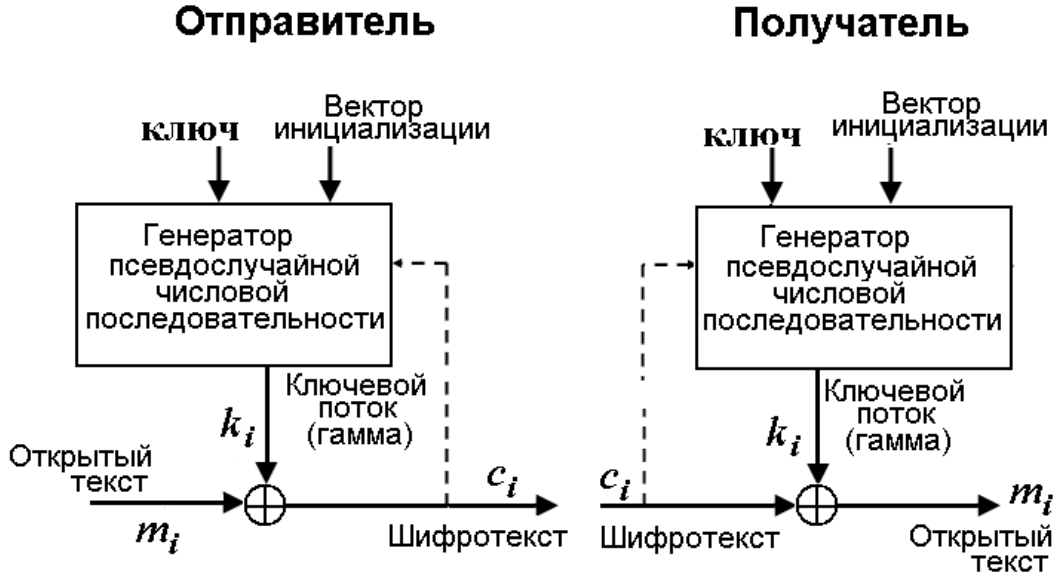

Рис. 27. Схема потокового шифрования в режиме гаммирования

Повторное использование одной и той же гаммы при потоковом шифровании, как отмечалось, может вызвать катастрофическую потерю стойкости шифрования. К сожалению, сменить ключи перед началом шифрования новых сообщений не всегда технично возможно. Чтобы примирить эти обстоятельства, в потоковых шифрах начали использовать «начальное состояние» шифратора. Действительно, двойной запуск шифратора без смены ключа, но с разными начальными состояниями, даст две абсолютно разные гаммы. Комбинацию знаков, передаваемую по каналу связи и предназначенную для вхождения в связь шифрующей аппаратуры на стороне отправителя и получателя, называют вектором инициализации (синхропосылкой). Его передают перед началом сообщения и сохраняют вместе с шифротекстом. Открытая передача вектора инициализации в том виде, как он используется в алгоритме шифрования, может снизить стойкость системы. Поэтому вектор инициализации закрывают с помощью дополнительного ключа (ключ инициализации) или применяют в алгоритме модификации векторов, зависящие от такого ключа. Таким образом, получатель и отправитель корреспонденции могут вычислять гамму по известному алгоритму на основе ключа и вектора инициализации (вектор часто является меткой времени). [11]

2.3.1. Типы потоковых шифров.

Гаммирование чувствительно к синхронизации гаммы и шифротекста. Если в ходе передачи будет поврежден один символ шифротекста, то это не повлияет на правильность дешифрования остальных символов. Если же при передаче потерять хотя бы один знак криптограммы, то весь дальнейший текст дешифруется неправильно. Поэтому необходимо обеспечить синхронизацию зашифрования и расшифрования текстов.

По способу синхронизации потоковые шифры классифицируют на 2 типа: 1) ***Синхронные потоковые шифры***, в которых знаки генерируемой ключевой гаммы не зависят от открытого и шифрованного текстов, а зависят только от исходного секретного ключа шифра, т.е. $k_i = f(K)$.

Плюсы

- отсутствие эффекта распространения ошибок (только искажённый бит будет расшифрован неверно);
- предохраняют от любых вставок и удалений шифротекста, так как они приведут к потере синхронизации и будут обнаружены.

Минусы:

- уязвимы к изменению отдельных бит шифрованного текста. Если злоумышленнику известен открытый текст, он может изменить эти биты так, чтобы они расшифровывались, как ему надо.

2) ***Самосинхронизирущиеся потоковые шифры***, в которых знаки ключевой гаммы зависят от исходного секретного ключа шифра K и от конечного числа последних знаков шифрованного текста, т.е. $k_i = f(K, y_{i-1}, y_{i-2},..., y_{i-L})$. В этом случае на стороне получателя генератор начнет синхронно работать с передающей стороной после получения L битов.

Плюсы:

- Размешивание статистики открытого текста. Так как каждый знак открытого текста влияет на следующий шифротекст, статистические свойства открытого текста распространяются на весь шифротекст. Следовательно, АПШ может быть более устойчивым к атакам на основе избыточности открытого текста, чем СПШ.

Минусы:

- Распространение ошибки (каждому неправильному биту шифротекста соответствуют N ошибок в открытом тексте);
- чувствительны к вскрытию повторной передачей.

Самосинхронизирующиеся потоковые шифры можно реализовать в виде блокового шифра в режиме CFB (обратной связи по шифротексту), при этом за один раз может шифроваться произвольное число битов, меньшее либо равное длине блока. [11]

Основные отличия поточных шифров от блочных

Большинство существующих шифров с секретным ключом однозначно могут быть отнесены либо к поточным, либо к блочным шифрам. Но теоретическая граница между ними является довольно размытой. Например, используются алгоритмы блочного шифрования в режиме поточного шифрования (пример: для алгоритма DES режимы CFB и OFB). Рассмотрим основные различия между поточными и блочными шифрами не только в аспектах их безопасности и удобства, но и с точки зрения их изучения в мире:

- важнейшим достоинством поточных шифров перед блочными является высокая скорость шифрования, соизмеримая со скоростью поступления входной информации; поэтому, обеспечивается шифрование практически в реальном масштабе времени вне зависимости от объема и разрядности потока преобразуемых данных.
- в синхронных поточных шифрах (в отличие от блочных) отсутствует эффект размножения ошибок, то есть число искаженных элементов в расшифрованной последовательности равно числу искаженных элементов зашифрованной последовательности, пришедшей из канала связи.
- структура поточного ключа может иметь уязвимые места, которые дают возможность криптоаналитику получить дополнительную информацию о ключе (например, при малом периоде ключа криптоаналитик может использовать найденные части поточного ключа для дешифрования последующего закрытого текста).
- Потоковые в отличие от блочных часто могут быть атакованы при помощи линейной алгебры (так как выходы отдельных регистров сдвига с обратной линейной связью могут иметь корреляцию с гаммой). Также для взлома поточных шифров весьма успешно применяется линейный и дифференциальный анализ. [10]

2.3.2.Алгоритм RC4

RC4 – широко используется для обеспечения безопасности финансовых транзакций в Интернет. Поток ключей в этом алгоритме не зависит от открытого текста.

Основные преимущества шифра – высокая скорость работы и переменный размер ключа.

Алгоритм RC4 строится, как и любой потоковый шифр, на основе параметризованного ключом генератора псевдослучайных битов с равномерным распределением.

Ядро алгоритма поточных шифров состоит из функции – генератора псевдо случайных битов (гаммы), который выдаёт поток битов ключа (ключевой поток, гамму, последовательность псевдо случайных битов).

Алгоритм шифрования.

1. Функция генерирует последовательность битов (k_i).
2. Затем последовательность битов посредством операции «суммирование по модулю два» (xor) объединяется с открытым текстом (m_i). В результате получается шифрограмма (c_i): $c_i=m_i \oplus k_i$

Алгоритм расшифровки.

1. Повторно создаётся (регенерируется) поток битов ключа (ключевой поток) (k_i).
2. Поток битов ключа складывается с шифрограммой (c_i) операцией «xor». В силу свойств операции «xor» на выходе получается исходный (не зашифрованный) текст (m_i): $m_i = c_i \oplus k_i = (m_i \oplus k_i) \oplus k_i$.

RC4 – фактически класс алгоритмов, определяемых размером блока (в дальнейшем S-блока). Параметр *n* является размером слова для алгоритма и определяет длину *S-блока*. Обычно, *n* = 8, но в целях анализа можно уменьшить его. Однако для повышения безопасности необходимо увеличить эту величину. В алгоритме нет противоречий на увеличение размера *S-блока* . При увеличении *n*, допустим, до 16 бит, элементов в *S-блоке* становится 65 536 и соответственно время начальной итерации будет увеличено. Однако, скорость шифрования возрастёт.

Внутреннее состояние RC4 представляется в виде массива размером 2^n и двух счётчиков. Массив известен как *S-блок*, и далее будет обозначаться как S. Он всегда содержит перестановку 2^n возможных значений слова. Два счётчика обозначены через i и j.

Инициализация RC4 состоит из двух частей:

1. инициализация *S-блока*;
2. генерация псевдо-случайного слова K.

Используется S-блок: S(0),....,S(255).Элементы представляют собой перестановку чисел от 0 до 255, а перестановка является функцией ключа переменной длины. В алгоритме применяются два счётчика (i и j) с нулевыми начальными значениями. Алгоритм шифрования представлен на рисунке 28. Сплошные стрелки означают передачу значений между элементами схемы (присваивание), пунктирные стрелки – индексацию в массиве.

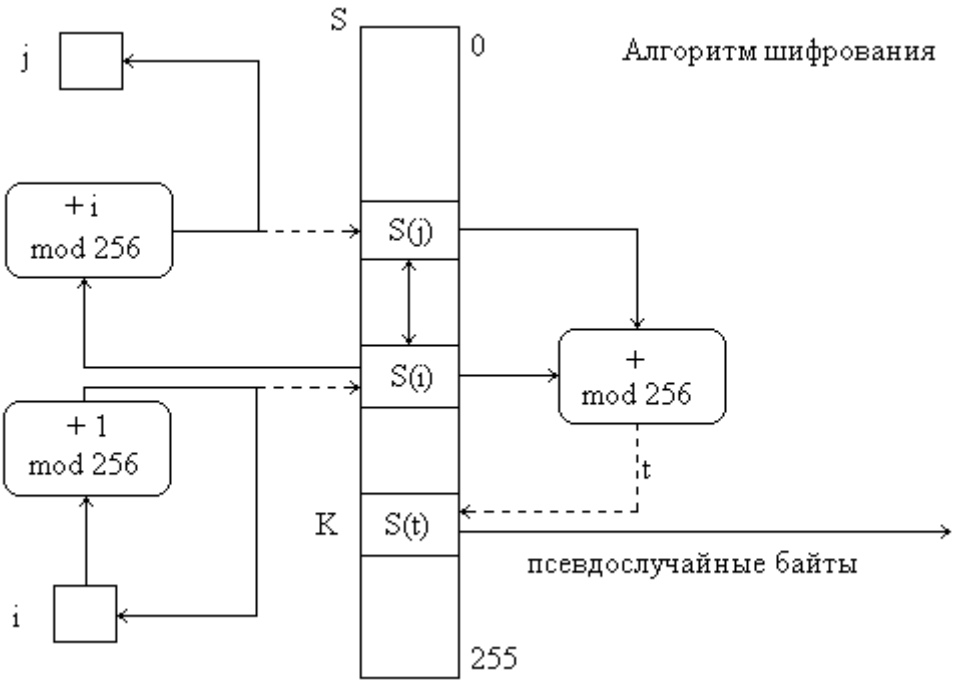

Рис. 28. Алгоритм шифрования RC4

Инициализация *S-блока*

Алгоритм также известен как «key-scheduling algorithm» или «KSA». Этот алгоритм использует ключ, подаваемый на вход пользователем, сохранённый в Key, и имеющий длину L байт. Инициализация начинается с заполнения массива S, далее этот массив перемешивается путём перестановок, определяемых ключом. Так как только одно действие выполняется над S, то должно выполняться утверждение, что S всегда содержит один набор значений, который был дан при первоначальной инициализации (*S[i] := i*).

for i **from** 0 **to** 255
  S[i] := i
endfor
j := 0
for i **from** 0 **to** 255
  j := (j + S[i] + Key[i mod L]) mod 256 // $n = 8$; $2^8 = 256$
  поменять местами S[i] и S[j]
endfor

Генерация псевдо-случайного слова *K*

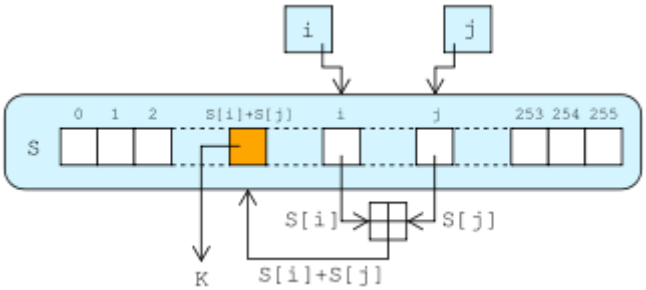

Рис. 29. Генератор ключевого потока RC4

Генератор ключевого потока RC4 переставляет значения, хранящиеся в S. В одном цикле RC4 определяется одно *n*-битное слово K из ключевого потока. В дальнейшем ключевое слово будет сложено по модулю два с исходным текстом, которое пользователь хочет зашифровать, и получен зашифрованный текст.

i := 0
j := 0
while Цикл генерации:
i := (i + 1) mod 256
j := (j + S[i]) mod 256
поменять местами S[i] и S[j]
t := (S[i] + S[j]) mod 256
K := S[t]
сгенерировано псевдослучайное слово K (для n = 8 будет сгенерирован один байт)
endwhile

RC4A

В 2004 году свет увидела работа Souradyuti Paul и Bart Preneel, в которой предлагалась модификация RC4A.

Для RC4A используется два *S-блока* вместо одного, как в RC4, обозначим S_1 и S_2. Для них соответствующе используются два счётчика j_1, j_2. Счётчик i, как и для RC4, используется в единственном числе для всего алгоритма. Принцип выполнения алгоритма остается прежним, но имеется ряд отличий:

1. S_1 является параметром для S_2.
2. За одну итерацию, то есть за одно увеличение индекса i, генерируется два байта шифротекста.

RC4+

В 2008 году была разработана и предложена модификация RC4+. Авторы Subhamoy Maitra и Goutam Paul модифицировали инициализацию *S-блока*(KSA+), использовав 3-уровневое скремблирование. Также модификации был подвергнут алгоритм генерации псевдослучайного слова (PRGA+).

Алгоритм:

Все арифметические операции выполняются по mod 256. Символами «<<» и «>>» обозначены битовые сдвиги влево и вправо соответственно. Символ «⊕» обозначает операцию «исключающее ИЛИ»

while Цикл генерации:

```
i := i + 1
a := S[i]
j := j + a
b := S[j]
S[i] := b     (поменяли местами S[i] и S[j])
S[j] := a
c := S[ i<<5 ⊕ j>>3 ] + S[ j<<5 ⊕ i>>3 ]
output ( S[a+b] + S[c⊕0xAA] ) ⊕ S[ j+b ]
```

endwhile

Криптосистемы и протоколы, использующие RC4: WEP; BitTorrent protocol encryption; Microsoft point-to-point encryption; браузер Opera Min; протокол SSL (вариативно); протокол SSH (вариативно); протокол RDP; Kerberos (вариативно); SASL mechanism digest-MD5 (вариативно);формат PDF; Skype (в модифицированном виде). [7]

2.4. АСИММЕТРИЧНЫЕ ШИФРЫ

В отличие от симметричных, в асимметричных алгоритмах ключи используются парами – открытый ключ (англ. «public key») и секретный или закрытый (англ. «private key»).

Первой публикацией в области криптографии с открытым ключом принято считать статью Уитфилда Диффи и Мартина Хеллмана «Новые направления в криптографии», вышедшую в свет в 1976 году.

Схема шифрования будет выглядеть следующим образом.

Получатель *B* генерирует пару ключей – открытый K_{B_pub} и секретный K_{B_pr}. Процедура генерация ключа должна быть такой, чтобы выполнялись следующие условия:

1) ключевую пару можно было бы легко сгенерировать;

2) сообщение, зашифрованное на открытом ключе, может быть расшифровано только с использованием секретного ключа;

3) зная только открытый ключ, невозможно рассчитать значение секретного.

После генерации ключей, абонент B передает открытый ключ отправителю A, а секретный ключ надежно защищает и хранит у себя (рис. 30). Пересылка открытого ключа может осуществляться по незащищенному каналу связи. Отправитель A, зная сообщение M и

открытый ключ, может рассчитать криптограмму C = E(M, K_{B_pub}) и передать ее получателю B. Получатель, зная секретный ключ, может расшифровать криптограмму M = D(C, K_{B_pr}).

Нарушитель, даже в том случае, если он смог перехватить криптограмму и открытый ключ, не может расшифровать криптограмму.

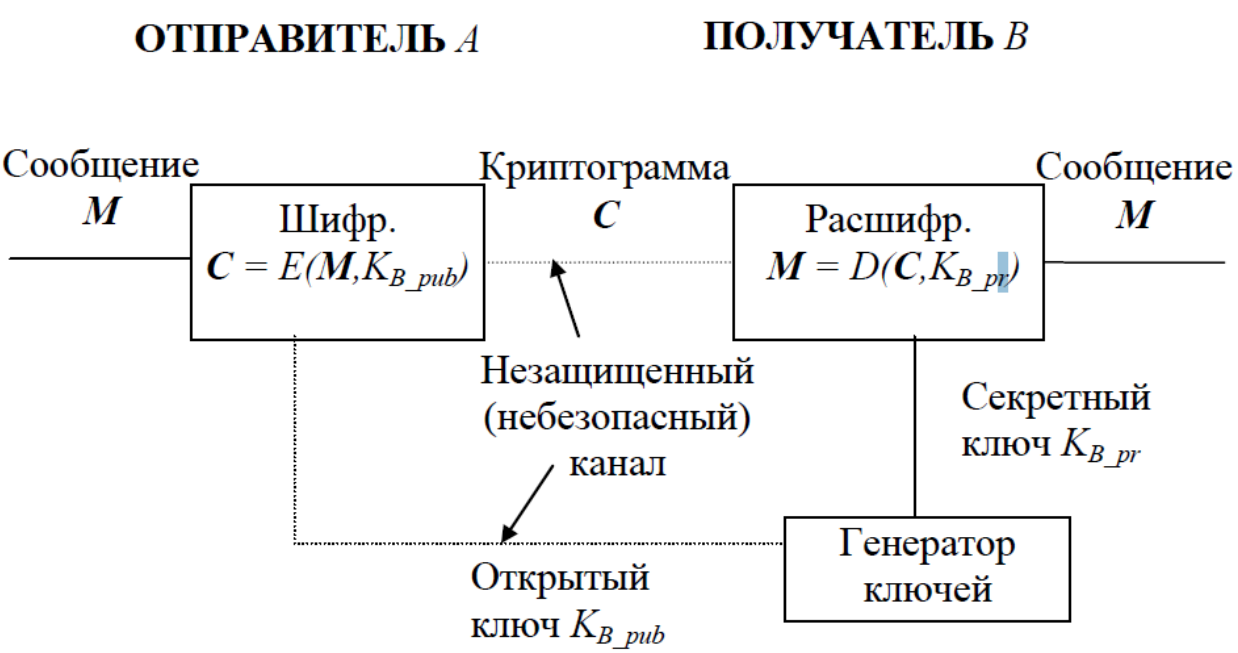

Рис. 30. Асимметричное шифрование

Рассмотрим ряд определений.

Односторонней (однонаправленной) функцией называется такая функция F:$X \to Y$, для которой выполняются следующие условия:

1) для всякого x∈X легко вычислить значение функции y = F(x), где y∈Y;

2) для произвольного y∈Y невозможно (чрезвычайно сложно) найти значение x∈X, такое что x = F^{-1}(y) (т. е. найти значение функции обратной F).

Односторонней функцией с секретом k, называется такая функция F_k: X→Y, для которой выполняются следующие условия:

1) для всякого x∈X легко вычислить значение функции y = F_k(x), где y∈Y, даже в том случае, если значение k неизвестно;

2) не существует легкого (эффективного) алгоритма вычисления обратной функции F_k^{-1}(y) без знания секрета k;

3) при известном k вычисление F_k^{-1}(y) для y∈Y не представляет существенной сложности.

В частности, односторонняя функция с секретом может быть использована для шифрования информации. Пусть M – исходное

сообщение. Получатель выбирает одностороннюю функцию с секретом, и тогда любой, кто знает эту функцию, может зашифровать сообщение для данного получателя, вычислив значение криптограммы C = F_k(M). Расшифровать данную криптограмму может только законный получатель, которому известен секрет k. [9]

2.4.1. Распределение ключей по схеме Диффи-Хеллмана

Основы асимметричной криптографии были заложены американскими исследователями У.Диффи и М.Хеллманом. Ими был предложен алгоритм, позволяющий двум абонентам, обмениваясь сообщениями по небезопасному каналу связи, распределить между собой секретный ключ шифрования.

В алгоритме Диффи-Хеллмана в качестве односторонней функции используется возведение в степень по модулю простого числа:

y =a ^{x} mod p.

Здесь p – большое простое число (сейчас считается безопасным использовать модуль порядка 2^{1024} или более), a – первообразный ко рень по модулю p. Задача нахождения обратного значения, т. е. вычисления x по известному y, называется задачей дискретного логарифмирования и является вычислительно сложной.

Пусть p – простое число, p>2, и известно разложение p-1 на простые множители: $p-1=\prod_{j=1}^{k} q_j^{\alpha_j}$. Число a является первообразным корнем по модулю p тогда и только тогда, когда выполняются следующие условия: НОД(a, p) =1; $a^{(p-1)/q_j} \neq 1(mod\ p)$, j =1,...,k , где НОД(x,y) – наибольший общий делитель чисел x и y.

Рассмотрим теперь алгоритм Диффи-Хеллмана по шагам. Пусть абоненты сети Алиса и Боб предварительно согласовали значения a и p. При этом требуется, чтобы разложение числа (p-1) содержало большой простой множитель, например, (p-1)=2t, где t – простое.

1. Алиса выбирает секретный ключ X_A и вычисляет соответствующий ему открытый ключ $Y_A = a^{X_A} mod\ p$.

2. Боб в свою очередь определяет X_B и рассчитывает $Y_B = a^{X_B} mod\ p$.

3. Абоненты обмениваются открытыми ключами Y_A и Y_B.

4. Абоненты вычисляют общий секретный ключ. Алиса пользуется следующим соотношением: $K_{AB} = {Y_B}^{X_A} mod\ p$. Боб использует формулу: $K_{BA} = {Y_A}^{X_B} mod\ p$. Покажем, что K_{AB}=K_{BA}, воспользовавшись свойством ассоциативности операции умножения в конечном поле:

$$K_{AB} = {Y_B}^{X_A} mod\ p = (a^{X_B})^{X_A} mod\ p = (a^{X_A})^{X_B} mod\ p = K_{BA}.$$

Таким образом, стороны смогли распределить общий секретный ключ K_{AB}. Нарушитель, который может перехватить передаваемые открытые ключи Y_A и Y_B, должен попытаться по ним вычислить общий секретный ключ без знания секретных ключей абонентов. На данный момент не найдено существенно лучшего пути решения данной задачи, чем дискретное логарифмирование, что и обеспечивает криптографическую стойкость алгоритма. [9]

2.4.3. Шифр RSA

Алгоритм RSA был предложен в 1977 году и стал первым полноценным алгоритмом асимметричного шифрования и электронной цифровой подписи.

Стойкость алгоритма основывается на вычислительной сложности задачи факторизации (разложения на множители) больших чисел и задачи дискретного логарифмирования.

Криптосистема основана на теореме Эйлера, которая гласит, что для любых взаимно простых чисел M и n (M<n) выполняется соотношение:

$M^{\varphi(n)}=1(\mathrm{mod}\ n)$, где $\varphi(n)$ - функция Эйлера. Эта функция равна количеству натуральных чисел меньших n, которые взаимно просты с n.

По определению, $\varphi(1)=1$. Также доказано, что для любых двух натуральных взаимнопростых чисел a и b выполняется равенство

$\varphi(a \cdot b) = \varphi(a) \cdot \varphi(b)$.

Алгоритм строится следующим образом. Пусть M – блок сообщения, $0 \le M < n$. Он шифруется в соответствии с формулой: $C = M^e \pmod n$.

В этом случае e – открытый ключ получателя. Тогда соответствующий ему секретный ключ d должен быть таким, что

$M = C^d \pmod n = (M^e)^d \pmod n = M^{ed} \pmod n$.

Как уже отмечалось, теорема Эйлера утверждает, что

$M^{\varphi(n)}=1(\mathrm{mod}\ n)$ или, что то же самое, $M^{k\varphi(n)}=1(\mathrm{mod}\ n)$ где k – натуральный множитель. Сопоставив выражения, получаем, что e и d должны быть связаны соотношением: $e \cdot d = k \cdot \varphi(n) + 1 \leftrightarrow ed = 1(\mathrm{mod}\varphi(n))$. Теперь предположим, что $n = p \cdot q$, где p и q – простые числа, причем $p \neq q$. Нетрудно показать, что для простого числа $p \neq 1$ функция Эйлера будет равна $\varphi(p) = p - 1$. Тогда, учитывая, что p и q – простые и не равны друг другу (а значит, они и взаимно простые), будет справедливо равенство: $\varphi(p \cdot q) = \varphi(p) \cdot \varphi(q) = (p-1) \cdot (q-1)$.

Рассмотрим теперь алгоритм RSA «по шагам». Первым этапом является генерация ключей.

1) Выбираются два больших простых числа p и q, $p \neq q$.

2) Вычисляется модуль n: $n = p \cdot q$. Обратите внимание, что для криптосистемы RSA модуль n является частью открытого ключа и должен меняться при смене ключевой пары.

3) Случайным образом выбирается число d, такое что

$1 < d < (p-1)(q-1)$ и $НОД(d, (p-1)(q-1)) = 1$.

4) Вычисляется значение e такое что:

$1 < e < (p-1)(q-1)$

$e \cdot d = 1 \bmod ((p-1)(q-1))$

Доказано, что для случая, когда НОД(d, (p-1)(q-1))=1 такое e существует и единственно.

В результате выполнения данных вычислений имеем открытый ключ, представленный парой чисел (n, e) и секретный ключ d.

Шифрование производится следующим образом.

Отправителю известен открытый ключ получателя – (n,e). Пусть M – секретное сообщение, которое надо зашифровать $M<n$. Криптограмма вычисляется по формуле: $C = M^e \bmod n$.

Криптограмма C передается получателю. Владелец секретного ключа d может расшифровать сообщение по формуле $M = C^d \bmod n$. [7]

2.4.4. Шифр Эль-Гамаля

Криптографическая система Эль-Гамаля использует ту же математическую основу, что и рассмотренная ранее схема распределения ключей Диффи-Хеллмана: в качестве односторонней функции в этой криптосистеме используется возведение в степень по модулю большого простого числа.

Алгоритм **шифрования** строится следующим образом. Выбирается большое простое число p такое, что разложение числа $(p-1)$ содержит большой простой множитель, а также число a такое, что $1 < a < (p-1)$ и a – первообразный корень по модулю p.

Получатель сообщения генерирует ключевую пару: случайным образом выбирает секретный ключ x ($0 < x < p$) и вычисляет открытый ключ

$y = a^x \bmod p$.

Для шифрования сообщения M ($0 \le M < p$), отправитель должен выполнить следующие действия

1. Выбрать случайное число k такое, что $1 < k < p-1$, НОД(k,p-1)=1.
2. Вычислить вспомогательное значение $r = y^k \bmod p$.
3. Рассчитать значение криптограммы, состоящее из двух частей:

$c_1 = a^k \bmod p$, $c_2 = rM \bmod p$.

Рассмотрим процесс расшифровки. Для того, чтобы по c_2 определить M, получателю потребуется рассчитать значение r. С учетом того, что ему

известен секретный ключ *x* и значение c_1 это становится возможным: $c_1^x=(a^k)^x=r \bmod p$. Тогда $M=c_2r^{-1}= c_2(c_1^x)^{-1} \bmod p$. При использовании шифра Эль-Гамаля требуется, чтобы выбираемое в процессе шифрования число k, каждый раз менялось. В противном случае, если сообщения M и M' предназначены одному получателю, и нарушитель смог узнать одно из них, то ему не составит труда рассчитать и второе. [9]

2.5. Комбинированная криптосистема шифрования

Анализ рассмотренных выше особенностей симметричных и асимметричных криптографических систем показывает, что при совместном использовании они эффективно дополняют друг друга, компенсируя недостатки.

Действительно, главным достоинством асимметричных криптосистем с открытым ключом является их потенциально высокая безопасность: нет необходимости ни передавать, ни сообщать кому-либо значения секретных ключей, ни убеждаться в их подлинности. Однако их быстродействие обычно в сотни (и более) раз меньше быстродействия симметричных криптосистем с секретным ключом. [7]

В свою очередь, быстродействующие симметричные криптосистемы страдают существенным недостатком: обновляемый секретный ключ симметричной криптосистемы должен регулярно передаваться партнерам по информационному обмену и во время этих передач возникает опасность раскрытия секретного ключа.

Совместное использование этих криптосистем позволяет эффективно реализовывать такую базовую функцию защиты, как криптографическое закрытие передаваемой информации с целью обеспечения ее конфиденциальности. Комбинированное применение симметричного и асимметричного шифрования устраняет основные недостатки, присущие обоим методам, и позволяет сочетать преимущества высокой секретности, предоставляемые асимметричными криптосистемами с открытым ключом, с преимуществами высокой скорости работы, присущими симметричным криптосистемам с секретным ключом.

Метод комбинированного использования симметричного и асимметричного шифрования заключается в следующем.

Симметричную криптосистему применяют для шифрования исходного открытого текста, а асимметричную криптосистему с открытым ключом применяют только для шифрования секретного ключа симметричной криптосистемы. В результате асимметричная криптосистема с открытым ключом не заменяет, а лишь дополняет симметричную криптосистему с секретным ключом, позволяя повысить в

целом защищенность передаваемой информации. Такой подход иногда называют схемой электронного «цифрового конверта».

Пусть пользователь А хочет использовать комбинированный метод шифрования для защищенной передачи сообщения М пользователю В.

Тогда последовательность действий пользователей А и В будет следующей.

Действия пользователя А:

1. Он создает (например, генерирует случайным образом) сеансовый секретный ключ K_s, который будет использован в алгоритме симметричного шифрования для зашифрования конкретного сообщения или цепочки сообщений.

2. Зашифровывает симметричным алгоритмом сообщение М на сеансовом секретном ключе K_s.

3. Зашифровывает асимметричным алгоритмом секретный сеансовый ключ K_s на открытом ключе $K_в$ пользователя В (получателя сообщения).

4. Передает по открытому каналу связи в адрес пользователя В зашифрованное сообщение М вместе с зашифрованным сеансовым ключом K_s.

Действия пользователя А иллюстрируются схемой шифрования сообщения комбинированным методом (рис. 31).

Рис. 31. Схема шифрования сообщения комбинированным методом

Действия пользователя В (при получении электронного «цифрового конверта» – зашифрованного сообщения М и зашифрованного сеансового ключа K_s):

5. Расшифровывает асимметричным алгоритмом сеансовый ключ K_s с помощью своего секретного ключа $K_в$.

6. Расшифровывает симметричным алгоритмом принятое сообщение М с помощью полученного сеансового ключа K_s.

Действия пользователя В иллюстрируются схемой расшифровывания сообщения комбинированным методом (рис. 32).

Рис. 32. Схема расшифровывания сообщения комбинированным методом

Полученный электронный «цифровой конверт» может раскрыть только законный получатель – пользователь В. Только пользователь В, владеющий личным секретным ключом $к_в$ сможет правильно расшифровать секретный сеансовый ключ K_s и затем с помощью этого ключа расшифровать и прочитать полученное сообщение М.

При методе «цифрового конверта» недостатки симметричного и асимметричного криптоалгоритмов компенсируются следующим образом:

• проблема распространения ключей симметричного криптоалгоритма устраняется тем, что сеансовый ключ K_s, на котором шифруются собственно сообщения, передается по открытым каналам связи в зашифрованном виде; для зашифровывания ключа K_s используется асимметричный криптоалгоритм;

• проблемы медленной скорости асимметричного шифрования в данном случае практически не возникает, поскольку асимметричным криптоалгоритмом шифруется только короткий ключ K_s, а все данные шифруются быстрым симметричным криптоалгоритмом.

В результате получают быстрое шифрование в сочетании с удобным распределением ключей.

Когда требуется реализовать протоколы взаимодействия не доверяющих друг другу сторон, используется следующий способ взаимодействия. Для каждого сообщения на основе случайных параметров генерируется отдельный секретный ключ симметричного шифрования, который и зашифровывается асимметричной системой для передачи вместе с сообщением, зашифрованным этим ключом. В этом случае разглашение ключа симметричного шифрования не будет иметь смысла, так как для зашифровывания следующего сообщения будет использован другой случайный секретный ключ.

При комбинированном методе шифрования применяются криптографические ключи как симметричных, так и асимметричных криптосистем. Очевидно, выбор длин ключей для криптосистемы каждого

типа следует осуществлять таким образом, чтобы злоумышленнику было одинаково трудно атаковать любой механизм защиты комбинированной криптосистемы. [8]

2.6. ХЭШ-ФУНКЦИИ

Для очень большого количества технологий безопасности (применяются односторонние функции шифрования, называемые также хэш-функциями. Основное назначение подобных функций – получение из сообщения произвольного размера его дайджеста – значения фиксированного размера. Дайджест может быть использован в качестве контрольной суммы исходного сообщения, обеспечивая таким образом (при использовании соответствующего протокола) контроль целостности информации. Основные свойства хэш-функции:

1. на вход хэш-функции подается сообщение произвольной длины;
2. на выходе хэш-функции формируется блок данных фиксированной длины;
3. значения на выходе хэш-функции распределены по равномерному закону;
4. при изменении одного бита на входе хэш-функции существенно изменяется выход.

Кроме того, для обеспечения устойчивости хэш-функции к атакам она должна удовлетворять следующим требованиям:

1. если мы знаем значение хэш-функции h, то задача нахождения сообщения M такого, что H(M) = h, должна быть вычислительно трудной;
2. при заданном сообщении M задача нахождения другого сообщения M’, такого, что H(M) = H(M’), должна быть вычислительно трудной.

Если хэш-функция будет удовлетворять перечисленным свойствам, то формируемое ею значение будет уникально идентифицировать сообщения, и всякая попытка изменения сообщения при передаче будет обнаружена путем выполнения хэширования на принимающей стороне и сравнением с дайджестом, полученным на передающей стороне.

Еще одной особенностью хэш-функций является то, что они не допускают обратного преобразования – получить исходное сообщения по его дайджесту невозможно. Поэтому их называют еще односторонними функциями шифрования.

Хэш-функции строятся по итеративной схеме, когда исходное сообщение разбивается на блоки определенного размера, и над ними выполняются ряд преобразований с использованием как обратимых, так и необратимых операций. Как правило, в состав хэширующего преобразования включается сжимающая функция, поскольку его выход

зачастую по размеру меньше блока, подаваемого на вход. На вход каждого цикла хэширования подается выход предыдущего цикла, а также очередной блок сообщения. Таким образом, на каждом цикле выход хэш-функции h_i представляет собой хэш первых i блоков.

Если вспомнить, насколько рандомизируют входное сообщение блочные шифры, можно в качестве функции хэш-преобразования использовать какой-нибудь блочный шифр. То, что блочные шифры являются обратимыми преобразованиями, не противоречит свойствам хэш-функции, поскольку блочный шифр необратим по ключу шифрования, и, если в качестве ключа шифрования использовать выход предыдущего шага хэш-преобразования, а в качестве шифруемого сообщения очередной блок сообщения (или наоборот), то можно получить хэш-функцию с хорошими криптографическими характеристиками.

Такой подход использован, например, в российском стандарте хэширования – ГОСТ Р 34.11-94. Эта хэш-функция формирует 256-битное выходное значение, используя в качестве преобразующей операции блочный шифр ГОСТ 28147-89 (рис.33). Функция хэширования H получает на вход хэш, полученный на предыдущем шаге (значение h_0 произвольное начальное число), а также очередной блок сообщения m_i. Ее внутренняя структура представлена на рис. 34. Здесь в блоке шифрующего преобразования для модификации h_i в s_i используется блочный шифр ГОСТ 28147-89. Перемешивающее преобразование представляет собой модифицированную перестановку Фейстеля.

Для последнего блока m_N (N – общее количество блоков сообщения) выполняется набивка до размера 256 бит с добавлением истинной длины сообщения. Параллельно подсчитывается контрольная сумма сообщения $\sum$ и суммарная длина L, которые участвуют в финальной функции сжатия.

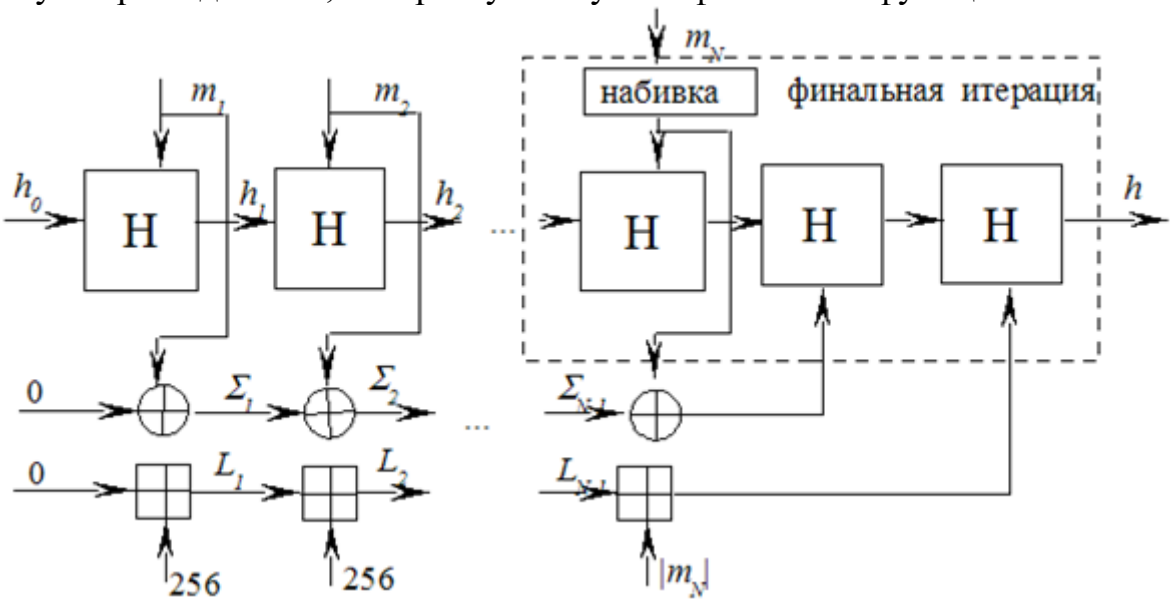

Рис. 33. Общая схема хэширования по ГОСТ Р 34.11-94

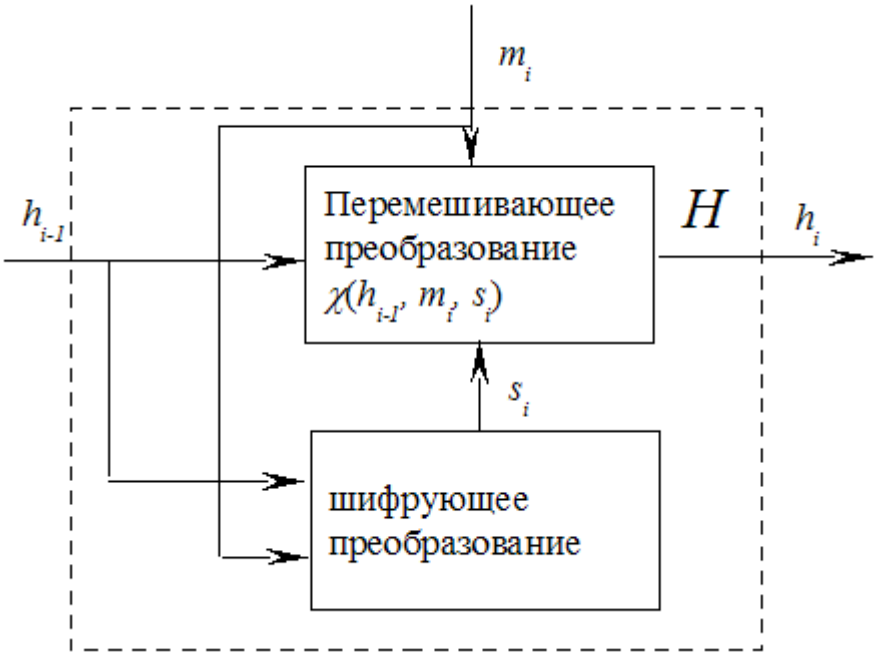

Рис. 34. Структура функции хэширования по ГОСТ Р 34.11-94

Основным недостатком хэш-функций на основе блочных шифров является невысокая скорость их работы. Поэтому были спроектированы ряд специализированных алгоритмов, которые, обеспечивая аналогичную стойкость к атакам, выполняют гораздо меньшее количество операций над входными данными и обеспечивают большую скорость работы. Примерами подобного рода алгоритмов являются: MD2, MD4, MD5, RIPEMD – 160, SHA.

Алгоритм применения хэш-функции заключается в следующем:

• перед отправлением сообщение обрабатывается при помощи хэш-функции. В результате получается его сжатый вариант (дайджест). Само сообщение при этом не изменяется и для передачи по каналам связи нуждается в шифровании;

• полученный дайджест шифруется закрытым ключом отправителя и пересылается получателю вместе с сообщением;

• получатель расшифровывает дайджест сообщения открытым ключом отправителя;

• получатель обрабатывает сообщение той же хэш-функцией, что и отправитель и в результате получает его дайджест. Если дайджест, присланный отправителем, и дайджест, полученный в результате обработки сообщения получателем, совпадают, это означает, что в сообщение не было внесено изменений [8]

2.6.1. Алгоритм SHA-1

SHA-1 реализует хеш-функцию, построенную на идее функции сжатия. Входами функции сжатия являются блок сообщения длиной 512 бит и выход предыдущего блока сообщения. Выход представляет собой значение всех хеш-блоков до этого момента. Иными словами хеш блока M_i равен $h_i=f(M_i, h_{i-1,})$. Хеш-значением всего сообщения является выход последнего блока

Описание алгоритма

Для сообщения произвольной длины *l*, не превышающей 2^{64} бит, алгоритм SHA-1 формирует 160-битный хеш-образ. Процедура формирования хеш-образа состоит из следующих шагов.

1. Исходное сообщение разбивается на блоки по 512 бит в каждом. Последний блок дополняется до длины, кратной 512 бит. Сначала добавляется 1 (бит), а потом нули, чтобы длина блока стала равной (512 - 64 = 448) бит. В оставшиеся 64 бита записывается длина исходного сообщения в битах (в big-endian формате). Если последний блок имеет длину более 448, но менее 512 бит, то дополнение выполняется следующим образом: сначала добавляется 1 (бит), затем нули вплоть до конца 512-битного блока; после этого создается ещё один 512-битный блок, который заполняется вплоть до 448 бит нулями, после чего в оставшиеся 64 бита записывается длина исходного сообщения в битах (в big-endian формате). Дополнение последнего блока осуществляется всегда, даже если сообщение уже имеет нужную длину.
2. Инициализируются пять 32-битовых переменных.
    - A = a = 0x67452301
    - B = b = 0xEFCDAB89
    - C = c = 0x98BADCFE
    - D = d = 0x10325476
    - E = e = 0xC3D2E1F0
    - Выполняется обработка очередных 512 бит исходного текста.

Для этого значения переменных *A, B, C, D, E* копируются в переменные *a, b, c, d, e* и далее для *t* от 1 до 80 выполняется преобразование значений данных переменных по схеме, изображенной на рис. 35.

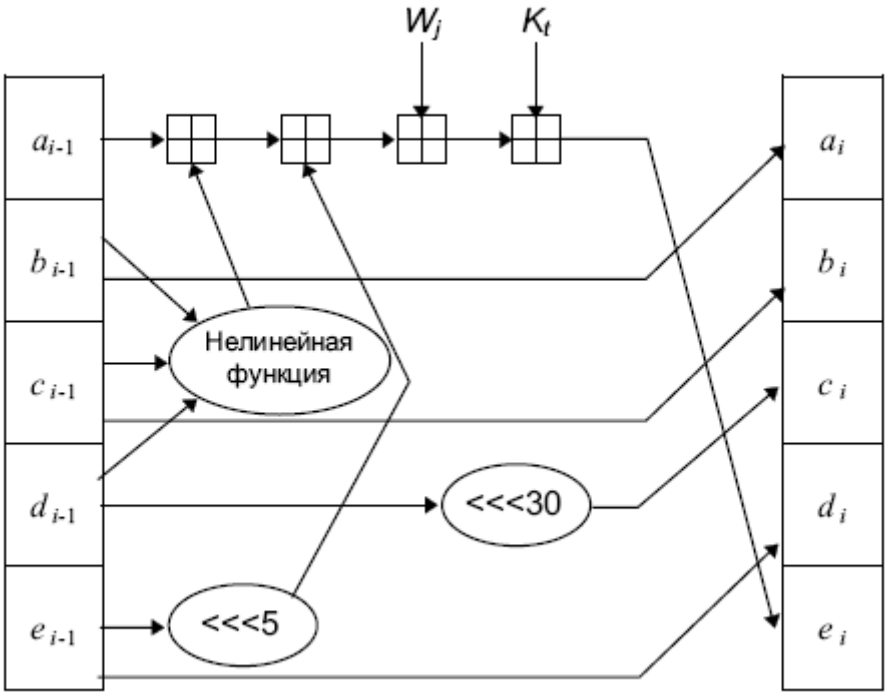

Рис. 35 . Схема итерации алгоритма SHA-1

Если t - номер операции (от 1 до 80), W_t представляет собой t-й подблок расширенного сообщения, а <<<s - циклический сдвиг влево на s битов, то главный цикл выглядит следующим образом:

для t **от** 0 **до** 79

temp = (a<<5) + f_t (b,c,d) + e +K_t
e = d
d = c
c = b<<30
b = a
a = temp

где «+» – операция сложения по модулю 2, $f_t(X, Y, Z)$ – нелинейная функция, имеющая следующий вид:

$$f_t(x,y,z) = \begin{cases} (x\&y)|(!\,x\&z), & t = \overline{1,20} \\ x \oplus y \oplus z, & t = \overline{21,40} \\ (x\&y)|(x\&z)|(y\&z), & t = \overline{41,60} \\ x \oplus y \oplus z, & t = \overline{61,80}\,, \end{cases}$$

где «&» – побитовая операция «И», «|» – побитовая операция «ИЛИ», «!» – операция побитового инвертирования, «⊕» – операция побитового сложения по модулю 2. Параметр K_t принимает четыре различных значения в зависимости от номера текущей итерации:

K_t = 5A82799916, $t = \overline{1,20}$;
K_t = 6ED9EBA116, $t = \overline{21,40}$;
K_t = 8F1BBCDC16, $t = \overline{41,60}$;
K_t = CA62C1D616, $t = \overline{61,80}$.

«<<<» – операция циклического сдвига на 30 либо 5 бит влево, W_t – одно из шестнадцати 32-битных слов 512-битного блока сообщения при $t = \overline{1,16}$, либо значение, определяемое в соответствии со следующим выражением при $t = \overline{17,80}$:

$$W_t = (W_t - 3 \oplus W_t - 8 \oplus W_t - 14 \oplus W_t - 16) <<< 1.$$

4. Значения переменных *a*, *b*, *c*, *d*, *e* независимо друг от друга складываются по модулю 2 со значениями переменных *A*, *B*, *C*, *D*, *E*, в которые затем и помещаются полученные результаты.
5. Шаги 3–4 выполняются до тех пор, пока не будет обработан весь текст.

После обработки последнего блока текста значение хеш-образа формируется как *ABCDE*.[4]

Использование

Хеш-функции используются в системах контроля версий, системах электронной подписи, а также для построения кодов аутентификации.

SHA-1 является наиболее распространенным из всего семейства SHA и применяется в различных широко распространенных криптографических приложениях и алгоритмах.

SHA-1 используется в следующих приложениях:S/MIME – дайджесты сообщений. SSL – дайджесты сообщений. IPSec – для алгоритма проверки целостности в соединении «точка-точка». SSH – для проверки целостности переданных данных. PGP – для создания электронной цифровой подписи. Git – для идентификации каждого объекта по SHA-1-хешу от хранимой в объекте информации. Mercurial – для идентификации ревизий. BitTorrent – для проверки целостности загружаемых данных. SHA-1 является основой блочного шифра SHACAL. [7]

2.7. Электронная цифровая подпись

Электронная цифровая подпись (ЭЦП) – это реквизит электронного документа, предназначенный для защиты данного электронного документа от подделки, полученный в результате криптографического преобразования информации с использованием закрытого ключа электронной цифровой подписи и позволяющий идентифицировать владельца сертификата ключа подписи, а также установить отсутствие искажения информации в электронном документе, а также обеспечивает неотказуемость подписавшегося.

Функции ЭЦП аналогичны обычной рукописной подписи:

- удостоверить, что подписанный текст исходит от лица, поставившего подпись;
- не дать лицу, подписавшему документ, возможности отказаться от обязательств, связанных с подписанным текстом;
- гарантировать целостность подписанного текста.

Важное отличие ЭЦП заключается в том, что электронный документ вместе с подписью может быть скопирован неограниченное число раз, при этом копия будет неотличима от оригинала.

Обобщенная схема системы ЭЦП представлена на рисунке 36.

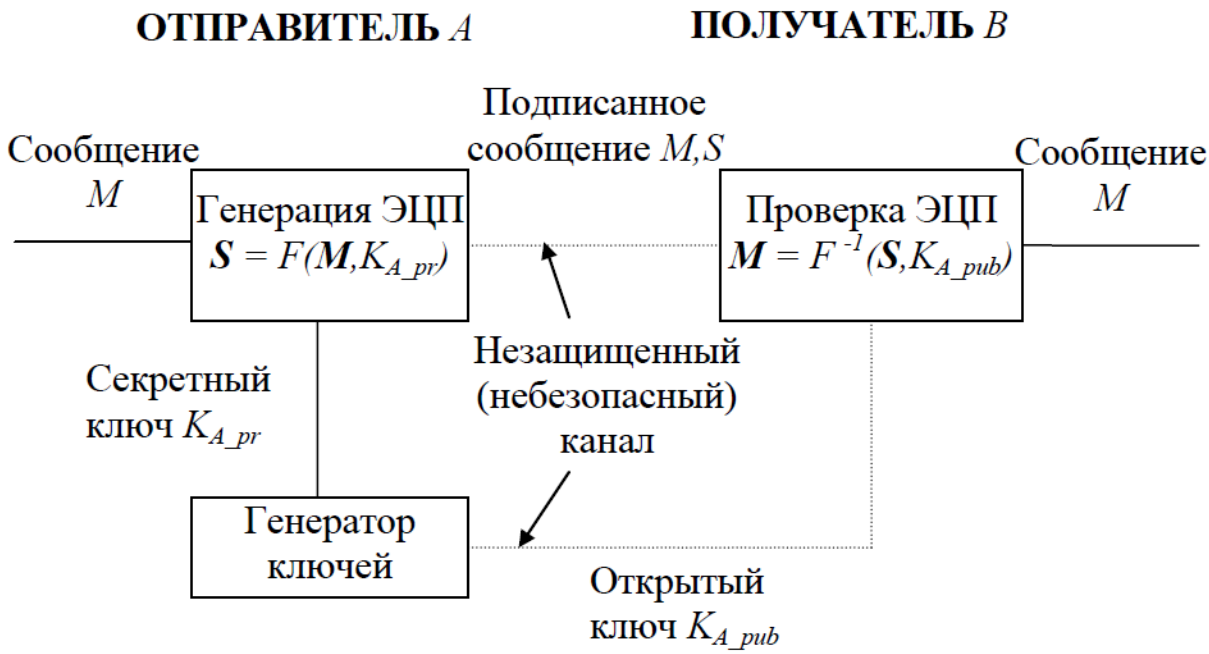

Рис. 36. Электронная цифровая подпись

В ходе преобразований здесь используется пара ключей отправителя сообщения. Тот факт, что при вычислении ЭЦП применяется секретный ключ отправителя, позволяет доказать происхождение и подлинность сообщения. Получатель, имея открытый ключ отправителя, проверяет ЭЦП, и если подпись корректна, то он может считать, что сообщение подлинное. [9]

2.7.1. Алгоритм формирования Электронно-цифровой подписи DSA

DSA (Digital Signature Algorithm) – алгоритм с использованием открытого ключа для создания электронной подписи, но не для шифрования. Секретное создание хеш-значения и возможность её публичной проверки означает, что только один субъект может создать

хеш-значение сообщения, но любой может проверить её корректность. Алгоритм DSA основывается на трудности вычисления дискретных логарифмов и является модификацией классической схемы Эль-Гамаля, где добавлено хэширование сообщения и дополнительное сравнение mod q , которое позволяет сделать подпись короче.

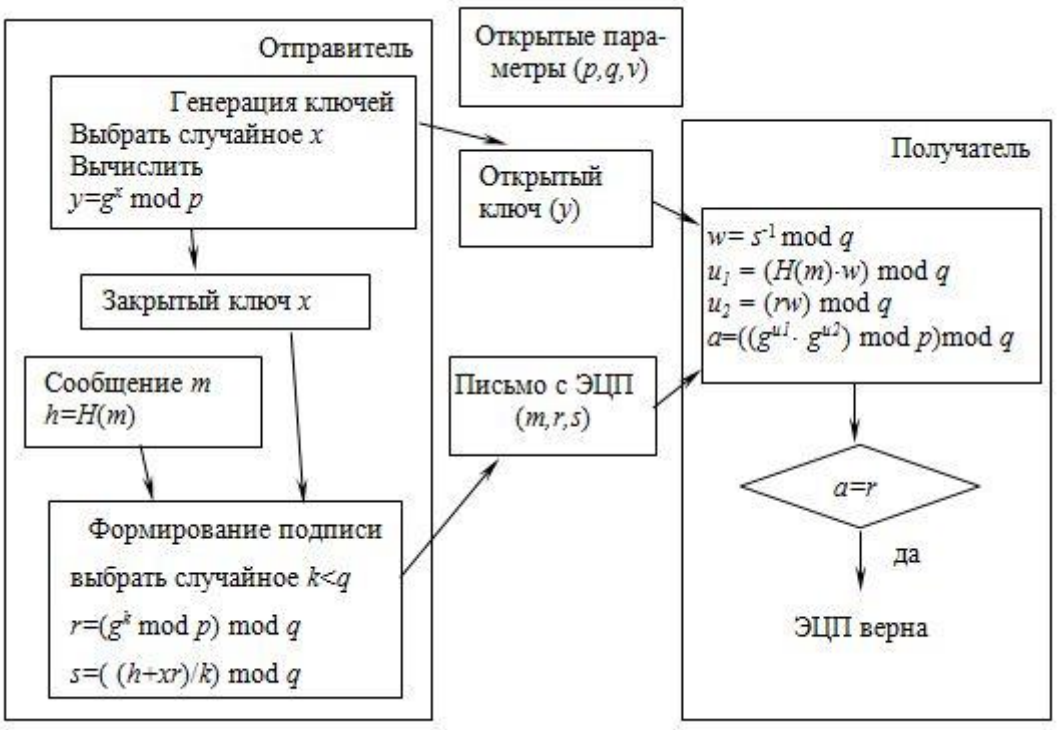

Рис.37. Схема алгоритма DSA

Для подписывания сообщений необходима пара ключей – открытый и закрытый. При этом закрытый ключ должен быть известен только тому, кто подписывает сообщения, а открытый – любому желающему проверить подлинность сообщения. Также общедоступными являются параметры самого алгоритма.

1. Выбор хеш-функции H(x). Для использования алгоритма необходимо, чтобы подписываемое сообщение являлось числом. Хеш-функция должна преобразовать любое сообщение в число.
2. Выбор большого простого числа q, размерность которого в битах совпадает с размерностью в битах значений хэш-функции H(x)
3. Выбор простого числа p, такого, что (p-1) делится на q. Размерность p задаёт криптостойкость системы. Ранее рекомендовалась длина в 1024 бита. В данный момент для систем, которые должны быть стойкими до 2030 года, рекомендуется длина в 2048 (3072) бита.
4. Выбор числа g такого, что его мультипликативный порядок по модулю p равен q. Для его вычисления можно воспользоваться формулой $g=h^{(p-1)/2} \bmod p$, где h – некоторое произвольное число, $h \in (1; p-1)$ такое, что

g≠1. В большинстве случаев значение h = 2 удовлетворяет этому требованию.

Подпись сообщения

Подпись сообщения выполняется по следующему алгоритму:

1. Выбор случайного числа k∈ (0;q)
2. Вычисление $r=(g^k \bmod p) \bmod p$.
3. Вычисление $s=(k^{-1}(H(m)+x*r)) \bmod q$
4. Выбор другого k, если оказалось, что r=0 или s=0

Подписью является пара чисел (r, s)

Проверка подписи

Проверка подписи выполняется по алгоритму:

1. Вычисление $w=s^{-1} \bmod p$
2. Вычисление $u_1=(H(m)*w) \bmod q$
3. Вычисление $u_2=(r*w) \bmod q$
4. Вычисление $v = \big((g^{u_1} g^{u_2}) mod\ p\big) mod\ q$

Подпись верна, если v = r. [11]

3. ЗАЩИТА ИНФОРМАЦИИ В IP-СЕТЯХ

3.1. ПРОТОКОЛ ЗАЩИТЫ ЭЛЕКТРОННОЙ ПОЧТЫ S/MIME

Протокол Secure Multipurpose Internet Mail Extensions (S/MIME) предназначен для защиты данных, передаваемых в формате MIME, в основном – электронной почты.

Протокол S/MIME предоставляет следующие криптографические услуги безопасности (криптографические сервисы):

- проверка целостности сообщения;
- установление подлинности отправителя (аутентификация);
- обеспечение секретности передаваемых данных (шифрование).

Протокол позволяет обычным почтовым клиентам защищать исходящую почту и интерпретировать криптографические сервисы, добавленные во входящую почту (расшифровывать сообщения, проверять их целостность и т. д.).

Стандарт определяет использование симметричных криптоалгоритмов для шифрования содержимого почтовых сообщений и алгоритма с открытым ключом для защиты передаваемого вместе с письмом ключа симметричного шифрования.

Протокол S/MIME позволяет использовать различные криптоалгоритмы, причем их список может расширяться. Изначально, из симметричных шифров могли использоваться RC2, DES или TripleDES.

Для формирования дайджестов – алгоритмы MD5 и SHA1. Защита симметричного ключа шифрования и ЭЦП в версии 2 осуществляется с помощью алгоритма RSA с ключом от 512 до 1024 бит. Версия 3 добавляет

возможность использовать другие алгоритмы, например алгоритм Диффи-Хеллмана с ключом длиной до 2048 бит. Распределение и аутентификация открытых ключей производится с помощью цифровых сертификатов формата X.509.

Таким образом, чтобы защищать переписку с помощью этого протокола, оба абонента должны сгенерировать ключевые пары и удостоверить открытые ключи с помощью сертификатов.

Альтернативой S/MIME является PGP. Данная программа положила основу работе над стандартом OpenPGP. По функциональности S/MIME и PGP во многом сходны. [9]

3.2. ПРОТОКОЛЫ SSL И TLS

Протокол Secure Sockets Layer (SSL) был разработан для обеспечения аутентификации, целостности и секретности трафика на сеансовом уровне модели OSI (с точки зрения четырехуровневой модели стека протоколов TCP/IP – на прикладном уровне).

С точки зрения выполняемых действий, различия между этими протоколами SSL и TLS весьма невелики, в то же время, они несовместимы друг с другом.

SSL обеспечивает защищенное соединение, которое могут использовать протоколы более высокого уровня – HTTP, FTP, SMTP и т. д. Наиболее широко он используется для защиты данных передаваемых по HTTP (режим HTTPS). Для этого должны использоваться SSL-совместимые web-сервер и браузер.

Протокол предусматривает два этапа взаимодействия клиента и сервера:

1) установление SSL-сессии (процедура «рукопожатия») на этом этапе может производиться аутентификация сторон соединения, распределение ключей сессии, определяются настраиваемые параметры соединения;

2) защищенное взаимодействие.

Протоколом SSL используются следующие криптоалгоритмы:

- асимметричные алгоритмы RSA и Диффи-Хеллмана;
- алгоритмы вычисления хэш-функций MD5 и SHA1;
- алгоритмы симметричного шифрования RC2,RC4, DES, TripleDES, IDEA.

В протоколе SSL v 3.0 и TLS перечень поддерживаемых алгоритмов является расширяемым. Для подтверждения подлинности открытых ключей используются цифровые сертификаты формата X.509.

Протокол SSL позволяет проводить следующие варианты аутентификации сторон взаимодействия:

- аутентификация сервера без аутентификации клиента (односторонняя аутентификация) – это наиболее часто используемый режим, позволяющий установить подлинность сервера, но не проводящий проверки клиента (ведь подобная проверка требует и от клиента наличия сертификата);

- взаимная аутентификация сторон (проверяется подлинность пкак клиента, так и сервера);

- отказ от аутентификации – полная анонимность; в данном случае SSL обеспечивает шифрование канала и проверку целостности, но не может защитить от атаки путем подмены участников взаимодействия.

Протоколы SSL и TLS получили широкое распространение, прежде всего благодаря их использованию для защиты трафика, передаваемого по протоколу HTTP в сети Интернет. В тоже время, предоставляемые SSL услуги не являются прозрачными для приложений, т. е. сетевые приложения, которые хотят воспользоваться возможностями SSL должны включать в себя реализацию протокола (или подключать ее в виде каких-то внешних модулей). [5]

3.3. ПРОТОКОЛЫ IPSEC И РАСПРЕДЕЛЕНИЕ КЛЮЧЕЙ

Протокол IPSec или, если точнее, набор протоколов, разработан как базовый протокол обеспечения безопасности на уровне IP-соединения. Он является дополнением к протоколу IP ver.4 и составной частью IP ver.6. Возможности, предоставляемые протоколами IPSec:

- контроль доступа;
- контроль целостности данных;
- аутентификация данных;
- защита от повторений;
- обеспечение конфиденциальности.

Основная задача IPSec – создание между двумя компьютерами, связанными через общедоступную (небезопасную) IP-сеть, безопасного туннеля по которому передаются конфиденциальные или чувствительные к несанкционированному изменению данные.

Подобный туннель создается с использованием криптографических методов защиты информации. Протокол работает на сетевом уровне модели OSI и, соответственно, он «прозрачен» для приложений.

Архитектура IPSec является открытой, что позволяет использовать для защиты передаваемых данных новые криптографические алгоритмы, например, соответствующие национальным стандартам.

Два протокола, входящие в состав IPSec это:

1) протокол аутентифицирующего заголовка – AH (от англ. «Authentication Header»), обеспечивающий проверку целостности и аутентификацию передаваемых данных;

2) протокол инкапсулирующей защиты данных – ESP (от англ. «Encapsulating Security Payload»), обеспечивающий конфиденциальность и, опционально, проверку целостности и аутентификацию;

Оба эти протокола имеют два режима работы – транспортный и туннельный, последний определен в качестве основного. Туннельный режим используется, если хотя бы один из соединяющихся узлов является шлюзом безопасности. В этом случае создается новый IP- заголовок, а исходный IP-пакет полностью инкапсулируется в новый.

Транспортный режим ориентирован на соединение хост-хост.

При использовании ESP в транспортном режиме защищаются только данные IP-пакета, заголовок не затрагивается. При использовании AH защита распространяется на данные и часть полей заголовка.[5]

3.3.3. Протокол SKIP

Протокол SKIP (Simple Key management for Internet Protocol) был как IP-совместимый протокол, обеспечивающий управление ключами и криптозащиту передаваемых данных на сетевом уровне модели OSI [4]. Первоначально SKIP выглядел следующим образом. Устанавливающие защищенное взаимодействие абоненты должны иметь аутентифицированные открытые ключи. По алгоритму Диффи-Хеллмана они вычисляют общий секретный ключ K_{ij}. Он используется для защиты ключевой информации.

Впоследствии, протокол SKIP был усовершенствован. В частности, были внесены изменения, позволяющие использовать SKIP совместно с ESP. Тогда SKIP отвечает за передачу ключевой информации и описание параметров соединения, а ESP решает задачи криптографической защиты данных. [1]

3.3.4. Протоколы ISAKMP и IKE

Протокол ISAKMP (Internet Security Association Key Management Protocol – протокол управления ключами и контекстами безопасности в Internet) был разработан IETF для решения задач согласования параметров и управления ключами при формировании защищенного канала. ISAKMP описывает общую схему взаимодействия, но не содержит конкретных криптоалгоритмов распределения ключей.

Поэтому он используется совместно с протоколом OAKLEY, основанном на алгоритме Диффи-Хеллмана [4]. Протокол IKE (англ.

«Internet Key Exchange» – обмен ключами в Internet) определяет совместное использование протоколов ISAKMP с OAKLEY и SKEMI (англ. «Secure Key Exchange Mechanism for Internet» – безопасный механизм обмена ключами в Интернет) для решения данной задачи. Сравнивая протоколы SKIP и IKE, надо отметить, что последний более сложен в реализации, но считается более надежным и гибким. [5]

3.4. МЕЖСЕТЕВЫЕ ЭКРАНЫ

Межсетевой экран (МЭ) – это средство защиты информации, осуществляющее анализ и фильтрацию проходящих через него сетевых пакетов. В зависимости от установленных правил, МЭ пропускает или уничтожает пакеты, разрешая или запрещая таким образом сетевые соединения. МЭ является классическим средством защиты периметра компьютерной сети: он устанавливается на границе между внутренней (защищаемой) и внешней (потенциально опасной) сетями и контролирует соединения между узлами этих сетей.

Фильтрация производится на основании правил. Наиболее безопасным при формировании правил для МЭ считается подход «запрещено все, что явно не разрешено». В этом случае, сетевой пакет проверяется на соответствие разрешающим правилам, а если таковых не найдется – отбрасывается. Но в некоторых случаях применяется и обратный принцип: «разрешено все, что явно не запрещено». Тогда проверка производится на соответствие запрещающим правилам и, если таких не будет найдено, пакет будет пропущен.

Фильтрацию можно производить на разных уровнях эталонной модели сетевого взаимодействия OSI. По этому признаку МЭ делятся на следующие классы [4]:

- экранирующий маршрутизатор;
- экранирующий транспорт (шлюз сеансового уровня);
- экранирующий шлюз (шлюз прикладного уровня).

Экранирующий маршрутизатор (или пакетный фильтр) функционирует на сетевом уровне модели OSI, но для выполнения проверок может использовать информацию и из заголовков протоколов транспортного уровня. Соответственно, фильтрация может производиться по ip-адресам отправителя и получателя, а также по TCP и UDP портам. Такие МЭ отличает высокая производительность и относительная простота – функциональностью пакетных фильтров обладают сейчас даже наиболее простые и недорогие аппаратные маршрутизаторы. В то же время, они не защищают от многих атак, например, связанных с подменой участников соединений.

Шлюз сеансового уровня работает на сеансовом уровне модели OSI и также может контролировать информацию сетевого и транспортного уровней. В дополнение к ранее перечисленным, появляются такие возможности, как аутентификация пользователей, анализ команд протоколов прикладного уровня, проверка передаваемых данных (на наличие компьютерных вирусов, соответствие политике безопасности) и т. д. [8]

4. КРИПТОАНАЛИЗ

4.1. Понятие криптоанализа

Криптоанализом называют науку восстановления (дешифрования) открытого текста без доступа к ключу. Задача криптоанализа состоит в том, чтобы определить вероятность взлома шифра и, таким образом, оценить его применимость в той или иной области.

Попытка криптоанализа называется атакой. Криптоанализ ставит своей задачей в разных условиях получить дополнительные сведения о ключе шифрования, чтобы значительно уменьшить диапазон вероятных ключей. Результаты криптоанализа могут варьироваться по степени практической применимости. Так, криптограф Ларс Кнудсен предлагает следующую классификацию успешных исходов криптоанализа блочных шифров в зависимости от объема и качества секретной информации, которую удалось получить:

- Полный взлом – криптоаналитик извлекает секретный ключ.
- Глобальная дедукция – криптоаналитик разрабатывает функциональный эквивалент исследуемого алгоритма, позволяющий зашифровывать и расшифровывать информацию без знания ключа.
- Частичная дедукция – криптоаналитику удается расшифровать или зашифровать некоторые сообщения.
- Информационная дедукция – криптоаналитик получает некоторую информацию об открытом тексте или ключе.

Появление новых криптографических алгоритмов приводит к разработке методов их взлома. Если целью криптоаналитика является раскрытие возможно большего числа шифров (независимо от того, хочет ли он этим нанести ущерб обществу, предупредить его о возможной опасности или просто получить известность), то для него наилучшей стратегией является разработка *универсальных методов анализа*. На рис.38 методы криптоанализа систематизированы по хронологии их появления и применимости для взлома различных категорий криптосистем. Горизонтальная ось разделена на временные промежутки: в область "вчера" попали атаки, которые успешно применялись для взлома шифров в прошлом; "сегодня" - методы криптоанализа, представляющие угрозу для

широко используемых в настоящее время криптосистем; "завтра" - эффективно применяемые уже сегодня методы, значение которых в будущем может возрасти, а также методы, которые пока не оказали серьезного влияния на криптологию, однако со временем могут привести прорывам во взломе шифров. На вертикальной оси обозначены области применения методов криптоанализа: для взлома криптосистем с секретным ключом, открытым ключом или хеш-функций.[8]

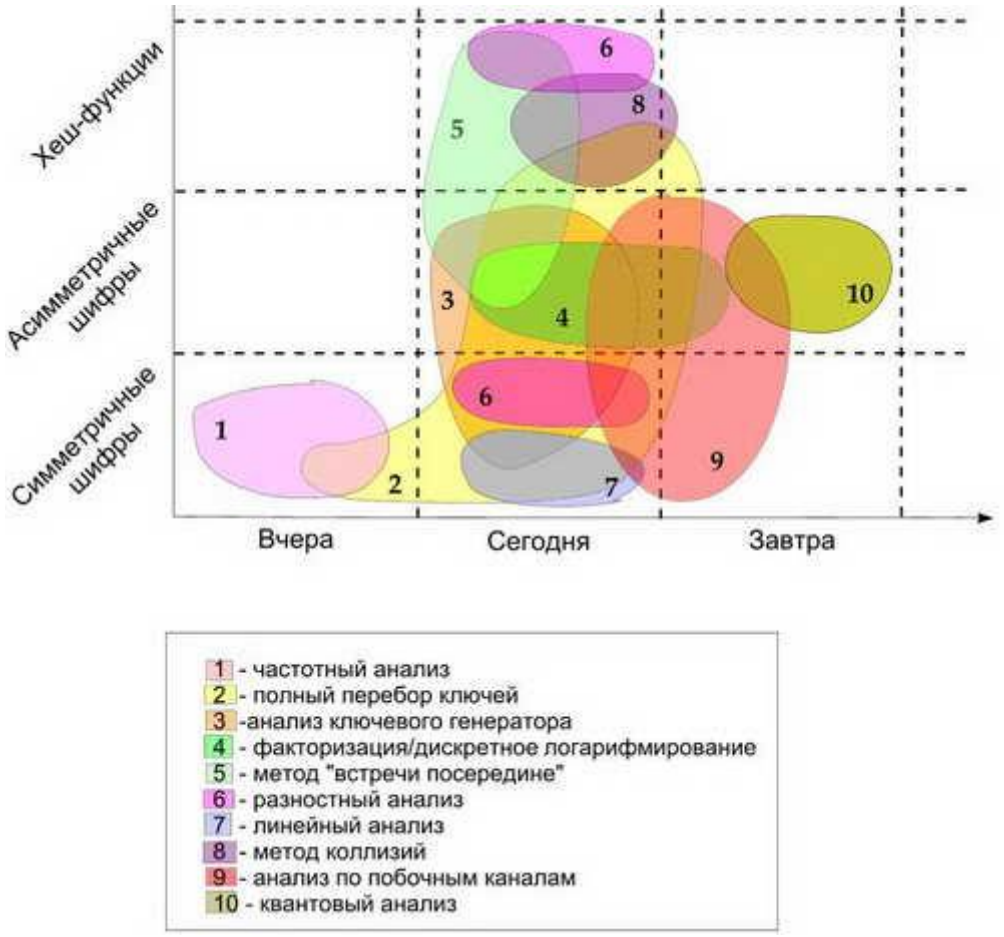

Рис. 38. Методы криптоанализа

Прежде чем перейти к рассмотрению изображенных на диаграмме типов криптоаналитических атак, введем ряд понятий и обозначений, которые потребуются нам в дальнейшем: открытый текст будем обозначать буквой x, шифртекст – буквой y (в качестве x может выступать любая последовательность битов: текстовый файл, оцифрованный звук, точечный рисунок и т.д.). Пусть для зашифрования и расшифрования используются ключи k и k` соответственно (в симметричной криптографии k=k`); обозначим функцию зашифрования E_k, расшифрования – $D_{k`}$. Тогда выполняются соотношения $E_k(x)=y$, $D_{k`}=x$.

4.2. Частотный анализ

На протяжении веков дешифрованию криптограмм помогает частотный анализ появления отдельных символов и их сочетаний. Вероятности появления отдельных букв, а также их порядок в словах и фразах естественного языка подчиняются задокументированным статистическим закономерностям: например, пара стоящих рядом букв "ся" в русском языке более вероятна, чем "цы", а "оь" не встречается никогда. Анализируя достаточно длинный текст, зашифрованный методом замены, можно по частотам появления символов произвести обратную замену и восстановить исходный текст.

Частотный метод, в котором по распределению символов в шифртексте выдвигаются гипотезы о ключе шифрования, породил требование равномерного распределения символов в шифртексте. Кроме того, принципы частотного анализа сегодня широко применяются в программах по поиску паролей и позволяют сократить перебор в десятки и сотни раз. [10]

4.3. Метод полного перебора

С появлением высокопроизводительной вычислительной техники у криптоаналитиков появилась возможность вскрывать шифры методом перебора ключей.

При осуществлении попытки атаки на основе только шифртекста криптоаналитику требуется анализировать выходные данные алгоритма и проверять их "осмысленность". В случае, когда в качестве объекта шифрования выступает графический файл или программа, задача определения "осмысленности" выходных данных становится очень трудной. Если известно, что открытый текст представляет собой предложение на естественном языке, проанализировать результат и опознать успешный исход дешифрования сравнительно несложно, тем более что криптоаналитик зачастую располагает некоторой априорной информацией о содержании сообщения. Задачу выделения осмысленного текста, т.е. определения факта правильной дешифрации, решают при помощи ЭВМ с использованием теоретических положений – *цепей Маркова*.

Атаки с использованием известного или подобранного открытого текста встречаются чаще, чем можно подумать. В среде криптоаналитиков нельзя назвать неслыханными факты добычи открытого текста шифрованного сообщения или подкупа лица, которое должно будет зашифровать избранное сообщение. Предположим, злоумышленнику

известна одна или несколько пар (x,y). Пусть для простоты для любой пары (x,y)существует единственный ключ k, удовлетворяющий соотношению $E_k(x)=y$. Примем проверку одного варианта ключа k из K за одну операцию. Тогда полный перебор ключей потребует N операций, где N - число элементов в множестве. Если в качестве оценки трудоемкости метода взять математическое ожидание случайной величины, соответствующей числу опробований до момента обнаружения использованного ключа, то мы получим N/2 операций. Кроме того, алгоритм полного перебора допускает распараллеливание, что позволяет значительно ускорить нахождение ключа. [8]

4.4. Оценка предельных мощностей взлома

Можно подумать, что с ростом мощности компьютеров разрядность ключа, достаточная для обеспечения безопасности информации против атаки методом полного перебора, будет неограниченно расти. Однако это не так. Существуют фундаментальные ограничения вычислительной мощности, наложенные структурой вселенной: например, скорость передачи любого сигнала не может превышать скорость распространения света в вакууме, а количество атомов во Вселенной (из которых, в конечном счете, состоят компьютеры) огромно, но конечно. Так, например, в описаны два фундаментальных ограничения:

1. Предел, основанный на выделяемой Солнцем энергии. Все вычисления потребляют энергию. Мощность излучения Солнца составляет приблизительно $3{,}86*10^{26}$ Вт; таким образом, за весь свой предполагаемый период существования – 10 млрд. лет, или $3*10^{17}$ секунд – Солнце выделит около 10^{44} Дж). Предположим, температура окружающей среды $T=10^{-6}$ градусов, тогда энергозатраты на одну операцию составляют $1{,}4*10^{-29}$ Дж . Значит, количество вычислительных операций, которые можно осуществить с использованием всей выделяемой солнцем энергии, равно выделяемой мощности, поделенной на количество энергии, необходимой для осуществления одной операции, т.е. всего 10^{73} операций. Такое количество операций потребовалось бы на взлом ключа из 73 десятичных цифр (или около 250 бит) методом прямого перебора при грубом предположении, что для проверки одного значения ключа необходима всего одна операция (на самом деле - сотни операций). Для справки, количество атомов солнечной системы - около 10^{60}.
2. Предел, основанный на массе Земли. Масса Земли составляет порядка $6*10^{24}$ кг. Масса протона – $1{,}6*10^{-27}$, т.е. Земля содержит приблизительно $4*10^{51}$ протонов. Сопоставим каждому протону отдельный компьютер и примем за скорость выполнения операции на таком компьютере время, за которое луч света проходит расстояние,

равное диаметру этого протона. Таким образом, каждый компьютер может выполнять $3*10^{25}$ операций в секунду. Если все эти компьютеры будут работать параллельно, их суммарное быстродействие составит $4*10^{51}*3*10^{25}$ операций в секунду, т.е. 10^{77}. Возраст Вселенной приблизительно 10 млрд. лет, т.е $3*10^{17}$. секунд. За весь период существования Вселенной такие гипотетические компьютеры размером с протон смогли бы выполнить $3*10^{94}$ операций. При описанных в п. 1 предположений относительно количества операций, необходимых на проверку значения ключа, такое количество операций позволит взломать ключ длиной 95 десятичных цифр, или 320 бит.

Таким образом, минимальный размер ключа, необходимый для защиты информации от атак злоумышленника, будет расти по мере повышения быстродействия компьютеров; тем не менее, приведенные выше вычисления показывают, что существует возможность выбрать такую длину ключа, что атаку методом полного перебора будет осуществить в принципе невозможно, вне зависимости от повышения вычислительной мощности компьютеров или успехов в области классической теории алгоритмов. [2]

4.5. Атака по ключам

Одной из причин ненадежности криптосистем является использование слабых ключей. Фундаментальное допущение криптоанализа, впервые сформулированное О. Кирхгоффом, состоит в том, что секретность сообщения всецело зависит от ключа, т.е. весь механизм шифрования, кроме значения ключа, известен противнику (секретность алгоритма не является большим препятствием: для определения типа программно реализованного криптографического алгоритма требуется лишь несколько дней инженерного анализа исполняемого кода). Слабый ключ - это ключ, не обеспечивающий достаточного уровня защиты или использующий в шифровании закономерности, которые могут быть взломаны. Алгоритм шифрования не должен иметь слабых ключей. Если это невозможно, то количество слабых ключей должно быть минимальным, чтобы уменьшить вероятность случайного выбора одного из них; все слабые ключи должны быть известны заранее, чтобы их можно было отбраковать в процессе создания ключа.

Генераторы случайных чисел – еще один источник угрозы для стойкости криптосистемы. Если для генерации ключей используется криптографический слабый алгоритм, независимо от используемого шифра вся система будет нестойкой. Качественный ключ, предназначенный для использования в рамках симметричной криптосистемы, представляет собой случайный двоичный набор. Если

требуется ключ разрядностью n, в процессе его генерации с одинаковой вероятностью должен получаться любой из 2^n возможных вариантов. Исследования компании Counterpane, показали, что определённые генераторы случайных чисел могут быть надёжными при использовании с одной целью, но ненадёжными для другой; обобщение анализа надёжности опасно. [8]

4.6. Метод "встречи посередине"

Известно, что если считать, что дни рождения распределены равномерно, то в группе из 23 человек с вероятностью 0,5 у двух человек дни рождения совпадут. В общем виде парадокс дней рождения формулируется так: если $a\sqrt{b}$ предметов выбираются с возвращением из некоторой совокупности размером b, то вероятность того, что два из них совпадут, равна $1 - e^{-\frac{a^2}{2}}$ (в описанном частном случае b=365 - количество дней в году, $a\sqrt{b} = 23$, т.е. a≈1,204).

Данный метод криптоанализа основан на "парадоксе дней рождения". Пусть нам нужно найти ключ k по известному открытому тексту xи криптограмме y. Если множество ключей криптоалгоритма замкнуто относительно композиции, т.е. для любых ключей k` и k`` найдется ключ k такой, что результат шифрования любого текста последовательно на k` и k`` равен результату шифрования этого же текста на k, т.е.$E_{K''}(E_{K'}, x) = E_k(x)$, то можно воспользоваться этим свойством. Поиск ключа k сведем к поиску эквивалентной ему пару ключей k` и k``. Для текста x построим базу данных, содержащую случайное множество ключей k`и соответствующих криптограмм $w = E_{K'}(x)$, и упорядочим ее по криптограммам w. Объем базы данных выбираем $O(\sqrt{|k'|})$, где | { k`}| – мощность множества ключей k`. Затем подбираем случайным образом ключи k`` для расшифровки текстов уи результат расшифрования $v = E_{K''}(y)$, сравниваем с базой данных. Если текст v окажется равным одной из криптограмм w, то ключ k`k``. эквивалентен искомому ключу k.

Алгоритм является вероятностным. Существуют детерминированный аналог этого алгоритма "giant step – baby step" с такой же сложностью, предложенный американским математиком Д. Шенксом. [2]

4.7. Криптоанализ симметричных шифров

Наибольший прогресс в разработке методов раскрытия блочных шифров был достигнут в самом конце XX века и связан с появлением двух

методов – разностного (дифференциального) криптоанализа и линейного криптоанализа.

Метод разностного анализа сочетает в себе обобщение идеи общей линейной структуры с применением вероятностно-статистических методов исследования. Этот метод относится к атакам по выбранному открытому тексту. Разностный анализ основан на использовании неравновероятности в распределении значений разности двух шифртекстов, полученных из пары открытых текстов, имеющих некоторую фиксированную разность. Отметим, что разностный анализ применим и для взлома хеш-функций.

Подобно разностному анализу, линейный криптоанализ является комбинированным методом, сочетающим в себе поиск линейных статаналогов для уравнений шифрования, статистический анализ имеющихся открытых и шифрованных текстов, использующий также методы согласования и перебора. Этот метод исследует статистические линейные соотношения между отдельными координатами векторов открытого текста, соответствующего шифртекста и ключа, и использует эти соотношения для определения статистическими методами отдельных координат ключевого вектора.

На сегодняшний день метод линейного криптоанализа позволил получить наиболее сильные результаты по раскрытию ряда итерационных систем блочного шифрования, в том числе и системы DES. В 1992 г. М. Мацуи формализовал этот подход, а позже успешно применил его к анализу криптоалгоритма DES. В 2001 г. в США на смену DES и Triple DES пришел новые стандарт AES, действующий и по сей день. [8]

4.8. Криптоанализ асимметричных шифров

Практически все используемые алгоритмы асимметричной криптографии основаны на задачах факторизации (например, известная криптосистема RSA) и дискретного логарифмирования в различных алгебраических структурах (схема электронно-цифровой подписи Эль-Гамаля). Для криптоанализа асимметричных криптосистем можно применять универсальные методы – например, метод "встречи посередине". Другой подход заключается в решении математической задачи, положенной в основу асимметричного шифра. Задача дискретного логарифмирования считается более сложной, чем задача факторизации. Если будет найден полиномиальный алгоритм ее решения, станет возможным и разложение на множители (обратное не доказано).

Последние достижения теории вычислительной сложности показали, что общая проблема логарифмирования в конечных полях уже не является достаточно прочным фундаментом. Наиболее эффективные на сегодняшний день алгоритмы дискретного логарифмирования имеют уже

не экспоненциальную, а субэкспоненциальную временную сложность. Это алгоритмы "index-calculus", использующие факторную базу. Ряд успешных атак на системы, основанные на сложности дискретного логарифмирования в конечных полях, привел к тому, что стандарты ЭЦП России и США, которые были приняты в 1994 г. и базировались на схеме Эль-Гамаля, в 2001 году были обновлены: переведены на эллиптические кривые. Схемы ЭЦП при этом остались прежними, но в качестве чисел, которыми они оперируют, теперь используются не элементы конечного поля $GF(2^n)$ или GF(p), а эллиптические числа - решения уравнения эллиптических кривых над указанными конечными полями. Алгоритмов, выполняющих дискретной логарифмирование на эллиптических кривых в общем случае хотя бы с субэкспоненциальной сложностью, на сегодняшний день не существует, хотя работы в этом направлении ведутся. [8]

4.9. Криптоанализ хеш-функций

Основная атака на хеш – это метод коллизий. Пусть M и M`– сообщения, H – хеш-функция, а ЭЦП представляет собой некоторую функцию S от хеша сообщения:C=S(H(M)). Законный обладатель пары "открытый ключ – секретный ключ " готов подписать сообщение M, но злоумышленник заинтересован в получении подписи под сообщением M`. Если M` выбрано так, что H(M)=H(M`), то злоумышленник может предъявить пару (M`,C): тогда атака удалась. Реализовать подбор такого сообщения можно методом, который основан на упомянутом выше "парадоксе дней рождения". Варьируя интервалы, шрифты, формат и т.п., злоумышленник получает n пар вариантов M и M` без изменения их смысла. Сообщения $M_1,\ldots M_n$ отличаются слабо, а их хеш-функции – значительно, т.е. можно считать, что значения хеш-функций выбираются случайно, равновероятно и независимо друг от друга. Тогда при $n = t\sqrt{N}$ (t>0 – некоторая константа, N – мощность множества всевозможных хеш-функций) вероятность того, что имеется пара сообщений $\widetilde{M}$ и $\widetilde{M}'$, для которых $H(\widetilde{M}) = H(\widetilde{M}')$, вычисляется по формуле $1 - e^{-\frac{t^2}{2}}$.

Этот метод криптоанализа породил требования устойчивости к коллизиям для хеш-функций. [2]

4.10. Криптоанализ по побочным каналам

Атаки по сторонним, или побочным, каналам используют информацию, которая может быть получена с устройства шифрования и не является при этом ни открытым текстом, ни шифртекстом. Такие атаки

основаны на корреляции между значениями физических параметров, измеряемых в разные моменты во время вычислений, и внутренним состоянием вычислительного устройства, имеющим отношение к секретному ключу. Этот подход менее обобщённый, но зачастую более мощный, чем классический криптоанализ.

В последние годы количество криптографических атак, использующих слабости в реализации и размещении механизмов криптоалгоритма, резко возросло. Противник может замерять время, затрачиваемое на выполнение криптографической операции, или анализировать поведение криптографического устройства при возникновении определённых ошибок вычисления. Другой подход предполагает отслеживание энергии, потребляемой смарт-картой в процессе выполнения операций с секретным ключом (например, расшифрования или генерации подписи). Побочную информацию собрать порой несложно - на сегодняшний день выделено более десяти побочных каналов, в т.ч. электромагнитное излучение, ошибки в канале связи, кэш-память и световое излучение. [8]

4.11. Нанотехнологии в криптоанализе

С помощью квантового компьютера можно проводить вычисления, не реализуемые на сегодняшних (классических) компьютерах. В 1994 году П. Шор открыл так называемый "ограниченно-вероятностный" алгоритм факторизации, который позволяет разложить на множители число N за полиномиальное от размерности задачи время $O((\log N)^3)$. Алгоритм Шора разложения чисел на множители явился главным достижением в области квантовых вычислительных алгоритмов. Это был не только крупный успех математики. Именно с этого момента началось усиленное финансирование работ по созданию квантовых компьютеров.

Важно отметить, что алгоритм Шора чрезвычайно прост и довольствуется гораздо более скромным аппаратным обеспечением, чем то, которое понадобилось бы для универсального квантового компьютера. Поэтому вероятно, что квантовое устройство для разложения на множители будет построено задолго до того, как весь диапазон квантовых вычислений станет технологически осуществимым. На сегодняшний день есть конкретные результаты. Так, IBM продемонстрировала использование созданного в лабораториях компании семикубитового квантового компьютера для факторизации чисел по алгоритму Шора. Хотя решённая им задача вряд ли способна поразить воображение (компьютер верно определил, что делителями числа 15 являются числа 5 и 3), это самое сложное вычисление в области теории чисел за всю историю квантовых компьютеров. [8]

5. Антивирусная защита

Компьютерный вирус – это специально написанная программа, которая может "приписывать" себя к другим программам, т.е. "заражать их", с целью выполнения различных нежелательных действий на компьютере, в вычислительной или информационной системе и в сети.

Когда такая программа начинает работу, то сначала, как правило, управление получает вирус. Вирус может действовать самостоятельно, выполняя определенные вредоносные действия (изменяет файлы или таблицу размещения файлов на диске, засоряет оперативную память, изменяет адресацию обращений к внешним устройствам, генерирует вредоносное приложение, крадет пароли и данные и т.д.), или "заражает" другие программы. Зараженные программы могут быть перенесены на другой компьютер с помощью дискет или локальной сети.

Формы организации вирусных атак весьма разнообразны, но в целом практически их можно "разбросать" по следующим категориям:

- удаленное проникновение в компьютер – программы, которые получают неавторизованный доступ к другому компьютеру через Internet (или локальную сеть);
- локальное проникновение в компьютер – программы, которые получают неавторизованный доступ к компьютеру, на котором они впоследствии работают;
- удаленное блокирование компьютера – программы, которые через Internet (или сеть) блокируют работу всего удаленного компьютера или отдельной программы на нем;
- локальное блокирование компьютера – программы, которые блокируют работу компьютера, на котором они работают;
- сетевые сканеры – программы, которые осуществляют сбор информации о сети, чтобы определить, какие из компьютеров и программ, работающих на них, потенциально уязвимы к атакам;
- сканеры уязвимых мест программ – программы, проверяют большие группы компьютеров в Интернет в поисках компьютеров, уязвимых к тому или иному конкретному виду атаки;
- "вскрыватели" паролей – программы, которые обнаруживают легко угадываемые пароли в зашифрованных файлах паролей;
- сетевые анализаторы (sniffers) – программы, которые слушают сетевой трафик; часто в них имеются возможности автоматического выделения имен пользователей, паролей и номеров кредитных карт из трафика;
- модификация передаваемых данных или подмена информации;

- подмена доверенного объекта распределённой вычислительной сети (работа от его имени) или ложный объект распределённой ВС (РВС).
- "социальная инженерия" – несанкционированный доступ к информации иначе, чем взлом программного обеспечения. Цель – ввести в заблуждение сотрудников (сетевых или системных администраторов, пользователей, менеджеров) для получения паролей к системе или иной информации, которая поможет нарушить безопасность системы.

К вредоносному программному обеспечению относятся сетевые черви, классические файловые вирусы, троянские программы, хакерские утилиты и прочие программы, наносящие заведомый вред компьютеру, на котором они запускаются на выполнение, или другим компьютерам в сети. [5]

5.1. Сетевые черви

Основным признаком, по которому типы червей различаются между собой, является способ распространения червя – каким способом он передает свою копию на удаленные компьютеры. Другими признаками различия КЧ между собой являются способы запуска копии червя на заражаемом компьютере, методы внедрения в систему, а также полиморфизм, "стелс" и прочие характеристики, присущие и другим типам вредоносного программного обеспечения (вирусам и троянским программам).

Пример – E-mail-Worm – почтовые черви. К данной категории червей относятся те из них, которые для своего распространения используют электронную почту. При этом червь отсылает либо свою копию в виде вложения в электронное письмо, либо ссылку на свой файл, расположенный на каком-либо сетевом ресурсе (например, URL на зараженный файл, расположенный на взломанном или хакерском веб-сайте). В первом случае код червя активизируется при открытии (запуске) зараженного вложения, во втором – при открытии ссылки на зараженный файл. В обоих случаях эффект одинаков – активизируется код червя.

Для отправки зараженных сообщений почтовые черви используют различные способы. Наиболее распространены:

- прямое подключение к SMTP-серверу, используя встроенную в код червя почтовую библиотеку;
- использование сервисов MS Outlook;
- использование функций Windows MAPI.

Различные методы используются почтовыми червями для поиска почтовых адресов, на которые будут рассылаться зараженные письма. Почтовые черви:

- рассылают себя по всем адресам, обнаруженным в адресной книге MS Outlook;
- считывает адреса из адресной базы WAB;
- сканируют "подходящие" файлы на диске и выделяет в них строки, являющиеся адресами электронной почты;
- отсылают себя по всем адресам, обнаруженным в письмах в почтовом ящике (при этом некоторые почтовые черви "отвечают" на обнаруженные в ящике письма).

Многие черви используют сразу несколько из перечисленных методов. Встречаются также и другие способы поиска адресов электронной почты. Другие виды червей: IM-Worm – черви, использующие Internet-пейджеры, IRC-Worm – черви в IRC-каналах, Net-Worm – прочие сетевые черви. [5]

5.2. Классические компьютерные вирусы

К данной категории относятся программы, распространяющие свои копии по ресурсам локального компьютера с целью: последующего запуска своего кода при каких-либо действиях пользователя или дальнейшего внедрения в другие ресурсы компьютера.

В отличие от червей, вирусы не используют сетевых сервисов для проникновения на другие компьютеры. Копия вируса попадает на удалённые компьютеры только в том случае, если зараженный объект по каким-либо не зависящим от функционала вируса причинам оказывается активизированным на другом компьютере, например:

- при заражении доступных дисков вирус проник в файлы, расположенные на сетевом ресурсе;
- вирус скопировал себя на съёмный носитель или заразил файлы на нем;
- пользователь отослал электронное письмо с зараженным вложением.

Некоторые вирусы содержат в себе свойства других разновидностей вредоносного программного обеспечения, например бэкдор-процедуру или троянскую компоненту уничтожения информации на диске.

Многие табличные и графические редакторы, системы проектирования, текстовые процессоры имеют свои макроязыки (макросы) для автоматизации выполнения повторяющихся действий. Эти макроязыки часто имеют сложную структуру и развитый набор команд. Макро-вирусы являются программами на макроязыках, встроенных в такие системы обработки данных. Для своего размножения вирусы этого класса используют возможности макроязыков и при их помощи переносят себя из одного зараженного файла (документа или таблицы) в другие. [5]

5.3. Скрипт-вирусы

Следует отметить также скрипт-вирусы, являющиеся подгруппой файловых вирусов. Данные вирусы, написаны на различных скрипт-языках (VBS, JS, BAT, PHP и т.д.). Они либо заражают другие скрипт-программы (командные и служебные файлы MS Windows или Linux), либо являются частями многокомпонентных вирусов. Также, данные вирусы могут заражать файлы других форматов (например, HTML), если в них возможно выполнение скриптов.

5.4. Троянские программы

В данную категорию входят программы, осуществляющие различные несанкционированные пользователем действия: сбор информации и её передачу злоумышленнику, ее разрушение или злонамеренную модификацию, нарушение работоспособности компьютера, использование ресурсов компьютера в неблаговидных целях. Отдельные категории троянских программ наносят ущерб удаленным компьютерам и сетям, не нарушая работоспособность зараженного компьютера (например, троянские программы, разработанные для массированных DoS-атак на удалённые ресурсы сети).

Троянские программы многообразны и различаются между собой по тем действиям, которые они производят на зараженном компьютере:

- Backdoor – троянские утилиты удаленного администрирования.
- Trojan-PSW – воровство паролей.
- Trojan-AOL – семейство троянских программ, "ворующих" коды доступа к сети AOL (America Online). Выделены в особую группу по причине своей многочисленности.
- Trojan-Clicker – Internet-кликеры. Семейство троянских программ, основная функция которых – организация несанкционированных обращений к Internet-ресурсам (обычно к Web-страницам). Достигается это либо посылкой соответствующих команд браузеру, либо заменой системных файлов, в которых указаны "стандартные" адреса Internet-ресурсов (например, файл hosts в MS Windows).
- Trojan-Downloader – доставка прочих вредоносных программ.
- Trojan-Dropper – инсталляторы прочих вредоносных программ. Троянские программы этого класса написаны в целях скрытной инсталляции других программ и практически всегда используются для "подсовывания" на компьютер-жертву вирусов или других троянских программ.

- Trojan-Proxy – троянские прокси-сервера. Семейство троянских программ, скрытно осуществляющих анонимный доступ к различным интернет-ресурсам. Обычно используются для рассылки спама.
- Trojan-Spy – шпионские программы. Данные троянцы осуществляют электронный шпионаж за пользователем зараженного компьютера: вводимая с клавиатуры информация, снимки экрана, список активных приложений и действия пользователя с ними сохраняются в какой-либо файл на диске и периодически отправляются злоумышленнику. Троянские программы этого типа часто используются для кражи информации пользователей различных систем онлайновых платежей и банковских систем.
- Trojan – прочие троянские программы. В данной категории также присутствуют "многоцелевые" троянские программы, например, те из них, которые одновременно шпионят за пользователем и предоставляют proxy-сервис удаленному злоумышленнику.
- Trojan ArcBomb – "бомбы" в архивах. Представляют собой архивы, специально оформленные таким образом, чтобы вызывать нештатное поведение архиваторов при попытке разархивировать данные – зависание или существенное замедление работы компьютера или заполнение диска большим количеством "пустых" данных. Особенно опасны "архивные бомбы" для файловых и почтовых серверов, если на сервере используется какая-либо система автоматической обработки входящей информации – "архивная бомба" может просто остановить работу сервера.
- Trojan-Notifier – оповещение об успешной атаке. Троянцы данного типа предназначены для сообщения своему "хозяину" о зараженном компьютере. При этом на адрес "хозяина" отправляется информация о компьютере, например, IP-адрес компьютера, номер открытого порта, адрес электронной почты и т. п. Отсылка осуществляется различными способами: электронным письмом, специально оформленным обращением к веб-странице "хозяина", ICQ-сообщением. Данные троянские программы используются в многокомпонентных троянских наборах для извещения своего "хозяина" об успешной инсталляции троянских компонент в атакуемую систему. [5]

Библиографический список

1. А. В. Аграновский, Р. А. Хади. Практическая криптография: алгоритмы и их программирование. – М.: Солон-Пресс, 2009

2. Басалова Г.В. Основы криптографии (2-е изд.). – М.: НОУ "ИНТУИТ", 2016, 282 с.

3. Сушко С.А. АЛГОРИТМ DES. Режим доступа: http://bit.nmu.org.ua/ua/student/metod/cryptology/%D0%BB%D0%B5%D0%BA%D1%86%D0%B8%D1%8F%206.pd

4. К.С. Пан, М.Л. Цымблер Алгоритм блочного симметричного шифрования Advanced Encryption Standard (AES) Технический отчет CELLAES-01. Режим доступа: http://crypto.pp.ua›wp-content/uploads/2010/03/aes.pdf

5. Руденков Н.А., Пролетарский А.В., Смирнова Е.В., Суровов А.М. Технологии защиты информации в компьютерных сетях (2-е изд.). – М.: НОУ "Интуит", 2016 , 368 с.

6. В.И. Кияев, О.Н. Граничин. Безопасность информационных систем . – М.: НОУ "ИНТУИТ", 2016, 192 с.

7. Кучерик А.О. Методические указания по выполнению лабораторных работ по дисциплине «Защита информации» / Владим. гос. уни-т имени Александра Григорьевича и Николая Григорьевича Столетовых; А.О. Кучерик, Д.Н. Бухаров, О.А. Новикова, В.Д. Самышкин – Владимир: Изд-во ВлГУ, 2017. – 69 с.

8. Лапонина О.Р. Криптографические основы безопасности. – М.: Изд-во "Интернет-университет информационных технологий – ИНТУИТ.ру", 2004. – 320 с.

9. Нестеров С. А. Информационная безопасность и защита информации: Учеб. пособие. – СПб.: Изд-во Политехн. ун-та, 2009. – 126 с.

10. Шнайер Брюс. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. – М.: "Триумф", 2002 – 816 с. ISBN 5-89392-055-4

11. Введение в криптографию / Под общ. ред. В. В. Ященко. 4-е изд., доп. М.: МЦНМО, 2012 – 348 с.