С.А. Нестеров

ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ и защита информации

Учебное пособие

С.А. НЕСТЕРОВ

ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ И ЗАЩИТА ИНФОРМАЦИИ

Учебное пособие

для студентов высших учебных заведений, обучающихся по направлению подготовки «Системный анализ и управление»

Санкт-Петербург
Издательство Политехнического университета
2009

ББК 32.81
Н 56

Рецензенты:
Доктор технических наук, профессор Санкт-Петербургского института информатики и автоматизации РАН И. В. Котенко
Кандидат физико-математических наук, доцент Санкт-Петербургского государственного политехнического университета А. Н. Фирсов

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

Системно излагаются теоретические основы информационной безопасности и описываются практические аспекты, связанные с их реализацией. Пособие состоит из трех разделов: «Теоретические основы защиты информации», «Основы криптографии», «Защита информации в IP-сетях».

Предназначено для студентов, обучающихся по направлению 220100 – «Системный анализ и управление». Также может быть полезно широкому кругу специалистов в области информационных технологий.

Печатается по решению редакционно-издательского совета Санкт-Петербургского государственного политехнического университета при финансовой поддержке в форме гранта Президента Российской Федерации МК-9472.2006.9.

ISBN 978-5-7422-2286-6

ОГЛАВЛЕНИЕ

СПИСОК ПРИНЯТЫХ СОКРАЩЕНИЙ

ACL – Access Control List – список управления доступом
AH – Authentication Header – протокол аутентифицирующего заголовка
CA – Certification Authority – центр сертификации или удостоверяющий центр
CBC – Cipher Block Chaining – сцепление блоков шифра (режим работы шифра DES)
CFB – Cipher FeedBack – обратная связь по шифртексту (режим работы шифра DES)
CRL – Certificate Revocation List – список отозванных сертификатов
ECB – Electronic Code Book – электронная кодовая книга (режим работы шифра DES)
ESP – Encapsulating Security Payload – протокол инкапсулирующей защиты данных
ICV – Integrity Check Value – значение контроля целостности
MAC – Message Authentication Code – код аутентификации сообщений, имитовставка
OFB – Output FeedBack – обратная связь по выходу (режим работы шифра DES)
PKI – Public Key Infrastructure – инфраструктура открытых ключей
SA – Security Association – контекст защиты или ассоциация безопасности
SPI – Security Parameter Index – индекс параметров защиты
АС – автоматизированная система (обработки информации)
ИТ – информационные технологии
МЭ – межсетевой экран
НСД – несанкционированный доступ
ОО – объект оценки
ЦС – центр сертификации
ЭЦП – электронная цифровая подпись

ВВЕДЕНИЕ

Современный специалист в области информационных технологий должен обладать знаниями и навыками обеспечения информационной безопасности. Связано это с тем, что в информационных системах предприятий и организаций хранится и обрабатывается критически важная информация, нарушение конфиденциальности, целостности или доступности, которой может привести к нежелательным последствиям. Поэтому вопросам обеспечения информационной безопасности должно уделяться внимание на всех этапах разработки и эксплуатации информационных систем.

В данном пособии изложен материал учебной дисциплины «Информационная безопасность и защита информации», в ходе изучения которой, студенты получают базовые знания о теории защиты информации, методах и средствах обеспечения информационной безопасности, а также практические навыки организации защиты информационных систем. Пособие включает в себя три раздела.

В разделе 1 «Теоретические основы защиты информации» вводятся базовые понятия, связанные с обеспечением информационной безопасности, рассматриваются основные угрозы безопасности и меры противодействия им. Также делается обзор формальных моделей безопасности и современных стандартов в этой области.

Раздел 2 «Основы криптографии» включает описание основных понятий криптографии. Также изучаются наиболее распространенные алгоритмы симметричного и асимметричного шифрования, формирования дайджестов сообщений с помощью хэш-функций, процесс создания инфраструктуры открытых ключей (PKI).

В разделе 3 «Защита информации в IP-сетях» рассматриваются протоколы криптографической защиты данных, передаваемых по телекоммуникационным сетям, использующим стек протоколов TCP/IP, использование межсетевых экранов для защиты сетей.

Пособие рекомендуется для студентов, обучающихся по направлению 220100 – «Системный анализ и управление», и также может быть полезно широкому кругу специалистов в области информационных технологий.

1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

1.1. БАЗОВЫЕ ПОНЯТИЯ

Начнем изучение дисциплины с определения ряда базовых понятий.

Информация – это сведения о лицах, предметах, фактах, событиях, явлениях и процессах независимо от формы их представления. Информация может существовать в различных формах в виде совокупностей некоторых знаков (символов, сигналов и т. д.) на носителях различных типов. Она может представлять ценность для отдельных лиц или организаций.

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 [1].

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. *Система обработки информации* – совокупность технических средств и программного обеспечения, а также методов обработки информации и действий персонала, необходимых для выполнения автоматизированной обработки информации [2]. *Объект информатизации* – совокупность информационных ресурсов, средств и систем обработки информации, используемых в соответствии с заданной информационной технологией, а также средств их обеспечения, помещений или объек-

тов (зданий, сооружений, технических средств), в которых эти средства и системы установлены, или помещений и объектов, предназначенных для ведения конфиденциальных переговоров.

В зависимости от конкретных условий, может решаться задача обеспечения комплексной безопасности объекта информатизации или защиты отдельных ресурсов – информационных, программных и т. д.

Информационные ресурсы (активы) – отдельные документы и отдельные массивы документов, документы и массивы документов, содержащиеся в информационных системах (библиотеках, архивах, фондах, банках данных, информационных системах других видов).

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 несвязанные 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. Например, угроза потери информации из-за сбоя в сети электропитания реализуется, если в АС не применяются источники бесперебойного питания или средства резервного электроснабжения (это является уязвимостью).

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.

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. Здесь надо пояснить разницу между секретной и конфиденциальной информацией. В отечественной литературе «секретной» обычно называют информацию, относящуюся к разряду государственной тайны, а «конфиденциальной» – персональные данные, коммерческую тайну и т. п.

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.

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

Ряд авторов [3] дополняют приведенную классификацию, вводя

угрозу раскрытия параметров АС, включающей в себя подсистему защиты. Угроза считается реализованной, если злоумышленником в ходе нелегального исследования системы определены все ее уязвимости. Данную угрозу относят к разряду опосредованных: последствия ее реализации не причиняют какой-либо ущерб обрабатываемой информации, но дают возможность для реализации первичных (непосредственных) угроз.

Таким образом, *безопасность информации* – это состояние защищенности информации, при котором обеспечены ее конфиденциальность, доступность и целостность. А *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* – **з**ащита информации с помощью ее криптографического преобразования[1];

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

---

[1] Вопросы, связанные с криптографической защитой информации, будут более подробно рассмотрены в разделе 2.

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.

1.2. ОБЩАЯ СХЕМА ПРОЦЕССА ОБЕСПЕЧЕНИЯ БЕЗОПАСНОСТИ

Рассмотрим теперь взаимосвязь основных субъектов и объектов обеспечения безопасности, как это предлагается в международном стандарте ISO/IEC-15408 (в России он принят как ГОСТ Р ИСО/МЭК 15408-2002 [4]).

Безопасность связана с защитой активов от угроз. Разработчики стандарта отмечают, что следует рассматривать все разновидности угроз, но в сфере безопасности наибольшее внимание уделяется тем из них, которые связаны с действиями человека. Рисунок 1.1 иллюстрирует взаимосвязь между высокоуровневыми понятиями безопасности.

За сохранность активов отвечают их владельцы, для которых они имеют ценность. Существующие или предполагаемые нарушители также могут придавать значение этим активам и стремиться использовать их вопреки интересам их владельца. Действия нарушителей приводят к появлению угроз. Как уже отмечалось выше, угрозы реализуются через имеющиеся в системе уязвимости.

Владельцы активов анализируют возможные угрозы, чтобы определить, какие из них могут быть реализованы в отношении рас-

сматриваемой системы. В результате анализа определяются риски (т. е. события или ситуации, которые предполагают возможность ущерба) и проводится их анализ.

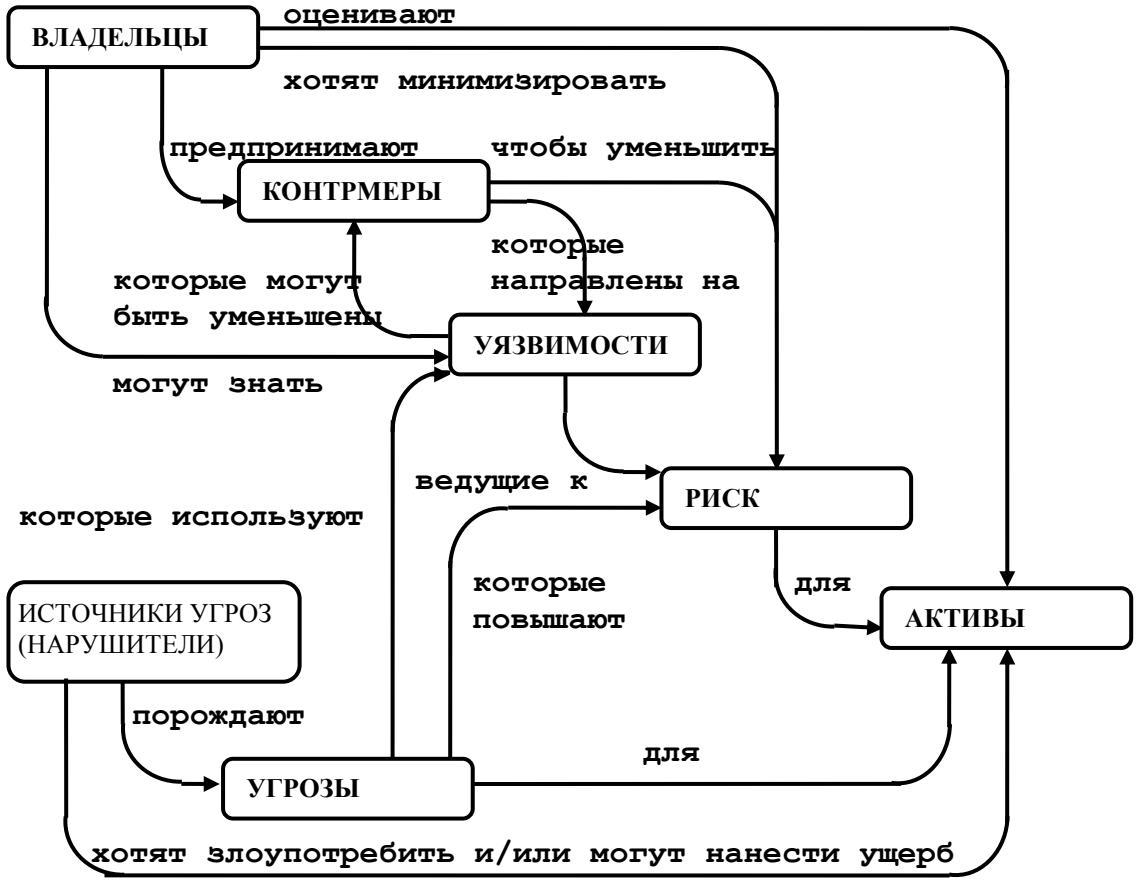


Рис. 1.1. Понятия безопасности и их взаимосвязь

Владельцы актива предпринимают контрмеры для уменьшения уязвимостей и выполнения политики безопасности. Но и после введения этих контрмер могут сохраняться остаточные уязвимости и соответственно – остаточный риск.

1.3. ИДЕНТИФИКАЦИЯ, АУТЕНТИФИКАЦИЯ, УПРАВЛЕНИЕ ДОСТУПОМ. ЗАЩИТА ОТ НЕСАНКЦИОНИРОВАННОГО ДОСТУПА

В этом разделе будут рассмотрены вопросы, связанные с защитой информации от несанкционированного доступа (НСД).

Защита информации от несанкционированного доступа – 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. Кроме идентификации пользователей, может проводиться идентификация групп пользователей, ресурсов АС и т. д. Идентификация нужна и для других системных задач, например, для ведения журналов событий. В большинстве случаев идентификация сопровождается аутентификацией. *Аутентификация* – установление подлинности – проверка принадлежности пользователю предъявленного им идентификатора. Например, в начале сеанса работы в АС пользователь вводит имя и пароль. На основании этих данных система проводит идентификацию (по имени пользователя) и аутентификацию (сопоставляя имя пользователя и введенный пароль).

Управление доступом – метод защиты информации путем регулирования использования всех ресурсов системы.

Система идентификации и аутентификации является одним из ключевых элементов инфраструктуры защиты от НСД любой инфор-

мационной системы. 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. Например, двухфактор-

ная аутентификация – пользователь предъявляет системе смарт-карту и вводит пин-код для ее активации.

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»).

1.3.1. Парольные системы аутентификации

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

Идентификатор пользователя – уникальная информация, позволяющая различить отдельных пользователей парольной системы (провести идентификацию). Это может быть имя учетной записи пользователя в системе или специально генерируемые уникальные числовые идентификаторы.

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. При прочих равных условиях, 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. Но системы с многоразовыми

паролями (в них пароль может быть использован многократно) проще реализовать и дешевле поддерживать, поэтому они более распространены.

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. В отдельных случаях подобная система может выполнять дополнительные функции, такие как генерация и распределение криптографических ключей и т. д. Как правило, парольная система включает в себя интерфейс пользователя, интерфейс администратора, базу учетных записей, модули сопряжения с другими компонентами подсистемы безопасности (подсистемой разграничения доступа, регистрации событий и т. д.).

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. Это также усложняет подбор.

3. Периодическая проверка администраторами безопасности качества используемых паролей путем имитации атак[1], таких как подбор паролей «по словарю» (т. е. проверка на использование в качестве

[1] *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.

пароля слов естественного языка и простых комбинаций символов, таких как «1234»).

4. Установление максимального и минимального сроков жизни пароля, использование механизма принудительной смены старых паролей. При внедрении данной меры надо учитывать, что при невысокой квалификации пользователей, от администратора потребуются дополнительные усилия по разъяснению пользователям того, что «от них требует система».

5. Ограничение числа неудачных попыток ввода пароля (блокирование учетной записи после заданного числа неудачных попыток войти в систему). Данная мера позволяет защититься от атак путем подбора паролей. Но при необдуманном внедрении также может привести к дополнительным проблемам – легальные пользователи из-за ошибок ввода паролей по невнимательности могут блокировать свои учетные записи, что потребует от администратора дополнительных усилий.

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

1.4. МОДЕЛИ БЕЗОПАСНОСТИ

Как уже отмечалось в разделе 1.1, важным этапом процесса обеспечения безопасности АС является разработка политики безопасности. Если отсутствует политика безопасности, невозможно даже четко провести разграничение между санкционированным (легальным) доступом к информации и НСД.

Политика безопасности может быть описана формальным или неформальным образом. Формальное описание политики безопасности производится в рамках модели безопасности. С этой точки зрения, модель безопасности можно определить как абстрактное описание

поведения целого класса систем, без рассмотрения конкретных деталей их реализации.

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

Сущность – любая именованная составляющая защищаемой АС.

Субъект – активная сущность, которая может инициировать запросы ресурсов и использовать их для выполнения каких-либо вычислительных операций. В качестве субъекта может выступать выполняющаяся в системе программа или «пользователь» (не реальный человек, а сущность АС).

Объект – пассивная сущность, используемая для хранения или получения информации. В качестве объекта может рассматриваться, например, файл с данными.

Обычно предполагается, что существует безошибочный способ различения объектов и субъектов.

Доступ – взаимодействие между субъектом и объектом, в результате которого производится 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.

В литературе выделяются три основных класса моделей политики безопасности: дискреционные, мандатные и ролевые.

Основу *дискреционной* (избирательной) политики безопасности составляет дискреционное управление доступом, которое характеризуется следующими свойствами [3]:

- 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. Она будет подробно описана ниже.

Этот тип управления доступом наиболее часто используется в операционных системах в связи с относительной простотой реализации. В этом случае, правила управления доступом часто описываются через *списки управления доступом* (англ. «Access Control List», сокр. ACL). Список связан с защищаемым объектом и хранит перечень субъектов и их разрешений на данный объект. В качестве примера можно привести использование ACL для описания прав доступа пользователей и групп к файлу в файловой системе NTFS в операционных системах семейства Windows NT.

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, которая будет рассмотрена ниже в данном разделе.

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 избирательных моделей.

1.4.1. Модель Харрисона-Рузо-Ульмана

Модель Харрисона-Рузо-Ульмана (матричная модель) используется для анализа системы защиты, реализующей дискреционную политику безопасности. При этом система представляется конечным автоматом, функционирующим согласно определенным правилам перехода.

При использовании матричной модели доступа должны быть определены множества субъектов **S**, объектов **O** и прав доступа **R**. В качестве субъектов системы рассматриваются в первую очередь выполняющиеся программы, поэтому предполагается, что $\mathbf{S} \subset \mathbf{O}$. Условия доступа субъекта $s \in \mathbf{S}$ к объекту $o \in \mathbf{O}$ определяются матрицей доступа. Пусть, например, множество прав доступа состоит из прав на чтение (r), запись (w), выполнение (e). Запрет будет соответствовать пустому множеству прав доступа ($\varnothing$). Тогда матрица доступа может быть такой, как представлено в табл. 1.1.

Таблица 1.1

Пример матрицы доступа

	o_1	o_2	o_3	o_4
s_1	rwe	∅	rw	rw
s_2	e	rwe	r	∅

Здесь мы предполагаем, что объекты o_1, o_2 – это исполняемые файлы, которые после запуска становятся субъектами s_1 и s_2.

Могут определяться и другие наборы прав, например, {чтение, запись, владение}.

При описании систем с большим числом объектов и субъектов, размерность матрицы доступа может получиться весьма значительной. Для ее снижения, одинаковые по имеющимся правам субъекты и сходные по значимости объекты можно организовать в группы и давать разрешения группе субъектов на группу объектов.

Функционирование системы рассматривается с точки зрения изменений в матрице доступа. Модель определяет 6 примитивных операций: «создать»/«уничтожить» объект и субъект, «внести»/«удалить» право доступа субъекта к объекту. Их описание приведено в табл. 1.2.

Начальное состояние системы описывается множеством прав доступа **R**, множеством субъектов **S**, множеством объектов **O** ($\mathbf{S} \subseteq \mathbf{O}$, мощности указанных множеств $|\mathbf{S}|=i$, $|\mathbf{O}|=j$, $i \le j$), матрицей доступа $\mathbf{M}_{i \times j}$ (элемент матрицы, соответствующий субъекту s и объекту o обозначается **M**[s,o] и является подмножеством множества прав доступа). Конечное состояние (после выполнения операции) – **S'**, **O'**, **M'**, **R** (множество прав доступа не изменяется).

Из примитивных операторов могут составляться команды. Команда состоит из двух частей: условия, при котором она выполняется, и последовательности операторов.

Таблица 1.2

Элементарные операции модели Харрисона-Рузо-Ульмана

Операция	Результат операции
«создать» субъект s', где s'$\notin$**S**	**S'**=**S**$\cup${s'}; **O'**=**O**$\cup${s'}; **M'**[s,o]= **M**[s,o] для всех s$\in$**S,** o$\in$**O**; **M'**[s',o]= $\varnothing$ для всех o$\in$**O'**, **M'**[s,s']= $\varnothing$ для всех s$\in$**S'**
«создать» объект o', где o'$\notin$**O**	**S'**=**S**; **O'**=**O**$\cup${o'}; **M'**[s,o]= **M**[s,o] для всех s$\in$**S,** o$\in$**O**; **M'**[s,o']= $\varnothing$ для всех s$\in$**S'**
«уничтожить» субъект s', где s'$\in$**S**	**S'**=**S**\{s'}; **O'**=**O**\{s'}; **M'**[s,o]= **M**[s,o] для всех s$\in$**S',** o$\in$**O'**;
«уничтожить» объект o', где o'$\in$**O**	**S'**=**S**; **O'**=**O**\{o'}; **M'**[s,o]= **M**[s,o] для всех s$\in$**S',** o$\in$**O'**;
«внести» право r'$\in$**R** в **M**[s',o'], где s'$\in$**S,** o'$\in$**O**	**S'**=**S**; **O'**=**O**; **M'**[s,o]=**M**[s,o] для s$\neq$s',s$\in$**S',** o$\neq$o', o$\in$**O'**; **M'**[s',o']=**M**[s',o'] $\cup${r'}
«удалить» право r'$\in$**R** из **M**[s',o'], где s'$\in$**S,** o'$\in$**O**	**S'**=**S**; **O'**=**O**; **M'**[s,o]=**M**[s,o] для s$\neq$s',s$\in$**S',** o$\neq$o', o$\in$**O'**; **M'**[s',o']=**M**[s',o']\ {r'}

Общий вид команды [3]:

command C (x_1, …, x_k):

if $r_1\in$ **M**[x_{s1},x_{o1}] and … and $r_m\in$ **M**[x_{sm},x_{om}] then

α_1;

…

α_n;

end ,

где $r_1, \ldots, r_m \in \mathbf{R}$ – права доступа, $\alpha_1, \ldots, \alpha_n$ – последовательность примитивных операторов. При выполнении команды, система переходит из состояния **Q** в новое состояние **Q'**. При этом, если хотя бы одно из условий команды не выполнено, **Q=Q'**. Для примера, рассмотрим команду создания субъектом s файла f. Множество прав доступа – чтение (read), запись (write), владение (own). Считаем, что для создания файла не требуется выполнения каких-либо дополнительных условий.

command «создать файл» (s, f)
    «создать» объект f;
    «внести» право владения own в M[s,f];
    «внести» право на чтение read в M[s,f];
    «внести» право на запись write в M[s,f];
end

Как показали результаты анализа данной модели безопасности, задача построения алгоритма проверки безопасности систем, реализующих дискреционную политику безопасности, не может быть решена в общем случае.

Введем ряд определений.

Будем считать, что возможна *утечка права* $\mathbf{r} \in \mathbf{R}$ в результате выполнения команды **c**, если при переходе системы в конечное состояние **Q'** выполняется примитивный оператор, вносящий **r** в элемент матрицы доступов **M**, до этого **r** не содержавший.

Начальное состояние $\mathbf{Q_0}$ называется *безопасным по отношению к некоторому праву* **r**, если невозможен переход системы в такое состояние **Q**, в котором может возникнуть утечка права **r**.

Система называется *монооперационной*, если каждая команда содержит только один примитивный оператор.

Для модели Харрисона-Рузо-Ульмана были доказаны следующие утверждения:

1. существует алгоритм, который проверяет, является ли исходное состояние монооперационной системы безопасным для данного права **r**;

2. задача проверки безопасности произвольных систем алгоритмически неразрешима.

Таким образом, с одной стороны, общая модель Харрисона-Рузо-Ульмана может выражать большое разнообразие политик дискреционного доступа, но при этом не существует алгоритма проверки их безопасности. С другой стороны, можно предпочесть монооперационную систему, для которой алгоритм проверки безопасности существует, но данный класс систем является слишком узким. Например, монооперационные системы не могут выразить политику, дающую субъектам права на созданные ими объекты, т.к. не существует одной операции, которая и создает объект, и одновременно помечает его как принадлежащий создающему субъекту.

1.4.2. Модель Белла-ЛаПадула

Классической мандатной моделью безопасности является модель Белла–ЛаПадула. В ней для описания системы используются:

S – множество субъектов (например, множество пользователей и программ);

O – множество объектов (например, множество файлов);

L – линейно упорядоченное множество уровней безопасности (например, «общий доступ», «для служебного пользования», «секретно», «совершенно секретно»);

F: $\mathbf{S} \cup \mathbf{O} \rightarrow \mathbf{L}$ – функция, определяющая уровень безопасности субъекта или объекта в данном состоянии;

V – множество состояний – множество упорядоченных пар (F, **M**), где **M** – матрица доступа субъектов к объектам (матрица, строки которой соответствуют субъектам системы, столбцы – объектам, элемент матрицы M_{so}, далее обозначаемый как M[s,o], описывает права на доступ субъекта *s* к объекту *o*).

Система описывается начальным состоянием $v_0 \in \mathbf{V}$, множеством запросов **R** и функцией переходов T: $(\mathbf{V} \times \mathbf{R}) \rightarrow \mathbf{V}$, описывающей переход системы из состояния в состояние под действием запроса.

В модели Белла–ЛаПадула вводится определение двух свойств безопасности системы: безопасность по чтению и безопасность по записи.

Состояние (F,**M**) *безопасно по чтению* тогда и только тогда, когда для $\forall$ s$\in$**S**, $\forall$ o$\in$**O** выполняется требование:

$$\text{чтение} \in M[s,o] \Rightarrow F(s) \geq F(o) ,$$

т. е. субъект *s* может прочитать информацию из объекта *o*, только если уровень секретности *o* меньше или равен уровню доступа *s*. Данное свойство безопасности также называется правилом запрета чтения с верхнего уровня.

Состояние (F,**M**) *безопасно по записи* тогда и только тогда, когда для $\forall$ s$\in$**S**, $\forall$ o$\in$**O** выполняется требование:

$$\text{запись} \in M[s,o] \Rightarrow F(o) \geq F(s) ,$$

т. е. субъект *s* может записать информацию в объект *o*, только если уровень секретности *o* выше или равен уровню доступа *s*. Данное свойство безопасности также называется правилом запрета записи на нижний уровень.

Состояние системы v$\in$**V** *безопасно* тогда и только тогда, когда оно безопасно и по чтению, и по записи.

Система (v_0,**R**,T) *безопасна* тогда и только тогда, когда ее начальное состояние v_0 безопасно и любое состояние, достижимое из v_0 после выполнения конечной последовательности запросов из **R**, также безопасно.

Большим достоинством модели Белла–ЛаПадула является то, что для нее доказана основная теорема безопасности. В общем случае, данная теорема формулируется следующим образом: если начальное состояние системы безопасно и все переходы из состояния в состояние не нарушают ограничений, сформулированных политикой безопасности, то любое состояние системы, достижимое за конечное чис-

ло переходов будет безопасным. В случае модели Белла-ЛаПадула, ограничения не позволяют нарушить безопасность по чтению и записи.

Основная теорема безопасности для модели Белла – ЛаПадула.

Система (v_0,**R**,T) (т. е. система с начальным состоянием v_0, множеством запросов **R**, функцией переходов T) безопасна тогда и только тогда, когда состояние v_0 безопасно и функция переходов T такова, что для $\forall$ v$\in$**V,** достижимого из состояния v_0 после выполнения конечной последовательности запросов из **R** (таких что $T(v,r)=v^*$, где v=(F,**M**) – исходное состояние, $v^*=(F^*,\mathbf{M}^*)$ – состояние после перехода), для $\forall$ s$\in$**S**, $\forall$ o$\in$**O** выполняются следующие условия:

1. если чтение$\in M^*[s,o]$ и чтение$\notin$M[s,o], то $F^*(s)\geq F^*(o)$;
2. если чтение$\in$M[s,o] и $F^*(s)<F^*(o)$, то чтение$\notin M^*[s,o]$;
3. если запись$\in M^*[s,o]$ и запись$\notin$M[s,o], то $F^*(o)\geq F^*(s)$;
4. если запись$\in$M[s,o] и $F^*(o)<F^*(s)$, то запись$\notin M^*[s,o]$.

Кратко рассмотрим доказательство теоремы.

Необходимость. Если система безопасна, то начальное состояние v_0 безопасно по определению. Пусть существует некоторое состояние v^*, достижимое из v_0 путем выполнения конечного числа запросов из **R** и полученное в результате перехода из безопасного состояния v: $T(v,r)=v^*$. Тогда, если при таком переходе нарушено хотя бы одно из первых двух ограничений, накладываемых теоремой на функцию T, то состояние v^* не будет безопасным по чтению. Если функция T нарушает одно из двух последних условий теоремы, то состояние v^* не будет безопасным по записи. Таким образом, при нарушении условий теоремы система становится небезопасной. Необходимость доказана.

Достаточность. Используем метод доказательства от противного. Пусть система небезопасна. В этом случае, либо начальное состояние v_0 небезопасно, что противоречит условиям теоремы, либо должно существовать небезопасное состояние v^*, достижимое из

безопасного начального состояния v_0 путем выполнения конечного числа запросов из **R**. В этом случае, обязательно будет иметь место переход $T(v,r)=v^*$, при котором состояние v – безопасно, а v^* - нет. Однако четыре условия теоремы делают такой переход невозможным.

Несмотря на достоинства модели Белла–ЛаПадула, при ее строгой реализации в реальных АС возникает ряд проблем.

1. *Завышение уровня секретности*, связанное с одноуровневой природой объектов и правилом безопасности по записи. Если субъект с высоким уровнем доступа хочет записать что-то в объект с низким уровнем секретности, то сначала приходится повысить уровень секретности объекта, а потом осуществлять запись. Таким образом, даже один параграф, добавленный в большой документ субъектом с высоким уровнем доступа, повышает уровень секретности всего этого документа. Если по ходу работы изменения в документ вносят субъекты со все более высоким уровнем доступа, уровень секретности документа также постоянно растет.

2. *Запись вслепую*. Эта проблема возникает, когда субъект производит операцию записи в объект с более высоким уровнем безопасности, чем его собственный. В этом случае, после завершения операции записи, субъект не сможет проверить правильность выполнения записи при помощи контрольного чтения, так как ему это запрещено в соответствии с правилом безопасности по чтению.

3. *Проблема удаленного чтения-записи*. В распределенных системах при удаленном чтении файла создаются два потока: от субъекта к объекту (запросы на чтение, подтверждения, прочая служебная информация) и от объекта к субъекту (сами запрашиваемые данные). При этом, например, если $F(s)>F(o)$, то первый поток будет противоречить свойству безопасности по записи. На практике для решения этой проблемы надо разделять служебные потоки (запросы, подтверждения) и собственно передачу информации.

4. *Доверенные субъекты*. Модель Белла–ЛаПадула не учитыва-

ет, что в реальной системе, как правило, существуют субъекты, действующие в интересах администратора, а также системные процессы, например, драйверы. Жесткое соблюдение правил запрета чтения с верхнего уровня и запрета записи на нижний уровень в ряде случаев делает невозможной работу подобных процессов. Соответственно, их также приходится выделять.

1.4.3. Ролевая модель безопасности

Ролевая модель безопасности появилась как результат развития дискреционной модели. Однако она обладает новыми по отношению к исходной модели свойствами: управление доступом в ней осуществляется как на основе определения прав доступа для ролей, так и путем сопоставления ролей пользователям и установки правил, регламентирующих использование ролей во время сеансов.

В ролевой модели понятие «субъект» замещается понятиями «пользователь» и «роль» [5]. Пользователь – человек, работающий с системой и выполняющий определенные служебные обязанности. Роль – это активно действующая в системе абстрактная сущность, с которой связан набор полномочий, необходимых для выполнения определенной деятельности. Подобное разделение хорошо отражает особенности деятельности различных организаций, что привело к распространению ролевых политик безопасности. При этом, как один пользователь может быть авторизован администратором на выполнение одной или нескольких ролей, так и одна роль может быть сопоставлена одному или нескольким пользователям.

При использовании ролевой политики, управление доступом осуществляется в две стадии:

- для каждой роли указывается набор полномочий (разрешений на доступ к различным объектам системы);
- каждому пользователю сопоставляется список доступных ему ролей.

При определении ролевой политики безопасности используются следующие множества:

U – множество пользователей;

R – множество ролей;

P – множество полномочий (разрешений) на доступ к объектам системы;

S – множество сеансов работы пользователя с системой.

Как уже отмечалось выше, ролям сопоставляются полномочия, а пользователям – роли. Это задается путем определения следующих множеств:

$PA \subseteq P \times R$ – определяет множество полномочий, установленных ролям (для наглядности условно может быть представлено в виде матрицы доступа);

$UA \subseteq U \times R$ – устанавливает соответствие между пользователями и доступными им ролями.

Рассмотрим процесс определения прав доступа для пользователя, открывшего сеанс работы с системой (в рамках одного сеанса работает только один пользователь). Правила управления доступом задаются с помощью следующих функций:

$user: S \to U$ – для каждого сеанса $s \in S$ эта функция определяет пользователя, который осуществляет этот сеанс работы с системой: $user(s) = u \mid u \in U$;

$roles$ – для каждого сеанса $s \in S$ данная функция определяет подмножество ролей, которые могут быть одновременно доступны пользователю в ходе этого сеанса: $roles(s) = \{r_i \mid (user(s), r_i) \in UA\}$;

$permissions: S \to P$ – для каждого сеанса $s \in S$ эта функция задает набор доступных в нем полномочий, который определяется путем объединения полномочий всех ролей, задействованных в этом сеансе: $permissions(s) = \bigcup_{r \in roles(s)} \{p_i \mid (p_i, r) \in PA\}$.

В качестве критерия безопасности ролевой модели используется следующее правило: система считается безопасной, если любой поль-

зователь системы, работающий в сеансе $s \in S$, может осуществлять действия, требующие полномочия $p \in P$ только в том случае, если $p \in permissions(s)$.

Существует несколько разновидностей ролевых моделей управления доступом, различающихся видом функций $user$, $roles$ и $permissions$, а также ограничениями, накладываемыми на множества PA и UA.

В частности, может определяться иерархическая организация ролей, при которой роли организуются в иерархии, и каждая роль наследует полномочия всех подчиненных ей ролей.

Могут быть определены взаимоисключающие роли (т. е. такие роли, которые не могут быть одновременно назначены одному пользователю). Также может вводиться ограничение на одновременное использование ролей в рамках одной сессии, количественные ограничения при назначении ролей и полномочий, может производиться группировка ролей и полномочий.

1.5. 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. Он был разработан на основе стандарта «Общие критерии безопасности информационных технологий» вер.2.1. 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» [4], часто называемый в литературе «Общие критерии».

Стандарт разработан таким образом, чтобы удовлетворить потребности трех групп специалистов: разработчиков, экспертов по сер-

тификации и пользователей объекта оценки. Под объектом оценки (ОО) в стандарте понимаются «подлежащие оценке продукт информационных технологий (ИТ) или система с руководствами администратора и пользователя». К таким объектам относятся, например, операционные системы, прикладные программы, информационные системы и т. д. Ранее, в разделе 1.2 пособия рассматривалась определяемая стандартом взаимосвязь высокоуровневых понятий в области информационной безопасности (рис.1.1).

«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.

Описание обоих типов требований выполнено в едином стиле: они организованы в иерархию «класс – семейство – компонент – элемент». Термин «класс» используется для наиболее общей группировки требований безопасности, а элемент – самый нижний, неделимый уровень требований безопасности. В стандарте выделены 11 классов функциональных требований:

- аудит безопасности;
- связь (передача данных);
- криптографическая поддержка (криптографическая защита);
- защита данных пользователя;
- идентификация и аутентификация;
- управление безопасностью;
- приватность (конфиденциальность);
- защита функций безопасности объекта;
- использование ресурсов;
- доступ к объекту оценки;
- доверенный маршрут/канал.

Основные структуры, определяемые «Общими критериями» – это профиль защиты и задание по безопасности. Профиль защиты – это независимая от реализации совокупность требований безопасности для некоторой категории ОО, отвечающая специфическим запросам потребителя. Профиль состоит из компонентов или пакетов функциональных требований и одного из уровней гарантированности. Структура профиля защиты представлена на рис. 1.2.

Профиль определяет «модель» системы безопасности или отдельного ее модуля. Количество профилей потенциально не ограничено, они разрабатываются для разных областей применения (например, профиль «Специализированные средства защиты от несанкционированного доступа к конфиденциальной информации»).

Профиль защиты служит основой для создания задания по безопасности, которое можно рассматривать как технический проект для разработки ОО. Задание по безопасности может включать требования одного или нескольких профилей защиты. Оно описывает также уровень функциональных возможностей средств и механизмов защиты, реализованных в ОО, и приводит обоснование степени их адекватности.

По результатам проводимых оценок, создаются каталоги сертифицированных профилей защиты и продуктов (операционных систем, средств защиты информации и т. д.), которые затем используются при оценке других объектов.

Рис. 1.2. Структура профиля защиты

2. ОСНОВЫ КРИПТОГРАФИИ

2.1. ОСНОВНЫЕ ПОНЯТИЯ. КЛАССИФИКАЦИЯ ШИФРОВ

Исторически *криптография* (в переводе с греческого – «тайнопись») зародилась как способ скрытой передачи сообщений без сокрытия самого факта их передачи [6]. 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*».

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 (Claude Shannon) «Теория связи в секретных системах», опубликованная в открытой печати в 1949 году. На рис. 2.1 изображена предложенная Шенноном схема секретной системы [7].

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.

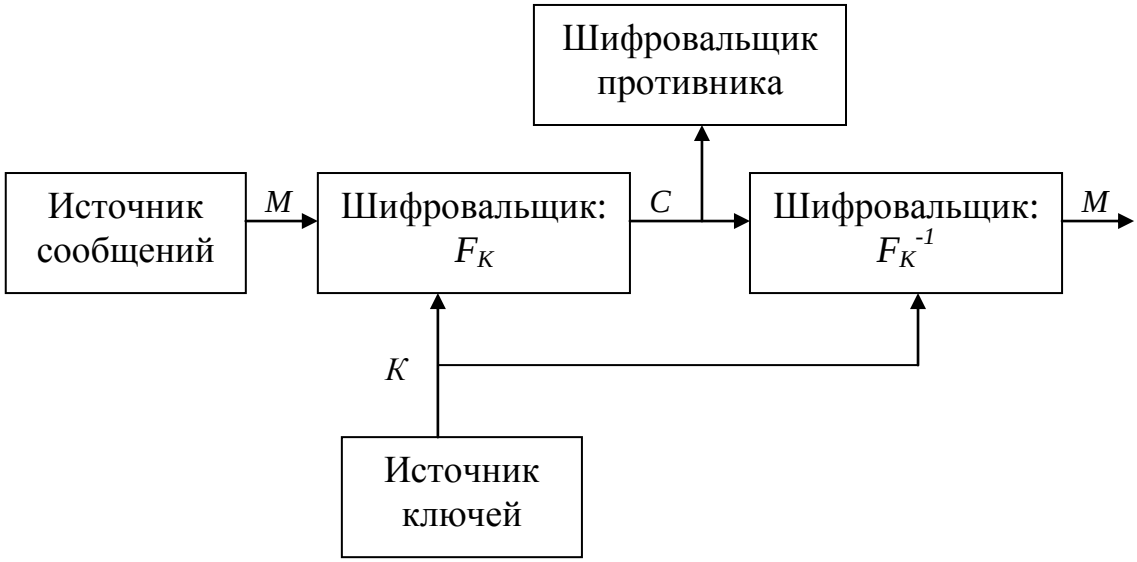


Рис. 2.1. Схема секретной системы

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

$$C = F_K M\,. \qquad (2.1)$$

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

$$M = F_K^{-1} C\,. \qquad (2.2)$$

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*. Иными словами, безусловно стойкими являются такие шифры, для которых криптоаналитик (даже если он обладает бесконечными вычислительными ресурсами) не может улучшить оценку исходного сообщения M на основе знания криптограммы C по сравнению с оценкой при неизвестной криптограмме.

Шифры другого типа характеризуются тем, что при определенном объеме перехваченных данных определить ключ (или расшифровать сообщение без знания ключа) становится теоретически возможно. Минимальный объем криптограммы, для которого существует единственное решение криптоаналитической задачи, называется *интервалом единственности***.** Однако, для криптоаналитика, обладающего ограниченными вычислительными ресурсами, вероятность найти это решение за время, в течение которого информация представляет ценность, чрезвычайно мала. 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. При соблюдении этого условия сама процедура преобразования может быть достаточно простой.

Рассмотрим следующий пример. Пусть нужно передать сообщение *M*, представленное в двоичной кодировке. Вероятность того, что очередной символ сообщения будет 1, равна *q*, 0 – (*1 - q)*. Криптограмма получается путем побитного сложения по модулю 2 (т. е. сложения без переноса старшего разряда) сообщения с бесконечным, случайным, равномерно распределенным ключом *K*:

$$C = M \oplus K . \qquad (2.3)$$

Подобное преобразование также называют *гаммированием*, а ключ *K* – *ключевой гаммой*. Найдем вероятность того, что очередной символ криптограммы будет равен 1. Это произойдет, если в исходном сообщении соответствующий символ равен 0, а в ключе – 1, или в сообщении – 1, в ключе – 0. Эти пары событий взаимоисключающие, так что следует применить формулу сложения вероятностей:

$$p(C = 1) = (1 - q) \times 0{,}5 + \mathrm{q} \times 0{,}5 = 0{,}5 . \qquad (2.4)$$

Таким образом, вероятность появления в криптограмме единицы не зависит от статистических свойств исходного сообщения. И анализируя криптограмму, нарушитель не сможет получить дополнительной информации об исходном сообщении. Надо отметить, что подобными свойствами обладает только случайный бесконечный равномерно распределенный ключ. Если вероятность появления в ключе единицы отлична от 0,5, то q в формуле (2.4) не удастся исключить из результата.

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 [8].

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 XIX веке голландским криптографом Огюстом Керкгоффсом (Auguste Kerckhoffs). В соответствии с ним, 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).

Существует несколько обобщенных постановок задачи криптоанализа. Все они формулируются в предположении, что криптоаналитику известны применяемый алгоритм шифрования и полученные криптограммы. Могут рассматриваться:

- атака при наличии только известной криптограммы;

- атака при наличии известного фрагмента открытого текста. В этом случае, криптоаналитик имеет доступ к криптограммам, а также к соответствующим некоторым из них исходным сообщениям. Зада-

ча – определить использующийся при шифровании ключ или расшифровать все остальные сообщения. Разновидность данного класса атак – атака с возможностью выбора открытого текста (когда криптоаналитик может навязать текст для шифрования и получить соответствующую ему криптограмму);

- атаки, использующие особенности реализации аппаратных шифраторов. В частности, может анализироваться тепловое и электромагнитное излучение от устройств, распространение ошибок после однократного воздействия на аппаратуру (по цепи электропитания или иным образом) и т. д.;

- атака методом полного перебора множества возможных ключей. Данная атака также называется «атака методом грубой силы» (англ. «brut force»).

2.1.1. Виды шифров

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 заключается в том, что символы шифруемого текста заменяются символами того или иного алфавита (алфавита криптограммы) в соответствии с заранее обусловленной схемой замены.

Подстановки разделяются на *одноалфавитные* и *многоалфавитные*. В первом случае, определенному символу алфавита исходного сообщения всегда ставится в соответствие один и тот же символ алфавита криптограммы. Один из наиболее известных шифров данного класса – шифр Цезаря. В нем каждая буква алфавита заменялась на

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.

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 шифртекста. Например, нечетные символы сообщения заменяются по одному правилу, четные – по другому.

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.

Шифрование *гаммированием* заключается в том, что символы шифруемого текста складываются с символами некоторой случайной последовательности, называемой гаммой шифра или ключевой гаммой. Стойкость шифрования определяется длиной (периодом) неповторяющейся части гаммы шифра, а также сложностью предугадывания следующих элементов гаммы по предыдущим.

Шифрование *аналитическими преобразованиями* подразумевает использование аналитического правила (формулы) по которому преобразуется текст.

По типу использования ключей шифры делятся на:

- *симметричные*, использующие для шифрования и расшифровывания информации один и тот же ключ;

- *асимметричные,* использующие для шифрования и расшифровывания два различных ключа. Данный тип шифров будет подробно рассматриваться в разделе 2.4.

По размеру преобразуемого блока шифры делятся на блочные и потоковые.

Блочные шифры осуществляют преобразование информации блоками фиксированной длины. Если длина шифруемого сообщения некратна размеру блока, то его добавляют до нужной длины последовательностью специального вида. Например, это может быть последовательность 100…0. После расшифровки, последний блок просматривают справа налево и отбрасывают «хвост» до первой единицы включительно. Чтобы подобное дополнение было применимо во всех случаях, если сообщение кратно длине блока, в его конец надо добавить целый блок указанного вида.

Потоковые шифры предназначены для преобразования сообщения поэлементно (элементом может быть бит, символ и т. п.). Примером такого вида шифров являются шифры гаммирования.

2.2. СИММЕТРИЧНЫЕ ШИФРЫ

2.2.1. Схема Фейстеля

Современные блочные шифры часто строятся на базе многократного повторения некоторого набора операций преобразования, называемых *раундом шифрования*.

В каждом раунде используется некоторая часть ключа, называемая *раундовым ключом*. Порядок генерации и использования раундовых ключей называется *расписанием использования ключа шифрования*.

В общем виде подобное итеративное преобразование может быть описано следующей формулой:

$$B_i = E(B_{i-1}, K_i), \qquad (2.5)$$

где E – раундовая функция шифрования, B_i – выходной блок, B_{i-1} – входной блок для i-го раунда, K_i – ключ, используемый на i-м раунде; $i = 1 \ldots N$, где N – число раундов. Преобразование E должно быть обратимо. Пусть D – раундовая функция дешифрования. Тогда раунд обратного преобразования описывает формула:

$$B_i = D(B_{i-1}, K_{N-i+1}). \qquad (2.6)$$

Здесь надо обратить внимание на расписание использования ключей в случае прямого и обратного преобразования. Так при шифровании, в первом раунде будет использоваться первый раундовый ключ, а в первом раунде при расшифровывании – последний.

Для разработки итерационных блочных шифров широко используется схема, предложенная в начале 1970-х годов Хорстом Фейстелем (Horst Feistel). Данная схема, также называемая сетью Фейстеля, приведена на рис. 2.2. Ее достоинство заключается в том, что она позволяет использовать любые (в том числе необратимые) функции F для реализации обратимых шифрующих преобразований.

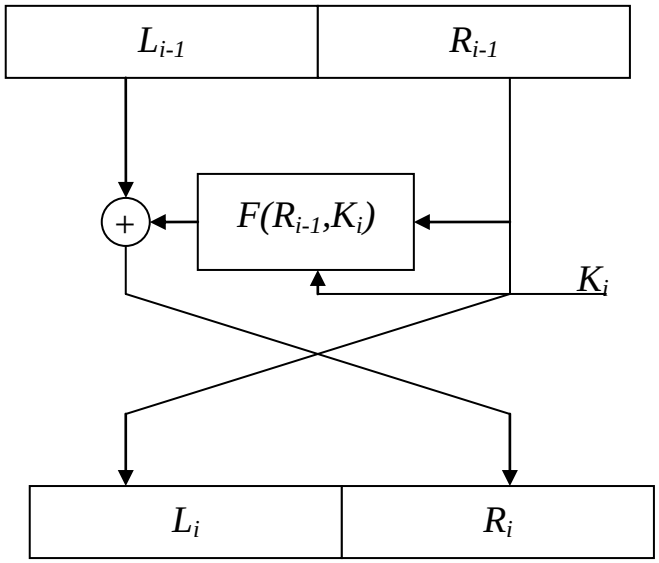


Рис. 2.2. Прямое преобразование по схеме Фейстеля

Входной блок сообщения разбивается на два равных по длине полублока: левый – L и правый – R.

Прямое преобразование осуществляется в соответствии со следующими соотношениями:

$$L_i = R_{i-1},$$
$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i), \qquad (2.7)$$

где $\oplus$ – операция побитного сложения по модулю 2 (в табл.2.1 приведено ее определение). Тогда исходный блок данных – это $L_0 \mid R_0$, а $L_N \mid R_N$ – это выходной блок данных.

Таблица 2.1

Операция сложения по модулю 2

X	Y	X⊕Y
0	0	0
0	1	1
1	0	1
1	1	0

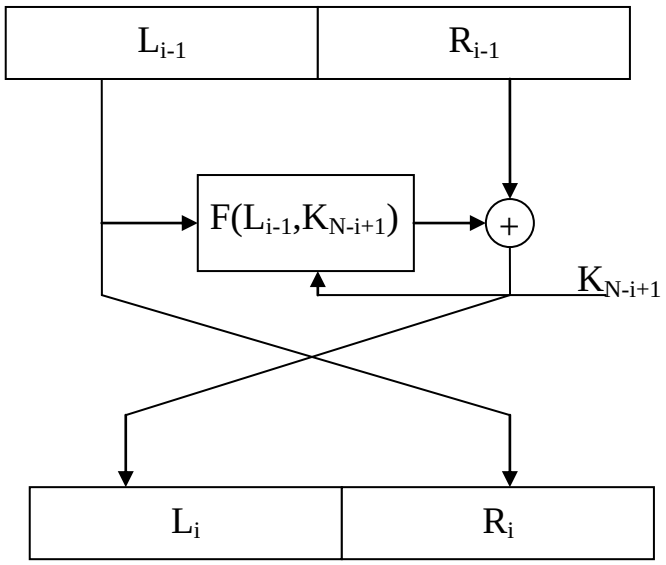


Рис. 2.3. Обратное преобразование по схеме Фейстеля

Как видно из таблицы 2.1, для всех возможных значений операндов выполняется соотношение: (X⊕Y)⊕Y=X. Именно это свойство и позволяет при преобразовании использовать в числе прочих необратимые функции *F* и, несмотря на это, иметь возможность про-

вести обратное преобразование. Обратное преобразование по схеме Фейстеля представлено на рис. 2.3 и описывается формулами:

$$R_i = L_{i-1},$$
$$L_i = R_{i-1} \oplus F(L_{i-1}, K_{N-i+1}). \quad (2.8)$$

2.2.2. Шифр DES

Алгоритм шифрования DES (от англ. «Data Encryption Standard») был опубликован в 1977 году и предназначался для защиты важной, но несекретной информации в государственных и коммерческих организациях США. Реализованные в нем идеи были во многом позаимствованы в более ранней разработке корпорации IBM – шифре «Люцифер» (а как раз в IBM работал Хорст Фейстель, автор рассмотренной в предыдущем параграфе схемы). Но для своего времени «Люцифер» был слишком сложным и его реализации отличались низким быстродействием.

Шифр DES является блочным – преобразования в нем проводятся блоками по 64 бита. Ключ также 64-битный, но значащими являются только 56 бит – каждый 8-й разряд использовался для контроля четности (шифр разрабатывался тогда, когда аппаратура была не слишком надежной и подобные проверки были необходимы). Обобщенная схема алгоритма представлена на рис. 2.4.

Конечная перестановка – IP^{-1} – является обратной по отношению к начальной – IP. Задающие их таблицы жестко определены стандартом (таблицы можно узнать из официального описания стандарта, также они приводятся, например в [9]).

Раунды шифрования стоятся в соответствии с рассмотренной ранее схемой Фейстеля (рис.2.2). Шифрование производится в 16 раундов. Схема раундовой функции F представлена на рис. 2.5.

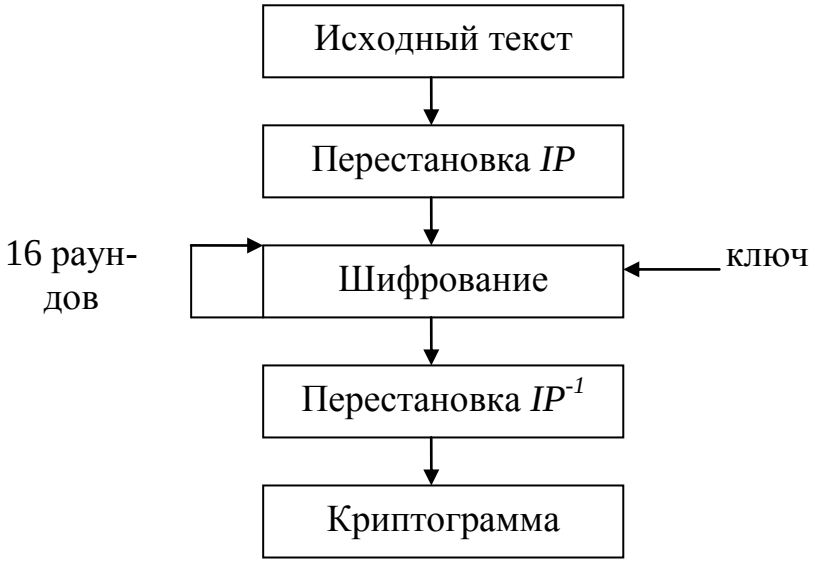


Рис. 2.4. Обобщенная схема шифра DES

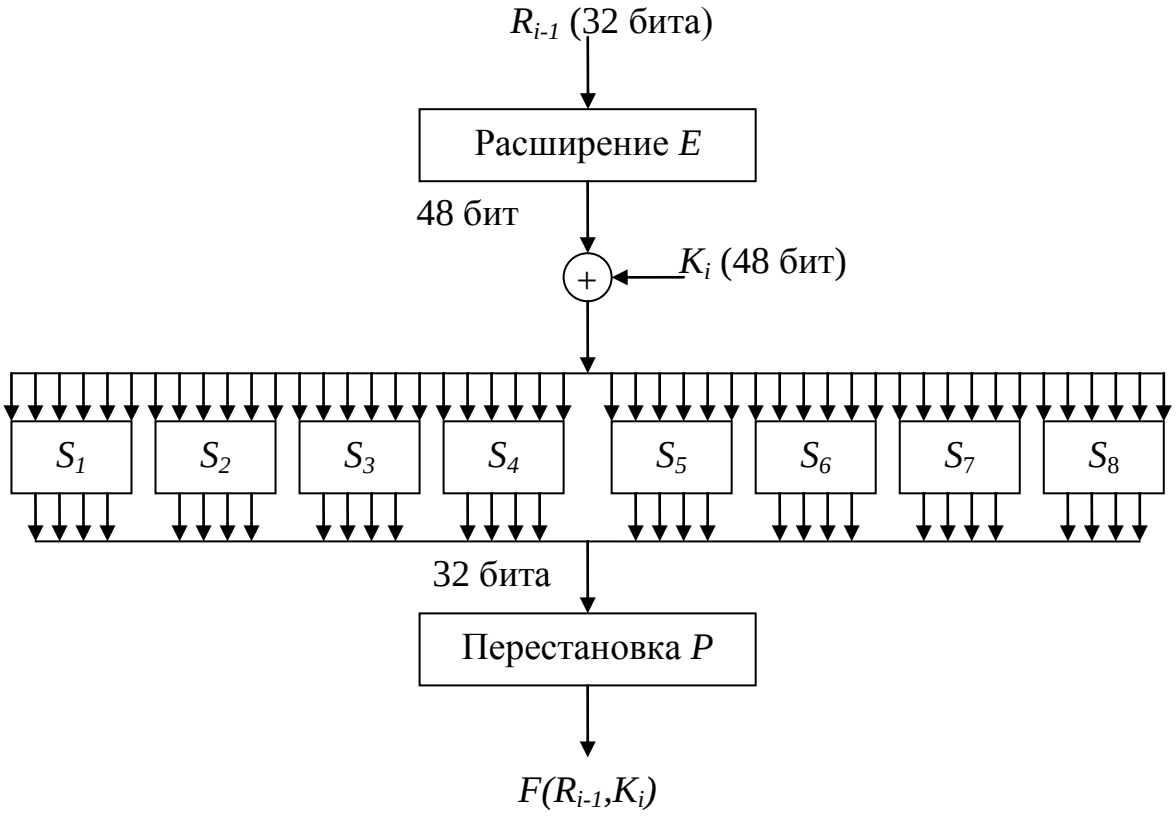


Рис. 2.5. Схема раундовой функции шифра DES

Сначала 32-битный правый полублок преобразуемого текста расширяется до 48 бит с помощью функции расширения *E*. Эта функция производит дублирование и перестановку некоторых элементов блока. В таблице 2.2 показано, как эта функция работает. После расширения в первых позициях полученного 48-битного блока будут стоять 32-й, 1-й и т. д. биты входного блока.

Таблица 2.2

Функция *E* – расширение блока

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, полученный 48-разрядный блок складывается побитно по модулю 2 с раундовым ключом K_i, схема генерации которого будет рассмотрена несколько ниже.

Далее производится подстановка по таблицам $S_1, \ldots, S_8$, в результате которой каждому 6-разрядному входному значению ставится в соответствие 4-разрядное выходное. Таким образом, получив на входе 48 бит, на выходе – снова имеем 32. S_i представляет собой таблицу с 16-ю столбцами и 4-мя строками, содержащую 4-битные элементы. Таблицы подстановки, также как и перестановки, четко определены стандартом.

Пусть на вход подстановки подается 6-разрядный блок $B=b_1b_2...b_6$. Тогда совокупность старшего и младшего разрядов b_1b_6 будет указывать номер строки, а четырехбитное значение $b_2b_3b_4b_5$ – номер столбца. Из ячейки на пересечении найденных строки и столбца будет браться выходное значение подстановки. Полученный соединением выходных значений подстановок $S_1, \ldots, S_8$ 32-битный блок

подвергается перестановке *P*, порядок которой также строго определен в стандарте.

Рассмотрим теперь порядок формирования раундовых ключей на основе секретного. В общем виде алгоритм представлен на рис. 2.6.

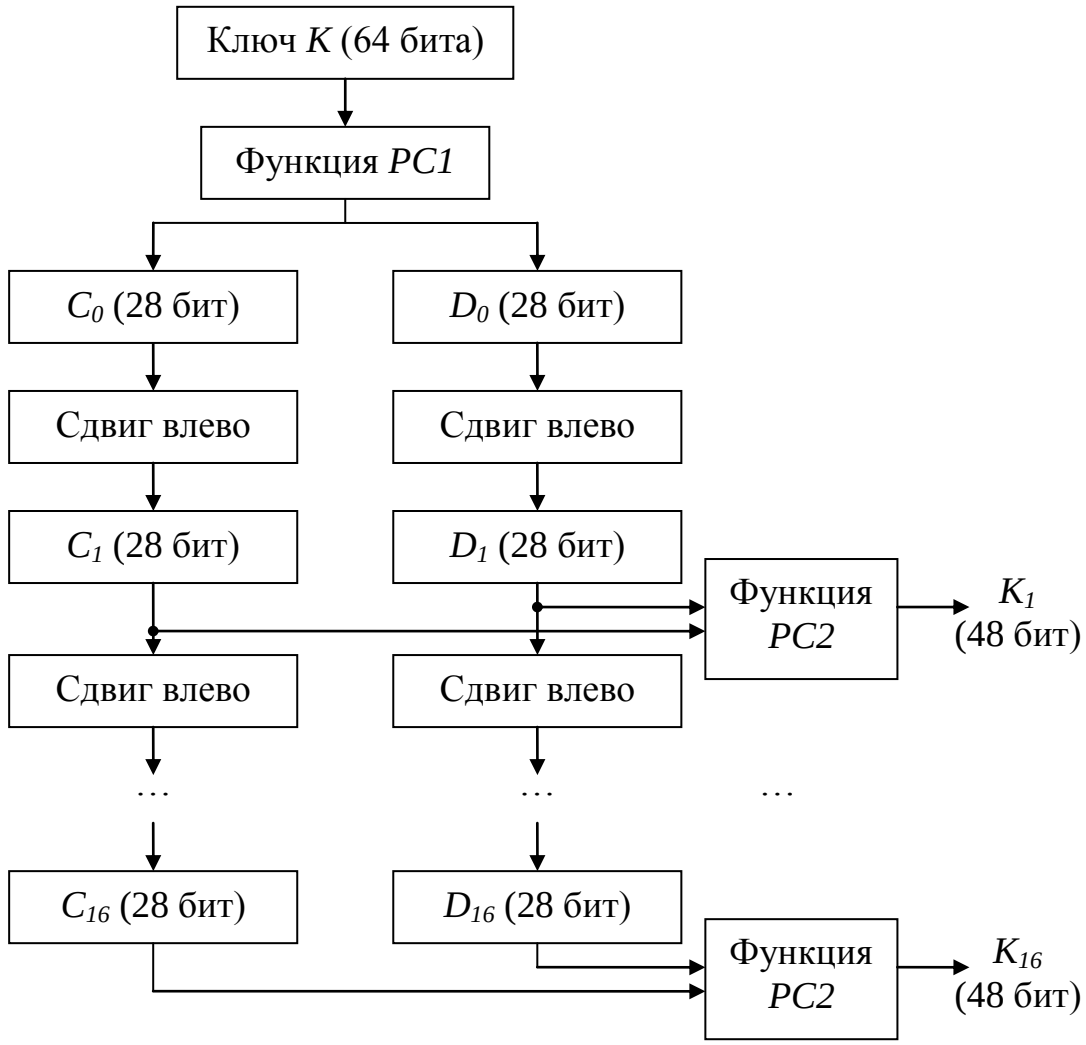


Рис. 2.6. Вычисление раундовых ключей

Как было ранее отмечено, секретный ключ шифра DES имеет длину 64 бита, но каждый 8-й предназначается для контроля четности, поэтому эффективная длина ключа – 56 бит. Функция *PC1* (сокращение «PC» от англ. «Permuted Choice» – выбор с перестановкой) осуществляет перестановку элементов исходного блока, отбрасывая

8-й, 16-й и т. д. биты. После перестановки полученный блок делится на полублоки C_0 и D_0 длиной 28 бит каждый.

В зависимости от номера шага, полублоки C_i и D_i независимо друг от друга преобразуются путем циклического сдвига влево на одну или две позиции (сдвиг на одну позицию производится на 1, 2, 9 и 16-м шагах, в остальных случаях – сдвиг на две позиции). Функция *PC2* преобразует блок $C_i|D_i$, переставляя элементы и отбирая 48 бит, которые и формируют раундовый ключ шифрования K_i (i=1…16).

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. 2.7). 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.

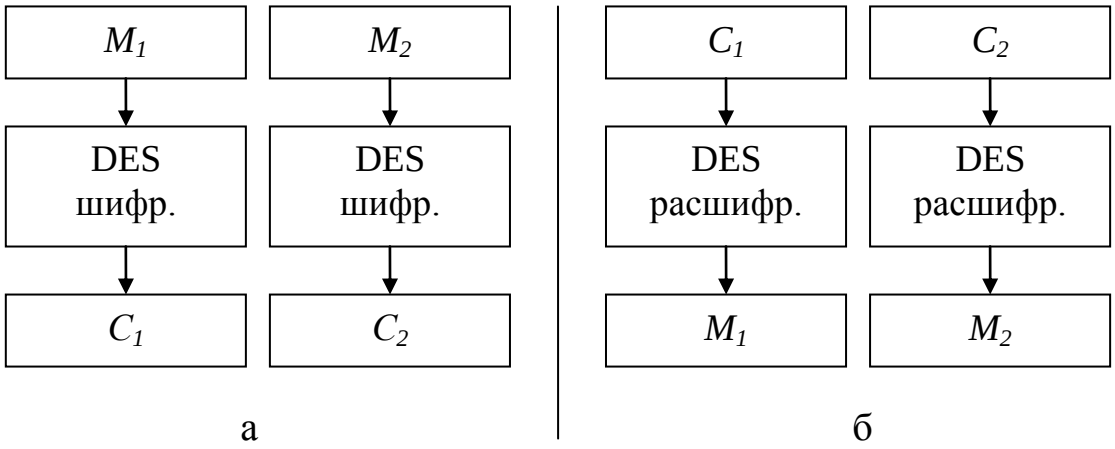


Рис. 2.7. Шифрование (а) и расшифровывание (б) в режиме ECB

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. 2.8.

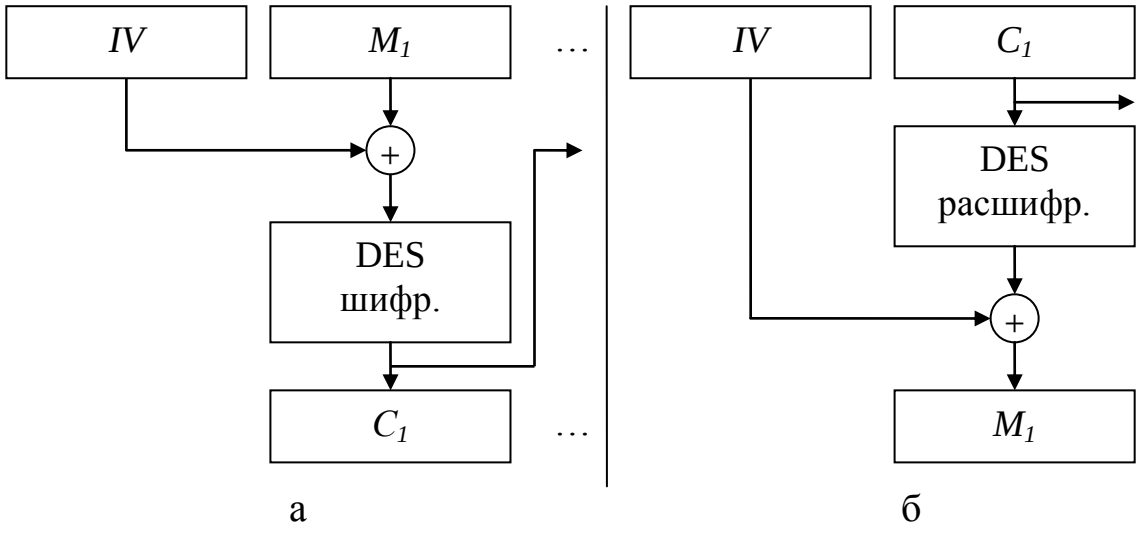


Рис. 2.8. Шифрование (а) и расшифровывание (б) в режиме CBC

Режим CBC используется для шифрования больших сообщений. Как легко заметить, 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 $1 \le r \le 64$. Для построения преобразования используется сдвиговый регистр I, куда на 1-м шаге преобразования помещается инициализирующий вектор IV. Схема преобразования представлена на рис. 2.9.

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 O_j, 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. 2.10. 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}.

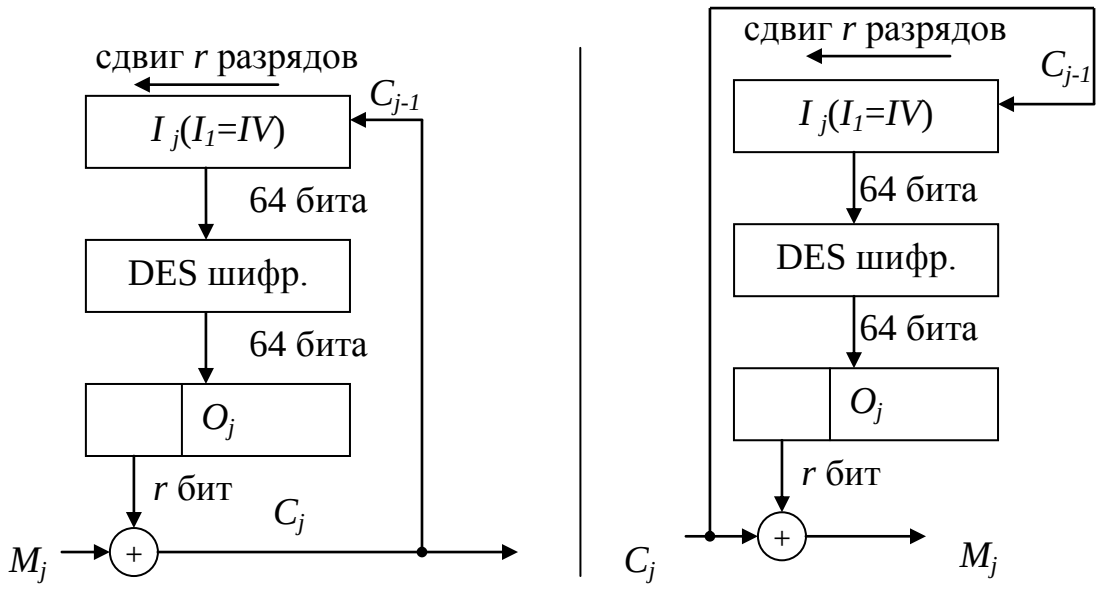


Рис. 2.9. Шифрование (а) и расшифровывание (б) в режиме CFB

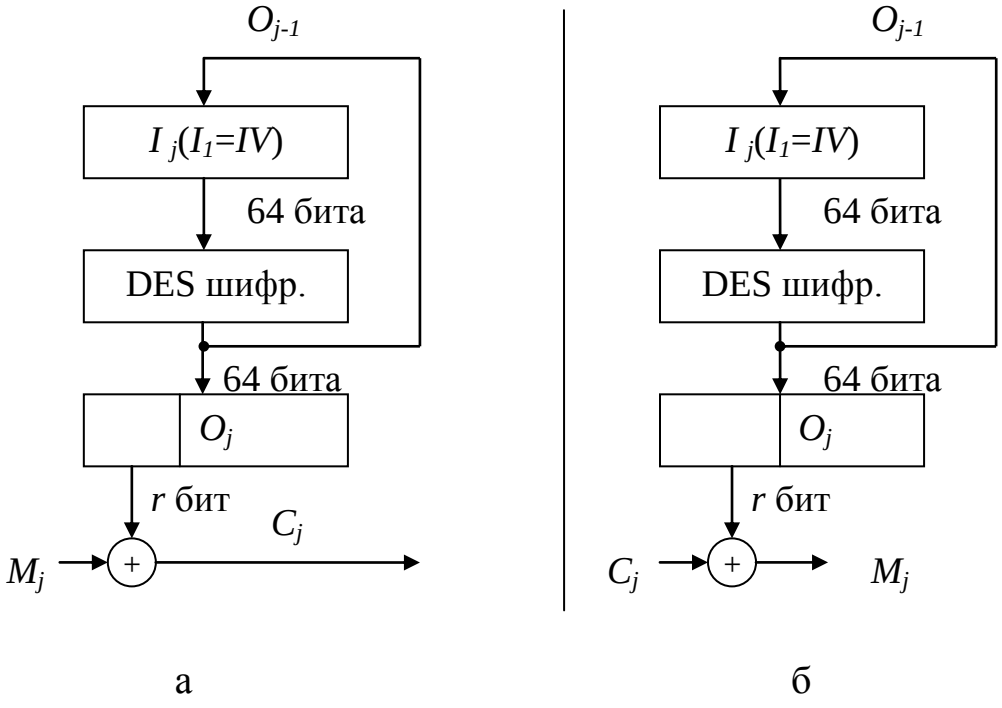


Рис. 2.10. Шифрование (а) и расшифровывание (б) в режиме OFB

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 псевдослучайная 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.

Сначала было предложено использовать повторное шифрование на разных ключах. Обозначим шифрование на ключе k как E_k, а расшифровывание как D_k. Тогда предлагаемая схема прямого преобразования описывалась как $C=E_{k2}(E_{k1}(M))$, обратного – $M=D_{k1}(D_{k2}(C))$. Однако впоследствии было доказано, что из-за того, что два раза используется одно и то же преобразование, против подобной схемы может быть успешно применена атака «встреча посередине». Ее суть заключается в том, что если хранить в памяти большой объем предвычислений (расчет криптограммы для всех возможных значений ключа), то можно взломать приведенную выше схему двукратного шифрования за 2^{n+1} попыток (вместо 2^{2n} как было бы при удвоении длины ключа).

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. 2.11.

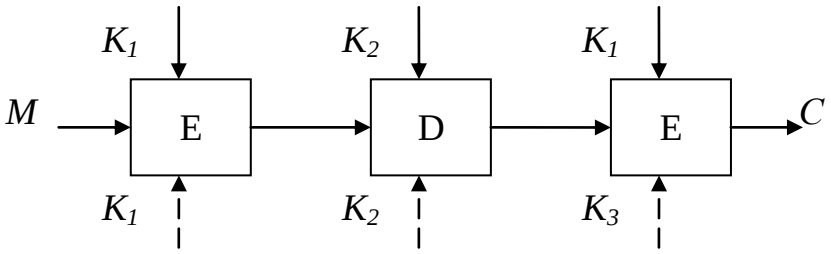


Рис. 2.11. Шифр Triple DES

Использование в шифре Triple DES различных ключей и преобразований (шифрование и расшифровывание) позволяет противостоять атаке «встреча посередине». В первом случае, выбор ключей производится так, как показано в верхней части рисунка: сначала производится шифрование на ключе K_1, далее – расшифровывание на ключе K_2, 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.

За счет более высокой надежности в настоящее время шифр Triple DES используется чаще, чем шифр DES.

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. 2.12.

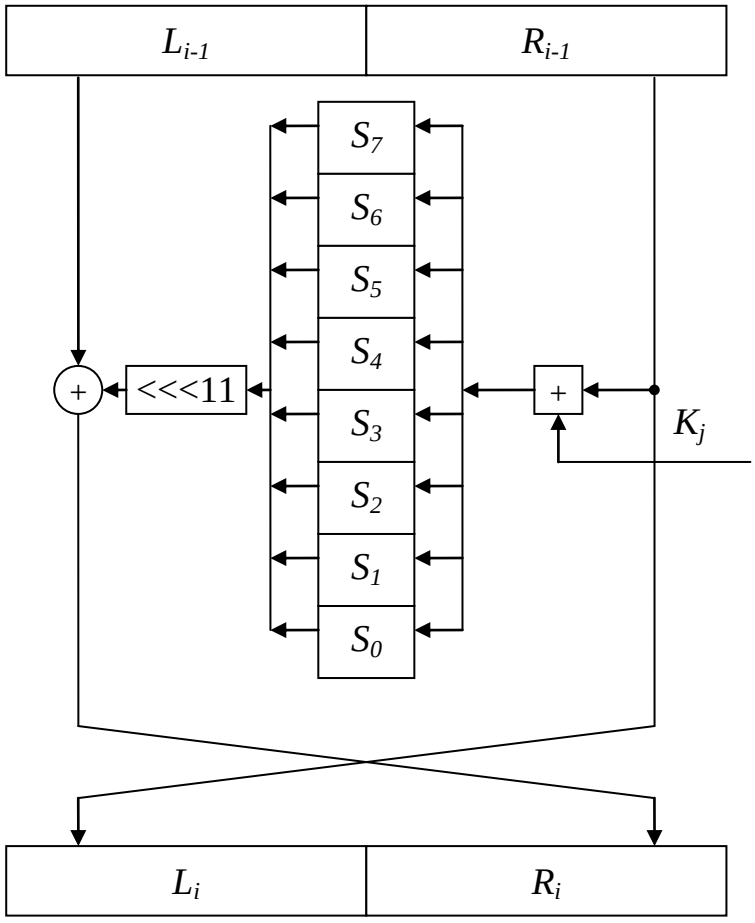


Рис. 2.12. Схема раунда алгоритма ГОСТ 28147-89

Преобразование производится в следующем порядке. Правый полублок R_{i-1} складывается по модулю 2^{32} (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:

$$j = (i-1) \bmod 8 \quad \text{для} \quad 1 \le i \le 24,$$
$$j = (32-i) \bmod 8 \quad \text{для} \quad 25 \le i \le 32. \qquad (2.9)$$

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. Надо отметить, что значения таблиц подстановки не определены стандартом, как это сделано, например, в шифре DES. В то же время, стойкость алгоритма существенно зависит от их правильного выбора.

Считается, что конкретные значения этих таблиц должны храниться в секрете и это своеобразные ключевые элементы, которые являются общими для всей организации или подразделения и редко изменяются.

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.

2.2.4. Шифр Blowfish

Шифр Blowfish был разработан известным американским криптографом Брюсом Шнейером (Bruce Schneier) в 1993 году. Алгоритм 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 $K_1, \ldots, K_{18}$ (размерность K_i – 32 бита) и матрицу подстановок Q с 4-мя строками, 256-ю столбцами и 32-битными элементами.

$$Q = \begin{vmatrix} Q_0^{(1)} & \cdots & Q_{255}^{(1)} \\ Q_0^{(2)} & \cdots & Q_{255}^{(2)} \\ Q_0^{(3)} & \cdots & Q_{255}^{(3)} \\ Q_0^{(4)} & \cdots & Q_{255}^{(4)} \end{vmatrix} . \qquad (2.10)$$

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):

$$F(X) = ((Q^{(1)}_{X_3} + Q^{(2)}_{X_2}) \oplus Q^{(3)}_{X_1}) + Q^{(4)}_{X_0}. \qquad (2.11)$$

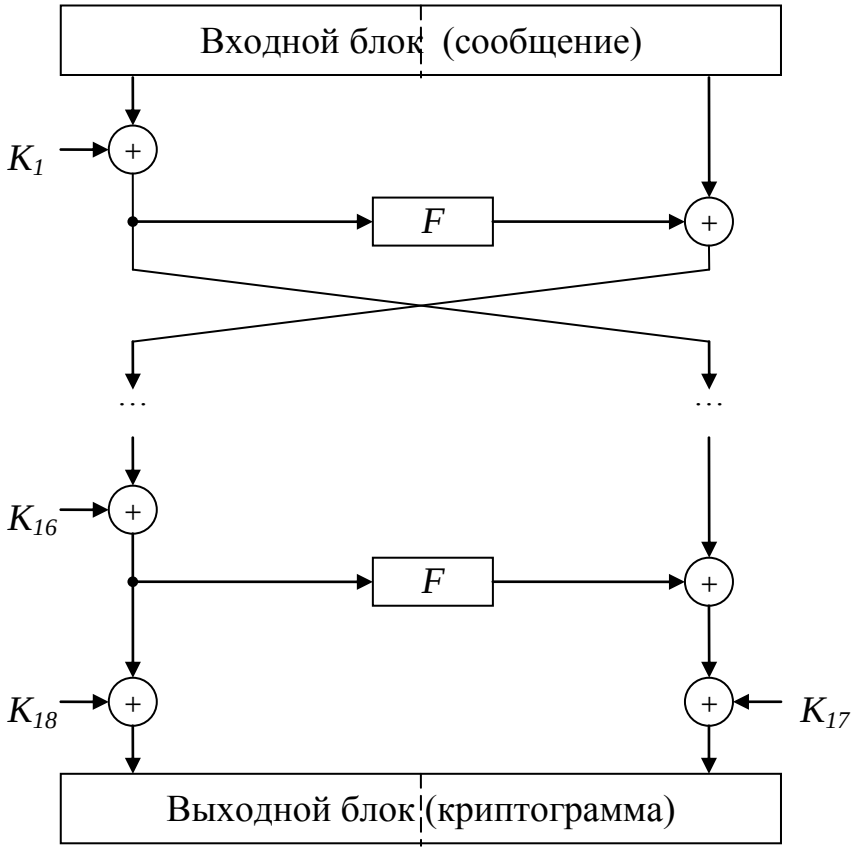


Рис. 2.13. Шифр Blowfish

Схема шифрующего преобразования приведена на рис.2.13. 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 *KS*, K_2 – со следующими 32-мя битами и т. д. Когда ключ *KS* заканчивается, его начинают использовать сначала. Суммируются только K_i (без *Q*).

3) 64-битный блок нулей $\boldsymbol{0}=(0\ldots0)$ зашифровывают с помощью Blowfish на ключах, полученных на шагах 1) и 2): $C_0=Blowfish(\boldsymbol{0})$.

4) Раундовые подключи K_1 и K_2 заменяют полученным на шаге 3) значением C_0.

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

6) Раундовые ключи K_3 и K_4 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.

2.3. УПРАВЛЕНИЕ КРИПТОГРАФИЧЕСКИМИ КЛЮЧАМИ ДЛЯ СИММЕТРИЧНЫХ ШИФРОВ

Под *ключевой информацией* понимают совокупность всех ключей, действующих в системе. Если не обеспечено достаточно надежное и безопасное управление ключевой информацией, то эффект от применения криптографической защиты данных может быть сведен к нулю: завладев ключами нарушитель сможет получить доступ и к защищаемой информации. Процесс управления ключами включает в себя реализацию трех основных функций:

- генерация ключей;
- хранение ключей;
- распределение ключей.

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. Но надо учитывать, что получаемая в этом случае последовательность будет псевдослучайной – если программный генератор повторно запустить с такими же начальными значениями, он выдаст ту же последовательность.

В программных генераторах ключей нередко используют алгоритмы шифрования и ключи, специально резервируемые для задач генерации. В качестве начальных значений могут браться, например, значения таймера вычислительной системы.

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, более подробно – см. раздел 2.5). Но в этом случае надо учитывать, что новый ключ безопасен в той же мере, что и прежний, т. к. противник может повторить всю цепочку модификаций.

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.

2.3.1. Протокол Kerberos

Протокол Kerberos был разработан в Массачусетском технологическом институте в середине 1980-х годов и 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. 2.14.

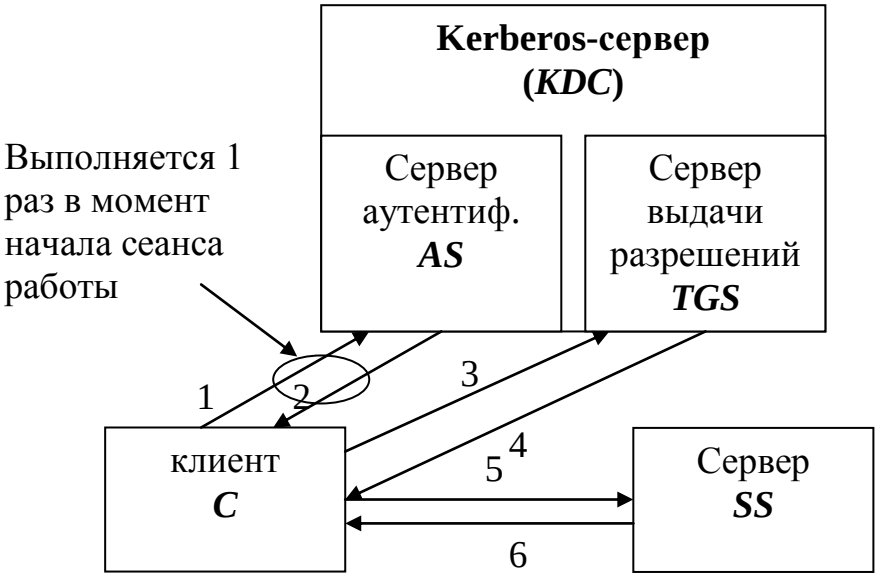


Рис. 2.14. Протокол Kerberos

Пусть клиент *C* собирается начать взаимодействие с сервером *SS* (от англ. «Service Server» – сервер, предоставляющий сетевые сервисы). В несколько упрощенном виде, протокол предполагает следующие шаги [10,11].

1). C->AS: {c}.

Клиент *C* посылает серверу аутентификации *AS* свой идентификатор *c* (идентификатор передается открытым текстом).

2). AS->C: $\{\{TGT\}K_{AS\_TGS}, K_{C\_TGS}\}K_C$,

где K_C – основной ключ *C*;

K_{C_TGS} – ключ, выдаваемый *C* для доступа к серверу выдачи разрешений *TGS*;

{TGT} — билет на доступ к серверу выдачи разрешений (англ. «Ticket Granting Ticket»); $\{TGT\}=\{c,tgs,t_1,p_1, K_{C\_TGS}\}$, где *tgs* – идентификатор сервера выдачи разрешений, t_1 – отметка времени, p_1 – период дейст-

вия билета. Запись $\{...\}K_X$ здесь и далее означает, что содержимое фигурных скобок зашифровано на ключе K_X.

На этом шаге сервер аутентификации *AS*, проверив, что клиент *C* имеется в его базе, возвращает ему билет для доступа к серверу выдачи разрешений и ключ для взаимодействия с сервером выдачи разрешений. Вся посылка зашифрована на ключе клиента *C*. Таким образом, даже если на первом шаге взаимодействия идентификатор *c* послал не клиент *C*, а нарушитель *X*, то полученную от *AS* посылку *X* расшифровать не сможет.

Получить доступ к содержимому билета *TGT* не может не только нарушитель, но и клиент *C*, так как билет зашифрован на ключе, который распределили между собой сервер аутентификации и сервер выдачи разрешений.

3). C->TGS: $\{TGT\}K_{AS\_TGS}$, $\{Aut_1\}\ K_{C\_TGS}$, *{ID},*

где $\{Aut_1\}$ – аутентификационный блок – $Aut_1 = \{c,t_2\}$, t_2 – метка времени; *ID* – идентификатор запрашиваемого сервиса (в частности, это может быть идентификатор сервера *SS*).

Клиент *C* на этот раз обращается к серверу выдачи разрешений *TGS*. Он пересылает полученный от *AS* билет, зашифрованный на ключе K_{AS_TGS}, и аутентификационный блок, содержащий идентификатор *c* и метку времени, показывающую, когда была сформирована посылка.

Сервер выдачи разрешений расшифровывает билет *TGT* и получает из него информацию о том, кому был выдан билет, когда и на какой срок, ключ шифрования, сгенерированный сервером *AS* для взаимодействия между клиентом *C* и сервером *TGS*. С помощью этого ключа расшифровывается аутентификационный блок. Если метка в блоке совпадает с меткой в билете, это доказывает, что посылку сгенерировал на самом деле *C* (ведь только он знал ключ K_{C_TGS} и мог правильно зашифровать свой идентификатор). Далее делается проверка времени действия билета и времени оправления посылки *3)*. Ес-

ли проверка проходит и действующая в системе политика позволяет клиенту *C* обращаться к клиенту *SS*, тогда выполняется шаг *4).*

4). TGS->C: {{TGS}K_{TGS_SS}*,*K_{C_SS}*}*K_{C_TGS}*,*

где K_{C_SS} – ключ для взаимодействия *C* и *SS*, *{TGS}* – от англ. «Ticket Granting Service» – билет для доступа к *SS* (обратите внимание, что такой же аббревиатурой в описании протокола обозначается и сервер выдачи разрешений). *{TGS} ={c,ss,*t_3*,*p_2*,* K_{C_SS} *}.*

Сейчас сервер выдачи разрешений *TGS* посылает клиенту *C* ключ шифрования и билет, необходимые для доступа к серверу *SS*. Структура билета такая же, как на шаге 2): идентификатор того, кому выдали билет; идентификатор того, для кого выдали билет; отметка времени; период действия; ключ шифрования.

5). C->SS: {TGS}K_{TGS_SS}*, {Aut*$_2$*}* K_{C_SS} *,*

где Aut_2*={c,*t_4*}.*

Клиент *C* посылает билет, полученный от сервера выдачи разрешений, и свой аутентификационный блок серверу *SS*, с которым хочет установить сеанс защищенного взаимодействия. Предполагается, что *SS* уже зарегистрировался в системе и распределил с сервером *TGS* ключ шифрования K_{TGS_SS}. Имея этот ключ, он может расшифровать билет, получить ключ шифрования K_{C_SS} и проверить подлинность отправителя сообщения.

6). SS->C: {t_4*+1}*K_{C_SS}

Смысл последнего шага заключается в том, что теперь уже *SS* должен доказать *C* свою подлинность. Он может сделать это, показав, что правильно расшифровал предыдущее сообщение. Вот поэтому, *SS* берет отметку времени из аутентификационного блока *C*, изменяет ее заранее определенным образом (увеличивает на 1), шифрует на ключе K_{C_SS} и возвращает *C*.

Если все шаги выполнены правильно и все проверки прошли успешно, то стороны взаимодействия *C* и *SS*, во-первых, удостоверились в подлинности друг друга, а во-вторых, получили ключ шифрования для защиты сеанса связи – ключ K_{C_SS}.

Нужно отметить, что в процессе сеанса работы клиент проходит шаги *1)* и *2)* только один раз. Когда нужно получить билет на доступ к другому серверу (назовем его *SS1*), клиент *C* обращается к серверу выдачи разрешений *TGS* с уже имеющимся у него билетом, т. е. протокол выполняется начиная с шага *3)*.

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. Схема взаимодействия между Kerberos-областями представлена на рис. 2.15.

Для взаимодействия между областями, должна быть осуществлена взаимная регистрация серверов Kerberos, в процессе которой сервер выдачи разрешений одной области регистрируется в качестве клиента в другой области (т. е. заносится в базу сервера аутентификации и разделяет с ним ключ).

После установки взаимных соглашений, клиент из области 1 (пусть это будет K_{11}) может установить сеанс взаимодействия с клиентом из области 2 (например, K_{21}). Для этого K_{11} должен получить у своего сервера выдачи разрешений билет на доступ к Kerberos-серверу, с клиентом которого он хочет установить взаимодействие (на рисунке это сервер *KDC2*). Полученный билет содержит отметку о том, в какой области зарегистрирован владелец билета. Билет шифруется на ключе, разделенном между серверами *KDC1* и *KDC2*. При успешной расшифровке билета, удаленный Kerberos-сервер может быть уверен, что билет выдан клиенту Kerberos-области, с которой установлены доверительные отношения. Далее протокол работает как обычно.

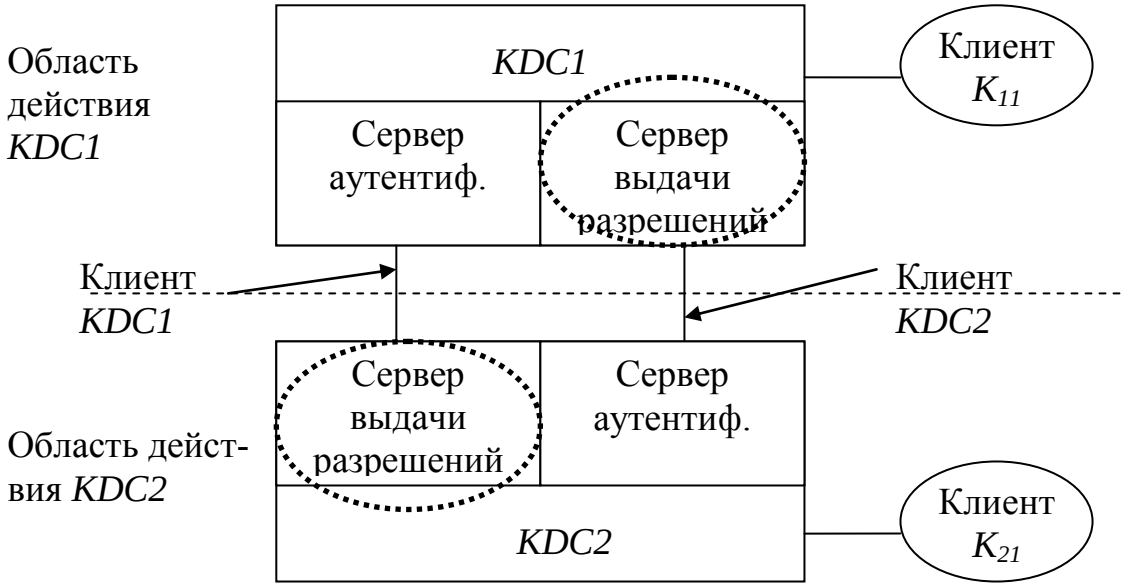


Рис. 2.15. Взаимодействие между Kerberos-областями

Кроме рассмотренных, Kerberos предоставляет еще ряд дополнительных возможностей. Например, указанный в структуре билета параметр *p* (период времени) задается парой значений «время начала действия» – «время окончания действия», что позволяет получать билеты отложенного действия.

Имеется тип билета «с правом передачи», что позволяет, например, серверу выполнять действия от имени обратившегося к нему клиента.

2.4. АСИММЕТРИЧНЫЕ ШИФРЫ

2.4.1 Основные понятия

Несмотря на достижения в области симметричной криптографии, к середине 1970-х годов стала остро осознаваться проблема неприменимости данных методов для решения целого ряда задач.

Во-первых, при использовании симметричных шифров необходимо отдельно решать часто нетривиальную задачу распределения ключей. Несмотря на использование иерархий ключей и центров распределения, в какой-то начальный момент ключ (или мастер-ключ)

должен быть передан по безопасному каналу. Но такого канала может просто не быть, или он может быть достаточно дорогостоящим.

Во-вторых, при использовании методов симметричного шифрования подразумевается взаимное доверие сторон, участвующих во взаимодействии. Если это не так, совместное использование одного и того же секретного ключа может быть нежелательно.

Третья проблема связана с необходимостью проведения аутентификации информации и защиты от угроз, связанных с отказом отправителя (получателя) от факта отправки (получения) сообщений.

Перечисленные проблемы являются весьма существенными, и работа над их решением привела к появлению асимметричной криптографии, также называемой криптографией с открытым ключом.

Рассмотрим ряд определений.

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

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

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

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

1) для всякого $x \in X$ легко вычислить значение функции $y = F_k(x)$, где $y \in Y$, 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 $y \in Y$ 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 со-

общение для данного получателя, вычислив значение криптограммы $C = F_k(M)$. Расшифровать данную криптограмму может только законный получатель, которому известен секрет k.

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

В отличие от симметричных, в асимметричных алгоритмах ключи используются парами – открытый ключ (англ. «public key») и секретный или закрытый (англ. «private key»). 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. 2.16). 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.

Если использовать определение односторонней функции с секретом, то алгоритм шифрования и открытый ключ задают прямое преобразование F_k, алгоритм расшифровывания задает обратное преобразование, а секретный ключ получателя играет роль «секрета» k.

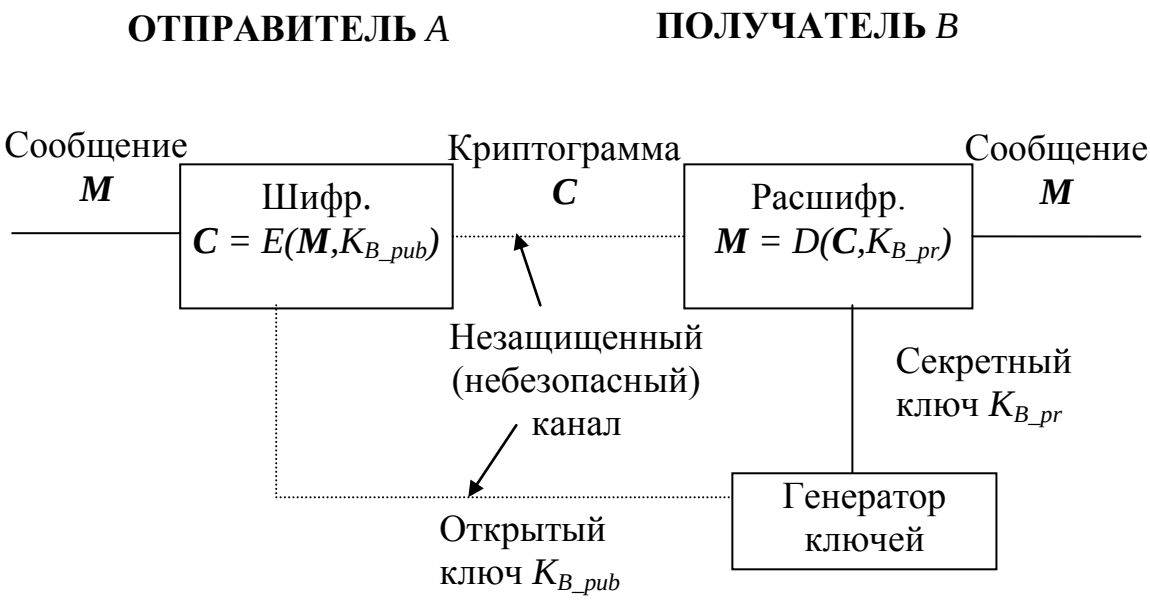


Рис. 2.16. Асимметричное шифрование

Рассмотрим теперь вопрос аутентификации сообщений.

Электронная цифровая подпись (ЭЦП)[1] – 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.

[1] Определение в соответствии федеральным законом Российской Федерации «Об электронной цифровой подписи».

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 2.17.

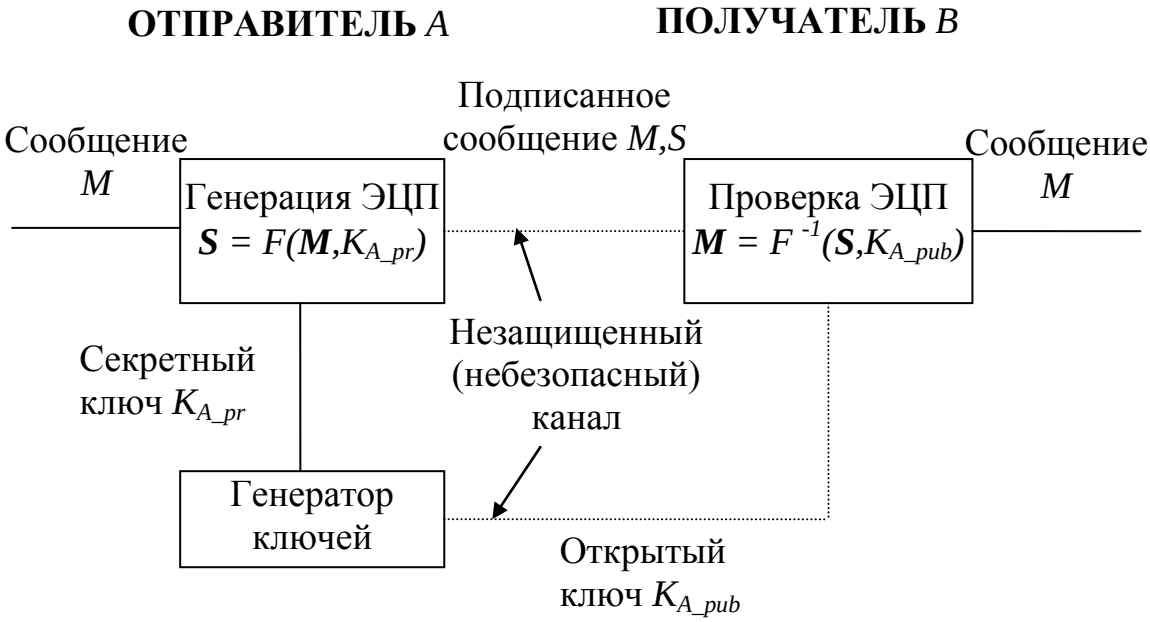


Рис.2.17. Электронная цифровая подпись

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.

2.4.2. Распределение ключей по схеме Диффи-Хеллмана

Как уже отмечалось выше, 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.

Для того, чтобы лучше разобраться с особенностями данного алгоритма, рассмотрим несколько определений из теории чисел. Два целых числа n и n' называются *сравнимыми по модулю* m, если при делении на m они дают одинаковые остатки: $n \equiv n' \ (mod\ m)$, m – модуль сравнения. Таким образом, можно разбить множество целых чисел $\boldsymbol{Z}$ на классы чисел, сравнимых между собой по модулю m и *называемых классами вычетов* по модулю m. Каждый класс вычетов имеет вид:

$$\{r\}_m = \{r + mk \mid k \in Z\}. \qquad (2.12)$$

Множество всех классов вычетов по модулю m обозначается как $\boldsymbol{Z_m}$ или $\boldsymbol{Z/mZ}$. Каждым двум классам $\{k\}_m$ и $\{l\}_m$ независимо от выбора их представителей, можно сопоставить класс, являющийся их суммой и произведением, таким образом, однозначно определяются операции сложения и умножения. Множество $\{\boldsymbol{Z_m}, +, \times\}$ является коммутативным кольцом с единицей, а если число m – простое, то конечным полем.

Мультипликативная группа $\{\boldsymbol{Z_m}, \times\}$ при $m=1,2,4,p^k,2p^k$ (где $k \in \boldsymbol{N}$, p – нечетное простое число) является циклической [12], т. е. существует образующий элемент $a \in \boldsymbol{Z_m}$, такой что степени a в определенном порядке дают все значения от 0 до $m-1$. Элемент a также называется *первообразным корнем* по модулю m.

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

$$y = a^x \bmod p. \qquad (2.13)$$

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

при достаточно больших значениях модуля, показателя и основания степени функцию (2.13) можно считать необратимой.

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 [12]:

$$НОД(a, p)=1;\ \ a^{(p-1)/q_j} \neq 1(\text{mod } p),\ \ j=1,...,k, \qquad (2.14)$$

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 из (2.13). 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 $Y_A = a^{X_A} \bmod p$.

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

3). Абоненты обмениваются открытыми ключами Y_A и Y_B.

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

$$K_{AB} = (Y_B)^{X_A} \bmod p = (a^{X_B})^{X_A} \bmod p = (a^{X_A})^{X_B} \bmod p = K_{BA}. \qquad (2.15)$$

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 зада-

чи, чем дискретное логарифмирование, что и обеспечивает криптографическую стойкость алгоритма.

2.4.3. Криптографическая система RSA

Алгоритм RSA был предложен в 1977 году и стал первым полноценным алгоритмом асимметричного шифрования и электронной цифровой подписи. Алгоритм назван по первым буквам фамилий авторов – Рональд Райвест (Ronald Rivest), Ади Шамир (Adi Shamir) и Леонард Адлеман (Leonard Adleman). 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:

$$M^{\varphi(n)} \equiv 1(\mathrm{mod}\, n), \qquad (2.16)$$

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 взаимно простых 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:

$$C = M^e (\mathrm{mod}\, n). \qquad (2.17)$$

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

$$M = C^d (\mathrm{mod}\, n) = (M^e)^d (\mathrm{mod}\, n) = M^{ed} (\mathrm{mod}\, n). \qquad (2.18)$$

Как уже отмечалось, теорема Эйлера утверждает, что $M^{\varphi(n)} = 1 (\mathrm{mod}\, n)$ или, что то же самое, $M^{k\varphi(n)+1} = M (\mathrm{mod}\, n)$, где k – натуральный множитель. Сопоставив данное выражение с выражением (2.18) получаем, что e и d должны быть связаны соотношением:

$$e \cdot d = k \cdot \varphi(n) + 1 \Leftrightarrow ed \equiv 1 (\mathrm{mod}\, \varphi(n)). \qquad (2.19)$$

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:

$$\varphi(p \cdot q) = \varphi(p) \cdot \varphi(q) = (p-1) \cdot (q-1). \qquad (2.20)$$

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

4) Вычисляется значение e такое что:

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

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

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:

$$C = M^e \bmod n. \qquad (2.21).$$

Криптограмма C передается получателю. Владелец секретного ключа d может расшифровать сообщение по формуле (2.22).

$$M = C^d \bmod n. \qquad (2.22)$$

Рассмотрим теперь схему построения э*лектронной цифровой подписи.* Сообщение *M* подписывается с использованием секретного ключа *d* (но для генерации подписи используется уже ключевая пара отправителя):

$$S = M^d \bmod n. \qquad (2.23)$$

Отправитель передает получателю подписанное сообщение, т. е. пару значений *(M,S).* Проверка ЭЦП по открытому ключу (*n,e*) производится так:

$$M' = S^e \bmod n. \qquad (2.24)$$

Совпадение значений *M* и *M'* служит доказательством того, что сообщение получено от владельца соответствующего секретного ключа и не было изменено в процессе передачи.

Как видно, сами преобразования относительно просты. Основную сложность при реализации алгоритма RSA представляет этап генерации ключей. В частности, простые числа *p* и *q* выбираются такими, что:

- одно из них должно быть меньше другого на несколько порядков;

- *(p-1)* и *(q-1)* должны иметь как можно меньший НОД;

- хотя бы одно из чисел *(p-1), (q-1)* должно иметь в разложении большой простой множитель (например, *(p-1) = 2t*, где *t* – простое).

Точное определение, является ли большое число простым или нет, во многих случаях является вычислительно сложной задачей. Поэтому, как правило, используются «псевдопростые» тесты, которые позволяют с достаточно большой вероятностью отбросить числа не являющиеся простыми. Один из таких тестов основан на малой теореме Ферма, которая гласит, что если *p* – простое число и $1 \le x < p$, то $x^{p-1} = 1 \bmod p$. Проверив «кандидата» в простые числа с несколькими *x*, выбираемыми в соответствии со специальными требованиями, можно с большой вероятностью выяснить, является ли он простым.

Нахождение наибольшего общего делителя и определение значения *e* на шаге 4) алгоритма генерации ключей производится в соот-

ветствии с алгоритмом Евклида и обобщенным алгоритмом Евклида [7,9,12].

2.4.4. Криптографическая система Эль-Гамаля

В 1984 году американским исследователем египетского происхождения Тахером Эль-Гамалем (Taher Elgamal) были опубликованы алгоритмы шифрования с открытым ключом и ЭЦП, получившие его имя. 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: $c_1 = a^k \bmod p$, $c_2 = rM \bmod p$.

Надо отметить, что в [10] приводится вариация данного алгоритма, где вторая часть криптограммы рассчитывается как $c_2^{'} = r \oplus M$, где $\oplus$ – побитное сложение по модулю 2.

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: $c_1^x \equiv (a^k)^x \equiv r \bmod p$. Тогда $M \equiv c_2 r^{-1} \equiv c_2 (c_1{}^x)^{-1} \bmod p$. Или для варианта со сложением по модулю 2: $M \equiv c_2{}' \oplus c_1{}^x \bmod p$.

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. Пусть, например, нарушитель знает сообщение M, соответствующие ему криптограммы c_1 и c_2, и криптограммы c_1' и c_2'. Из-за того, что k и ключи не менялись, будут совпадать r и r', c_1 и c_1'. M' нарушитель может рассчитать как:

$$M \equiv c_2 r^{-1} \bmod p; \qquad M' \equiv c_2' r^{-1} \equiv c_2' M c_2{}^{-1} \bmod p. \qquad (2.25).$$

Рассмотрим теперь предложенный Эль-Гамалем *алгоритм электронной цифровой подписи*. Надо отметить, что он получил широкое распространение и на его базе был разработан американский стандарт ЭЦП DSA (англ. «Digital Signature Algorithm»).

Как и при шифровании, стороны согласуют параметры a и p. После этого, отправитель выбирает секретный ключ x и рассчитывает открытый ключ $y = a^x \bmod p$.

Подписываемое сообщение M должно удовлетворять условию $0 \le M < p$. Подписью абонента служит пара чисел r и s ($0 \le r < p$, $0 \le s < p$), которые удовлетворяют соотношению:

$$a^M \equiv y^r r^s \bmod p \qquad (2.26).$$

Получатель, знающий сообщение и открытый ключ, может проверить выполнение этого равенства. Но только владелец секретного ключа x может правильно рассчитать значения r и s. Для этого он выполняет следующие действия.

1. Выбирает случайное число k: $1 < k < p-1$, $НОД(k,p-1)=1$.

2. Вычисляет $r = a^k \bmod p$.

3. Вычисляет s из уравнения $M \equiv xr + ks \bmod(p-1)$. Это уравнение получено, основываясь на доказанном в теории числе утверждении: если p – простое, a – целое, $1<a<p$ (в этом случае, a и p будут и взаимно просты), то $a^x \equiv a^y \bmod p$ равносильно $x \equiv y \bmod(p-1)$ для любых целых неотрицательных x, y. Таким образом, $s \equiv (M - xr)k^{-1} \bmod(p-1)$. При выполнении условия *НОД(k,p-1)=1*, s существует и единственно.

Отправитель передает сообщение с подписью (M,r,s) получателю, который, пользуясь соотношением (2.26), может проверить ЭЦП.

При применении алгоритма ЭЦП Эль-Гамаля, также как и в случае шифрования, недопустимо использовать одно и то же значение k для подписи двух разных сообщений.

2.4.5. Совместное использование симметричных и асимметричных шифров

Основным достоинством криптографических алгоритмов с открытым ключом является возможность решения таких задач, как распределение ключа по небезопасному каналу, аутентификации сообщения и отправителя, и т. д. В то же время, асимметричные шифры работают существенно более медленно, чем симметричные. Это связано с необходимостью производить операции над сверхбольшими числами. Поэтому симметричные и асимметричные алгоритмы часто используют вместе – для распределения ключей и ЭЦП используют криптографию с открытым ключом, данные шифруют с помощью симметричных алгоритмов.

При анализе системы, в которой совместно используются несколько алгоритмов, принято оценивать сложность ее взлома по сложности взлома самого слабого звена. В литературе [9] приводится примерное соответствие длин ключей для алгоритма симметричного шифрования (атака производится путем перебора ключевого множе-

ства) и алгоритма RSA, обеспечивающих сопоставимую стойкость. Например, 64-битному ключу симметричного шифрования примерно соответствует 512-битный ключ RSA, а 128-битному – ключ RSA длиной более 2300 бит.

2.5. ХЭШ-ФУНКЦИИ

В рассмотренных в разделе 2.4 алгоритмах формирования ЭЦП длина подписи получается равной или даже большей, чем длина самого сообщения. Очевидно, что удостоверять подобным образом большой документ неудобно. Поэтому подписывается, как правило, не само сообщение, а его «дайджест» – значение фиксированной длины, зависящее от подписываемого сообщения. Для формирования дайджеста используется *хэш-функция* (от англ. «hash function») – односторонняя функция, преобразующая строку произвольной длины в строку фиксированной длины. В криптографии используются хэш-функции 2 классов:

- хэш-функции без ключа;
- хэш-функции с ключом.

2.5.1. Хэш-функции без ключа

Хэш-функции без ключа делятся на слабые и сильные. *Слабой хэш-функцией* называется односторонняя функция $H(x)$, удовлетворяющая следующим условиям:

1) аргумент может быть строкой бит произвольной длины;

2) значение функции $H(x)$ должно быть строкой бит фиксированной длины;

3) значение $H(x)$ легко вычислить;

4) для любого фиксированного аргумента x вычислительно невозможно найти другой $x' \neq x$, такой что $H(x')=H(x)$.

Пара значений $x' \neq x$: $H(x')=H(x)$ называется *коллизией* хэш-функции.

Сильной хэш-функцией называется односторонняя функция $H(x)$, удовлетворяющая условиям 1) – 3) и последнему условию в следующей формулировке:

4') вычислительно невозможно любую пару значений $x' \neq x$, таких что $H(x')=H(x)$.

Любая сильная хэш-функция соответствует и требованиям для слабой, обратное в общем случае неверно. Для иллюстрации различия в сложности поиска коллизий слабой и сильной хэш-функции рассмотрим атаку с использованием «парадокса дней рождения»[1]. Зафиксируем значение аргумента x, и будем перебирать случайным образом $x' \neq x$ в поисках ситуации, когда $H(x')=H(x)$. Если предположить, что значения хэш-функции равномерно распределены, а число возможных значений $H(x)$ равно N, то потребуется *в среднем* перебор $N/2$ вариантов. Если же мы захотим найти какую-либо коллизию вообще, то задача оказывается проще: с вероятностью 0,63 для определения нужной пары значений потребуется опробовать $\sqrt{N}$ вариантов.

Чтобы минимизировать стоимость создания криптографических хэш-функций, разработчики часто используют один из существующих алгоритмов шифрования. Пусть $E(m,k)$ обозначает шифрование сообщения m на ключе k, а v_0 – стартовый вектор. Представим хэшируемое сообщение M в виде последовательности блоков $m_1, ..., m_t$ и будем их использовать в качестве раундовых ключей. Тогда $H(m)$ вычисляется следующим образом:

---

[1] Парадокс дней рождения – известный пример из теории вероятности – утверждение, что если дана группа из 23 или более человек, то вероятность того, что хотя бы у двух из них дни рождения (число и месяц) совпадут, превышает 1/2. Данное утверждение кажется противоречащим здравому смыслу, так как вероятность одному родиться в определенный день года довольно мала, а вероятность того, что двое родились в конкретный день – еще меньше.

$$h_0 = v_0,$$
$$h_i = E(h_{i-1}, m_i),\ i = 1 \ldots t\ , \qquad (2.27)$$
$$H(m) = h_t.$$

Однако в варианте с использованием в качестве $E(m,k)$ алгоритма DES, хэш-функция оказалась недостаточно стойкой из-за подверженности атаке, основанной на «парадоксе дней рождения». И было предложено улучшить эту схему, например, следующим образом:

$$h_0 = v_0,$$
$$h_i = E(h_{i-1}, m_i) \oplus h_{i-1},\ i=1 \ldots t, \qquad (2.28)$$
$$H(m) = h_t.$$

Существует и ряд специально разработанных алгоритмов хеширования, один из которых – SHA-1.

2.5.2. Алгоритм SHA-1

Алгоритм SHA (Secure Hash Algorithm) разработан в США как часть стандарта SHS (Secure Hash Standard), опубликованного в 1993 году. Но в нем были обнаружены уязвимости, которые привели к необходимости модифицировать алгоритм. Через два года была опубликована новая версия – SHA-1, получившая на сегодняшний день широкое распространение.

Получая на входе сообщение произвольной длины менее 2^{64} бит, SHA-1 формирует 160-битное выходное сообщение (дайджест). Вначале преобразуемое сообщение M дополняется до длины, кратной 512 битам. Заполнитель формируется следующим образом: в конец преобразуемого сообщения добавляется 1, потом – столько нулей, сколько необходимо для получения сообщения, которое на 64 бита короче, чем кратное 512, после чего добавляют 64-битное представление длины исходного сообщения. Например, если сообщение длиной 800 бит, то 801-й бит =1, потом добавляем нули до 960 бит, после чего – в оставшихся 64-разрядах записывается число «800», в итоге хэшируем 1024-битное сообщение. Общая схема преобразования представлена

на рис. 2.18. Перед началом преобразований инициализируется пять 32-битных переменных:

A=0x67452301;

B=0xEFCDAB89;

C=0x98BADCFE;

D=0x10325476;

E=0xC3D2E1F0.

Эти значения присваиваются также переменным a_0, b_0, c_0, d_0, e_0. Преобразование производится над блоком сообщения размером 512 бит в 80 раундов. В процессе преобразования используются следующие нелинейные функции f_t:

$f_t(X,Y,Z)=(X \wedge Y) \vee ((\neg X) \wedge Z)$ для t=0…19;

$f_t(X,Y,Z)=X \oplus Y \oplus Z$ для t=20…39 и t=60…79;

$f_t(X,Y,Z)=(X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$ для t=40…59.

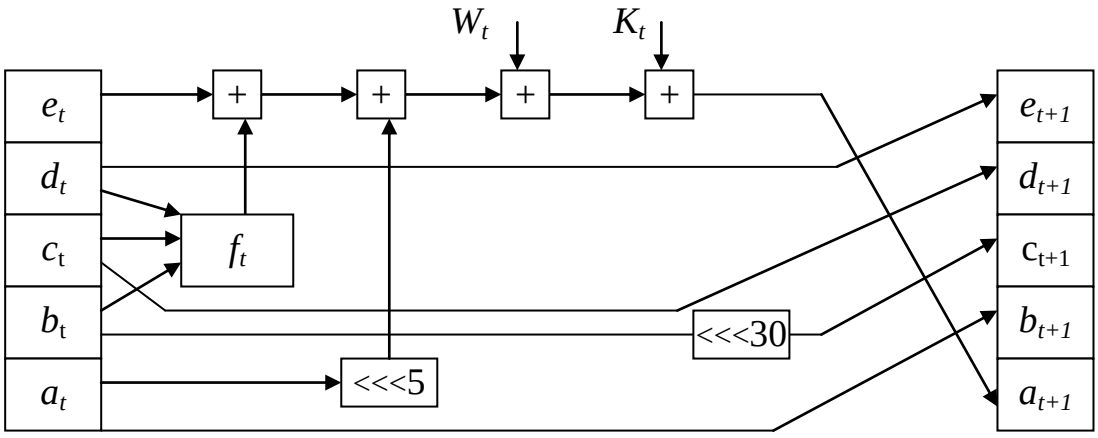


Рис. 2.18 Схема раунда алгоритма SHA-1

В процессе преобразования используются четыре константы:

K_t=0x5A827999 для t=0…19;

K_t=0x6ED9EBA1 для t=20…39;

K_t=0x8F1BBCDC для t=40…59;

K_t=0xCA62C1D6 для t=60…79.

Блок исходного сообщения M представляется в виде 16-ти 32-разрядных подблоков $M_0, \ldots, M_{15}$, которые используются для формирования значений W_t:

$W_t=M_t$ для t=0…15;

$W_t=(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16})<<<1$ для t=16…79.

Обозначение «<<< X» – циклический сдвиг влево на X разрядов, «+» – сложение по модулю 2^{32}.

После преобразования очередного 512-битного блока, полученные значения *a,b,c,d,e* складываются со значениями *A,B,C,D,E* соответственно, и начинается обработка следующего блока (или полученное значение в виде сцепления *a,b,c,d,e* подается на выход, если обработанный блок был последним).

Таким образом, на выходе получаем 160-битный дайджест исходного сообщения.

2.5.3. Хэш-функции с ключом

Хэш-функцией с ключом называется односторонняя функция *H(k,x)* со следующими свойствами:

- аргумент *x* функции *H(k,x)* может быть строкой бит произвольной длины;

- значение функции должно быть строкой бит фиксированной длины;

- при любых данных *k* и *x* легко вычислить *H(k,x)*;

- для любого *x* должно быть практически невозможно вычислить *H(k,x)*, не зная *k*;

- должно быть практически невозможно определить *k*, даже при большом числе известных пар *{x, H(k,x)}* или вычислить по этой информации *H(k,x')* для $x' \neq x$.

Часто такие функции также называются *кодами аутентификации сообщений* (англ. «Message Authentication Code», сокр. MAC). В отечественной литературе используется также термин *имитозащитная вставка* (или просто *имитовставка*).

Хэш-функцию с ключом можно построить на базе криптографической хэш-функции без ключа или алгоритма шифрования.

Пусть $H(x)$ – хэш-функция без ключа. Можно внедрить ключ в процесс хэширования, и получить хэш-функцию с ключом $H(k,x)$. Ниже представлены возможные варианты построения:

$$H(k,x) = H(k|x),$$
$$H(k,x) = H(x|k), \qquad (2.28)$$
$$H(k,x) = H(k_1|x|k_2), \text{ где } k=k_1|k_2.$$

Символ | в (2.28) обозначает конкатенацию, объединение строк аргументов.

Другой пример – построение хэш-функции с помощью шифра DES. Входной текст m разбивается на блоки $m_1,\ldots,m_t$ по 64 бита, которые преобразуются следующим образом (k – ключ шифрования):

$$c_0=0,$$
$$c_i=DES_k(m_i \oplus c_{i-1}),\ i=1,\ldots,t,$$
$$H(k,m)= c_t.$$

2.6. ИНФРАСТРУКТУРА ОТКРЫТЫХ КЛЮЧЕЙ. ЦИФРОВЫЕ СЕРТИФИКАТЫ

Использование методов асимметричной криптографии сделало возможным безопасный обмен криптографическими ключами между отправителем и получателем, которые никогда друг друга не встречали и, возможно, находятся за многие километры друг от друга.

Но возникает другая проблема – как убедиться в том, что имеющийся у Вас открытый ключ другого абонента на самом деле принадлежит ему. Иными словами, возникает проблема аутентификации ключа. Без этого, на криптографический протокол может быть осуществлена атака типа «человек посередине» (англ. «man in the middle»).

Идею данной атаки поясняет следующий пример. Абонент А (Алиса) хочет послать абоненту В (Боб) зашифрованное сообщение и

берет его отрытый ключ из общедоступного справочника. Но, на самом деле, ранее нарушитель Е (Ева) подменил в справочнике открытый ключ Боба своим открытым ключом. Теперь Ева может расшифровать те сообщения, которые Алиса формирует для Боба, ознакомиться с их содержимым, возможно, зашифровать их на настоящем ключе Боба и переслать ему (рис. 2.19).

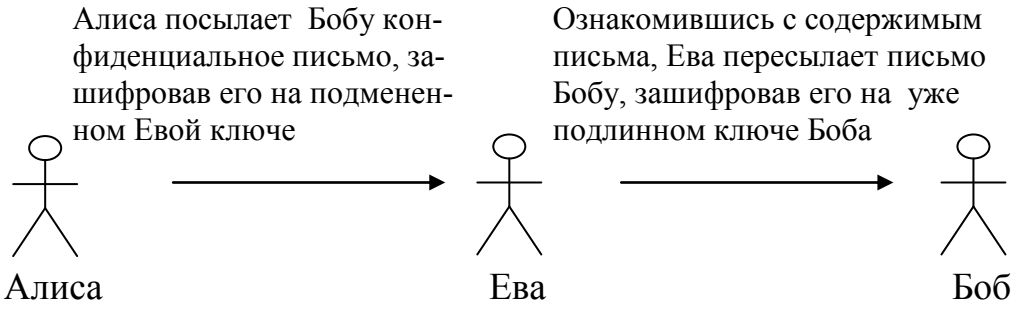


Рис. 2.19. Атака типа «man in the middle»

Избежать подобной атаки можно, подтвердив подлинность используемого ключа. Но Алиса и Боб лично никогда не встречались, и передать, например, дискету с ключом из рук в руки не могут. Поэтому, решение задачи подтверждения подлинности берет на себя третья доверенная сторона – некий арбитр, которому доверяют оба абонента. Заверяется ключ с помощью цифрового сертификата.

На самом деле, подобный способ применяется и вне компьютерных систем. Когда для подтверждения подлинности человека используется паспорт, то в роли третьей доверенной стороны выступает государство (от имени которого действовали в выдавшем паспорт отделе милиции).

Но вернемся к цифровым сертификатам. Для подтверждения подлинности открытых ключей создается инфраструктура открытых ключей (англ. «Public Key Infrastructure», сокр. PKI). PKI представляет собой набор средств, мер и правил, предназначенных для управления ключами, политикой безопасности и обменом защищенными сообщениями. Структура PKI представлена на рис. 2.20.

Для простоты последующего изложения, будем представлять сеть в виде совокупности удостоверяющих центров (другое название – центр сертификации, сокр. ЦС, от англ. «Certification Authority», сокр. – CA) и пользователей. Центр сертификации – абонент, которому доверено право удостоверять своей подписью сертификаты, связывающие открытые ключи абонентов с их идентификационной информацией. Сами центры сертификации тоже получают сертификаты своих ключей у центров более высокого уровня.

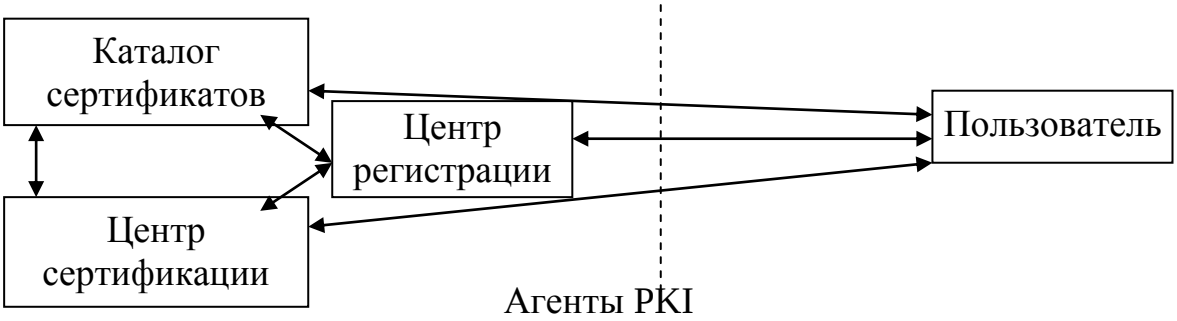


Рис. 2.20. Структура PKI

Таким образом, центры сертификации и пользователи формируют древовидную иерархическую структуру (рис. 2.21). В вершине этого дерева находится корневой центр сертификации, на рисунке – CA_1. Его особенность заключается в том, что он использует самоподписанный сертификат, т. е. сам заверяет свой ключ.

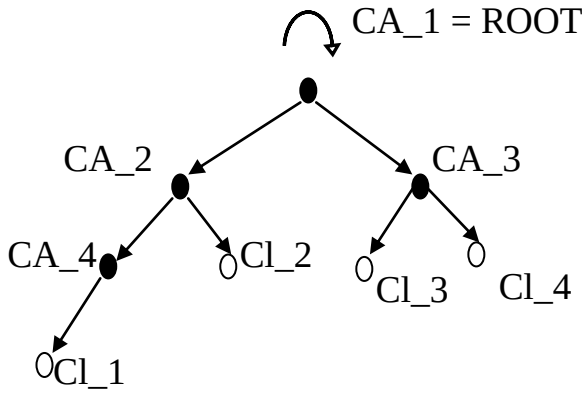


Рис. 2.21. Иерархия центров сертификации и клиентов

В приведенном примере, CA_1 заверяет электронной подписью сертификаты подчиненных центров сертификации CA_2 и CA_3, а те, в свою очередь, подписывают сертификаты пользователей и центров более низкого уровня.

Перейдем к рассмотрению самих сертификатов. Наибольшее распространение получили цифровые сертификаты, формат которых определен стандартом X.509. На данный момент, принята третья версия стандарта. Формат сертификата изображен на рис. 2.22 [11].

Номер версии содержит числовое значение, соответствующее номеру версии (для сертификата версии 1 равен 0 и т. д.). В первой версии X.509 не было уникальных номеров («ID Изготовителя», «ID Субъекта») и полей расширения. Во второй версии добавились указанные идентификаторы, в третьей – расширения.

Серийный номер – уникальный номер, присваиваемый каждому сертификату.

Алгоритм подписи – идентификатор алгоритма, используемого при подписании сертификата. Должен совпадать с полем *Алгоритм ЭЦП*.

Изготовитель - имя центра сертификации, выдавшего сертификат. Записывается в формате Relative Distinguished Name - RDN (варианты перевода названия – «относительное отдельное имя», «относительное характерное имя»). Данный формат используется в службах каталога, в частности, в протоколе LDAP. При записи Relative Distinguished Name используются специальные ключевые слова: CN (англ. «Common Name») – общее имя; OU (англ. «Organization Unit») – организационная единица; DC (англ. «Domain Component») – составная часть доменного имени.

Например, в сертификате Microsoft Windows Hardware Compatibility, который находится в хранилище сертификатов Windows'XP значение данного поля следующее:

CN = Microsoft Root Authority,

OU = Microsoft Corporation,

OU = Copyright (c) 1997 Microsoft Corp.

Как видно, указывается имя центра сертификации, компания, которой он принадлежит и т. д.

Рис. 2.22. Сертификат формата X.509 v.3

Субъект – имя владельца сертификата, представленное в том же формате RDN. Для указанного в предыдущем примере сертификата значения данного поля:

CN = Microsoft Windows Hardware Compatibility,

OU = Microsoft Corporation,

OU = Microsoft Windows Hardware Compatibility Intermediate CA,

OU = Copyright (c) 1997 Microsoft Corp.

Период действия – описывает временной интервал, в течение которого центр сертификации гарантирует отслеживание статуса сертификата (сообщит абонентам сети о факте досрочного отзыва сертификата и т. д.). Период задается датами начала и окончания действия.

Открытый ключ – составное поле, содержащее идентификатор алгоритма, для которого предназначается данный открытый ключ, и собственно сам открытый ключ в виде набора битов.

ID Изготовителя и *ID Субъекта* содержат уникальные идентификаторы центра сертификации и пользователя (на случай совпадения имен различных CA или пользователей).

Расширения – дополнительный атрибут, связанный с субъектом, изготовителем или открытым ключом, и предназначенный для управления процессами сертификации. Более подробно он описан ниже.

Алгоритм электронной цифровой подписи (ЭЦП) – идентификатор алгоритма, используемый для подписи сертификата. Должен совпадать со значением поля *Алгоритм подписи*.

ЭЦП – само значение электронно-цифровой подписи для данного сертификата.

Расширения могут определять следующие дополнительные параметры:

- идентификатор пары открытый/секретный ключ центра сертификации (изготовителя), если центр имеет несколько различных ключей для подписи сертификатов;

- идентификатор конкретного ключа пользователя (субъекта), если пользователь имеет несколько сертификатов;

- назначение ключа, например, ключ для шифрования данных, проверки ЭЦП данных, для проверки ЭЦП сертификатов и т. д.;

- уточнение периода использования – можно сократить время действия сертификата, указанное в поле *Период действия* (период, в течение которого статус сертификата отслеживается, станет больше, чем разрешенное время использования сертификата);

- политики использования сертификата;

- выбор соответствия политик использования сертификата для центра сертификации и пользователя, если имеются различные варианты;

- альтернативное имя пользователя и центра сертификации;

- указания, является ли пользователь сам центром сертификации и насколько глубоко разрешается разворачивать сертификационный путь.

Предположим, что ключевые пары сгенерированы, открытые ключи заверены сертификатами и размещены в каталоге, реализованном с помощью web-сервера, ftp-сервера, службы каталога или другой технологии. Теперь, если абонент *A* желает проверить подпись абонента *B* под полученным сообщением (или зашифровать для *B* сообщение с помощью его открытого ключа), он выполняет следующие действия [8]:

1) запрашивает в сетевом справочнике сертификат C_B открытого ключа подписи (шифрования) абонента *B*;

2) проверяет достоверность сертификата C_B (см. ниже);

3) в случае успеха проверяет подпись под сообщением (зашифровывает сообщение) с помощью открытого ключа, извлеченного из C_B.

Процедура проверки достоверности сертификата C_B состоит в следующем:

1) проверяется срок действия сертификата C_B, если он закончился, сертификат считается недостоверным;

2) из C_B извлекается имя ЦС, подписавшего этот сертификат, обозначим его D;

3) если $D=B$, то сертификат самоподписанный, он считается достоверным только, если $D=ROOT$ (хотя, возможно, в некоторых сетях право выдавать самоподписанные сертификаты имеет не один ROOT, это – политика сети);

4) если же $D\neq B$, то из справочника запрашивается сертификат C_D открытого ключа подписи абонента D, проверяется на достоверность сертификат C_D;

5) в случае отрицательного ответа принимается решение о недостоверности сертификата C_B, иначе из C_D извлекается открытый ключ K_D;

6) с помощью K_D проверяется подпись под сертификатом C_B, по результатам проверки этой подписи судят о достоверности C_B.

Если ключи шифрования досрочно вышли из обращения (причины могут быть различны – пользователь увольняется из компании, секретный ключ скомпрометирован и т. д.), центр сертификации извещает об этом остальных пользователей сети путем выпуска списка отозванных сертификатов (англ. «Certificate Revocation List», сокр. CRL). Структура CRL представлена на рис. 2.23. Поля списка содержат следующую информацию.

Номер версии определяет номер версии формата CRL. Текущая используемая версия – вторая.

Алгоритм подписи – идентификатор алгоритма, с помощью которого подписан CRL. Должен совпадать по значению с полем *Алгоритм ЭЦП*.

Изготовитель – имя центра сертификации в формате RDN.

Выпущен – дата выпуска CRL.

Следующий – дата, до которой будет выпущен следующий CRL.

Расширения – описывают центр сертификации, точку для поиска CRL данного центра, номер данного списка и т. д.

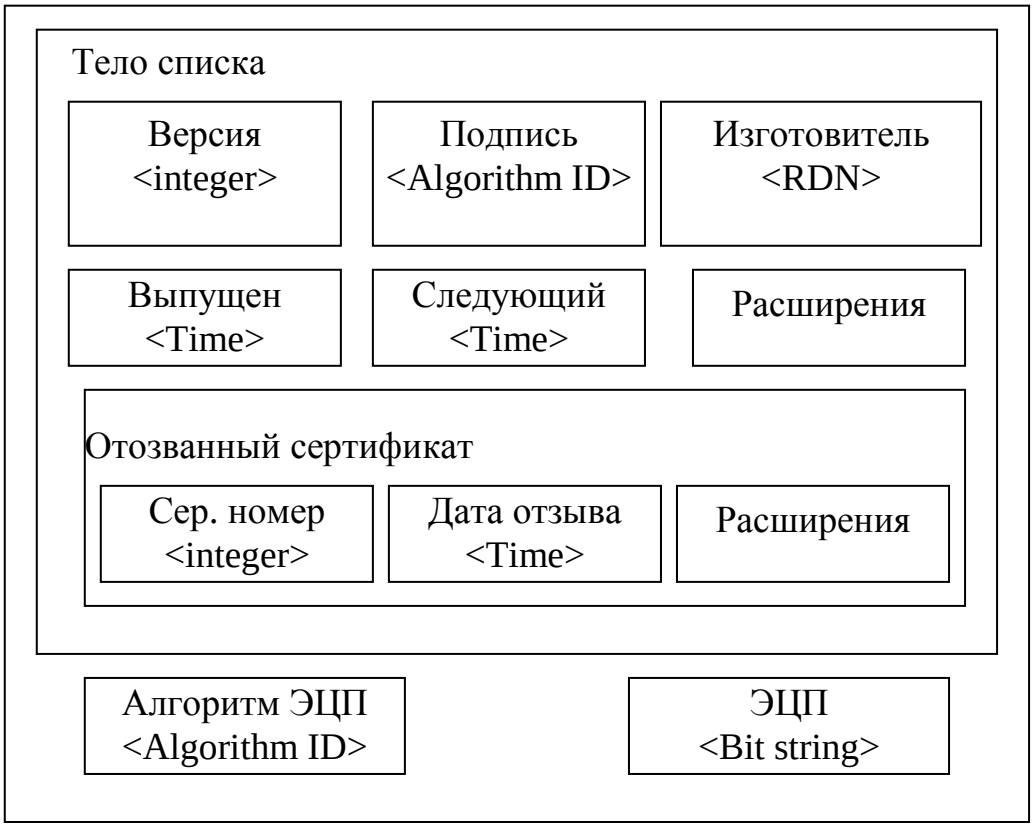


Рис. 2.23. Структура списка отозванных сертификатов

Отозванный сертификат – таких полей будет столько, сколько сертификатов отзывается – содержит номер отзываемого сертификата, дату, с которой сертификат отозван, описание причины отзыва.

Алгоритм ЭЦП – идентификатор алгоритма ЭЦП, используемого для подписи списка.

ЭЦП – сама электронная цифровая подпись.

Проблемы с CRL заключаются в том, что может возникнуть ситуация, когда ключ уже отозван, но CRL еще не выпущен, т. е. пользователи не могут получить информацию о компрометации ключа. Кроме того, распространение CRL идет по запросу клиента и нарушитель может препятствовать их получению.

Другая проблема PKI – самоподписанные сертификаты. Сертификат корневого ЦС должен раздаваться всем абонентам сети в начале работы и сохраняться в защищенном от подделки хранилище. Ина-

че нарушитель может попробовать навязать свой сертификат в качестве сертификата корневого центра.

Мы рассмотрели случай реализации *иерархической модели* PKI, при которой центры сертификации организованы в древовидную структуру с корневым центром сертификации на верху иерархии. На практике также встречаются другие варианты организации:

- *одиночный центр сертификации*, который выдает себе самоподписанный сертификат – данная модель часто реализуется в небольших организациях, но она имеет отмеченный выше недостаток, связанный с самоподписанными сертификатами;

- *одноранговая модель*, при которой независимые центры сертификации взаимно сертифицируют друг друга.

Надо отметить, что сфера применения цифровых сертификатов сейчас достаточно широка. В частности, они используются для распределения открытых ключей в протоколах защиты электронной почты S/MIME или PGP, с помощью цифровых сертификатов проверяется подлинность участников соединения по протоколу SSL и т. д.

3. ЗАЩИТА ИНФОРМАЦИИ В IP-СЕТЯХ

На сегодняшний день стек сетевых протоколов TCP/IP является наиболее широко используемым как в глобальных, так и в локальных компьютерных сетях. Именно поэтому методы и средства защиты передаваемых данных в IP-сетях представляют особый интерес.

В этой главе будут рассмотрены криптографические протоколы, позволяющие защищать электронную почту, передаваемые данные на транспортном и сетевом уровне. Кроме того, учитывая большую роль межсетевых экранов в решении задач обеспечения сетевой безопасности, в последнем разделе будет рассмотрен этот класс средств защиты.

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. Он был предложен в 1995 году компанией RSA Data Security Inc. Дальнейшие работы над протоколом велись рабочей группой организации Internet Engineering Task Force (IETF), разрабатывающей стандарты сети Интернет. На данный момент, последней является версия 3.1 этого протокола, описываемая в документах RFC 3850, 3851, 3852. Протокол S/MIME предоставляет следующие криптографические услуги безопасности (криптографические сервисы):

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

Нужно отметить, что сам по себе формат MIME описывает порядок форматирования писем, содержащих различные типы данных (обычный текст, текст в формате html, видео и графические файлы

различный типов и т. д.). При использовании S/MIME добавляются новые типы (например, application/pkcs7-mime и application/pkcs7-signature). Это позволяет указать на то, что данные в этом разделе являются зашифрованными, подписанными и т. д. 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, причем версия 3 стандарта рекомендует использовать именно последний алгоритм (из-за того, что он формирует более длинный дайджест и считается более надежным). 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. На рис. 3.1 приведен фрагмент письма, содержащий открытый текст «This is a clear-signed message.» и подпись к нему.

S/MIME поддерживается многими почтовыми клиентами: Microsoft Outlook, Mozilla, The Bat! и т. д. Более широкое применение протокола сдерживается необходимостью наличия сертификатов у абонентов и плохой совместимостью с системами Web-почты.

```
Content-Type: multipart/signed;
          protocol="application/pkcs7-signature";
          micalg=sha1; boundary=boundary42

        --boundary42
        Content-Type: text/plain

        This is a clear-signed message.

        --boundary42
        Content-Type: application/pkcs7-signature; name=smime.p7s
        Content-Transfer-Encoding: base64
        Content-Disposition: attachment; filename=smime.p7s

        ghyHhHUujhJhjH77n8HHGTrfvbnj756tbB9HG4VQpfyF467GhIGfHfYT6
        4VQpfyF467GhIGfHfYT6jH77n8HHGghyHhHUujhJh756tbB9HGTrfvbnj
        n8HHGTrfvhJhjH776tbB9HG4VQbnj7567GhIGfHfYT6ghyHhHUujpfyF4
        7GhIGfHfYT64VQbnj756

        --boundary42--
```

Рис. 3.1. Фрагмент электронного письма с подписью

Альтернативой S/MIME является PGP (англ. «Pretty Good Privacy») – компьютерная программа созданная Филиппом Циммерманном (Philip Zimmermann) в 1991 году. Данная программа положила основу работе над стандартом OpenPGP, последняя версия которого описана в RFC 4880. По функциональности S/MIME и PGP во многом сходны.

3.2. ПРОТОКОЛЫ SSL И TLS

Протокол Secure Sockets Layer (SSL) был разработан корпорацией Netscape Communications для обеспечения аутентификации, целостности и секретности трафика на сеансовом уровне модели OSI (с точки зрения четырехуровневой модели стека протоколов TCP/IP – на прикладном уровне). В январе 1999 года на смену SSL v3.0 пришел протокол TLS v1.0 (Transport Layer Security) последняя версия TLS v.1.2 описывается в RFC 5246. С точки зрения выполняемых действий, различия между этими протоколами SSL и TLS весьма невелики, в то же время, они несовместимы друг с другом [11].

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», от англ. «handshake») 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 как 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.

Рассмотрим более подробно процедуру рукопожатия в режиме аутентификации сервера без аутентификации клиента. Она включает следующие шаги [13].

1. Клиент посылает серверу запрос на установление защищенного соединения, в котором передает, в частности, следующие параметры:

- дату и текущее время;

- сгенерированную клиентом случайную последовательность (RAND_CL);

- перечень поддерживаемых клиентом алгоритмов шифрования, хеширования и сжатия (если сжатие используется).

2. Сервер обрабатывает запрос от клиента и передает ему согласованный набор параметров:

- идентификатор SSL-сессии;

- идентификаторы криптографических алгоритмов, из числа предложенных клиентом, которые будут использоваться в данной сессии (если по какой-либо причине предложенные алгоритмы или их параметры не удовлетворяют требованиям сервера, сессия закрывается);

- цифровой сертификат сервера формата X.509;

- случайную последовательность (RAND_SERV).

3. Клиент проверяет полученный сертификат и соответствие роли ключа его назначению, описанному в сертификате. При отрицательном результате проверки сессия закрывается, а при положительном – клиент выполняет следующие действия:

- генерирует случайную 48-байтную последовательность, называемую Pre_MasterSecret, предназначенную для расчета общего секретного ключа;

- шифрует значение Pre_MasterSecret с использованием открытого ключа сервера, взятого из сертификата, и посылает криптограмму серверу;

- с помощью согласованной с сервером хэш-функции формирует общий секретный ключ (MasterSecret), используя в качестве параметров последовательность Pre_MasterSecret, посланную ранее серверу случайную последовательность RAND_CL и полученную от него случайную последовательность RAND_SERV;

- используя значение MasterSecret, вычисляет криптографические параметры SSL-сессии: формирует общие с сервером сеансовые секретные ключи для симметричного шифрования и вычисления хэш-функций;

- переходит в режим защищенного взаимодействия.

4. Сервер, используя свой секретный ключ, расшифровывает полученное значение Pre_MasterSecret и выполняет те же операции, что и клиент:

- с помощью согласованной с клиентом хэш-функции формирует общий секретный мастер-ключ (MasterSecret), используя в качестве параметров значение Pre_MasterSecret, а также посланную клиенту случайную последовательность RAND_SERV и полученную от него случайную последовательность RAND_CL;

- используя значение MasterSecret, вычисляет криптографические параметры SSL-сессии: формирует общие с клиентом сеансовые секретные ключи для симметричного шифрования и вычисления хэш-функций;

- переходит в режим защищенного взаимодействия.

Так как при формировании параметров SSL-сессии и клиент, и сервер пользовались одними и теми же исходными данными (согласованными алгоритмами, общей секретной последовательностью Pre_MasterSecret и случайными последовательностями RAND_CL и RAND_SERV), то очевидно, что в результате описанных выше действий они выработали одинаковые сеансовые секретные ключи. Для

проверки идентичности параметров SSL-сессии клиент и сервер посылают друг другу тестовые сообщения, содержание которых известно каждой из сторон:

- клиент формирует сообщение из собственных посылок в адрес сервера на шаге 1 и посылок, полученных от сервера на шаге 2, внося элемент случайности в виде последовательности MasterSecret, уникальной для данной сессии; формирует код для проверки целостности сообщения (MAC), шифрует сообщение на общем сеансовом секретном ключе и отправляет серверу;

- сервер, в свою очередь, формирует сообщение из собственных посылок в адрес клиента на шаге 2, посылок, полученных от клиента на шаге 1, и последовательности MasterSecret; формирует код для проверки целостности сообщения (MAC), шифрует сообщение на общем сеансовом секретном ключе и отправляет клиенту;

- в случае успешной расшифровки и проверки каждой из сторон целостности полученных тестовых сообщений, SSL-сессия считается установленной, и стороны переходят в штатный режим защищенного взаимодействия.

Необязательная вторая фаза рукопожатия позволяет аутентифицировать клиента. Она заключается в том, что сервер шлет запрос клиенту, клиент аутентифицирует себя, возвращая подписанное сообщение (запрос сервера) и свой цифровой сертификат.

В процессе защищенного взаимодействия с установленными криптографическими параметрами SSL-сессии выполняются следующие действия:

- каждая сторона при передаче сообщения формирует имитовставку (MAC) для последующей проверки целостности сообщения и затем зашифровывает исходное сообщение вместе с MAC по сеансовому секретному ключу;

- каждая сторона при приеме сообщения расшифровывает его и проверяет на целостность (вычисляется текущее значение MAC и сверяется со значением, полученным вместе с сообщением); в случае

обнаружения нарушения целостности сообщения, SSL-сессия закрывается.

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).

3.3. ПРОТОКОЛЫ IPSEC И РАСПРЕДЕЛЕНИЕ КЛЮЧЕЙ

Протокол IPSec или, если точнее, набор протоколов, разработан организацией IETF как базовый протокол обеспечения безопасности на уровне IP-соединения. Он является дополнением к использующемуся сейчас 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 (рис. 3.2), 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. Иными словами, на работу приложений (таких как web-сервер, браузер, СУБД и т. д.) не влияет, используется ли защита передаваемых данных с помощью IPSec или нет.

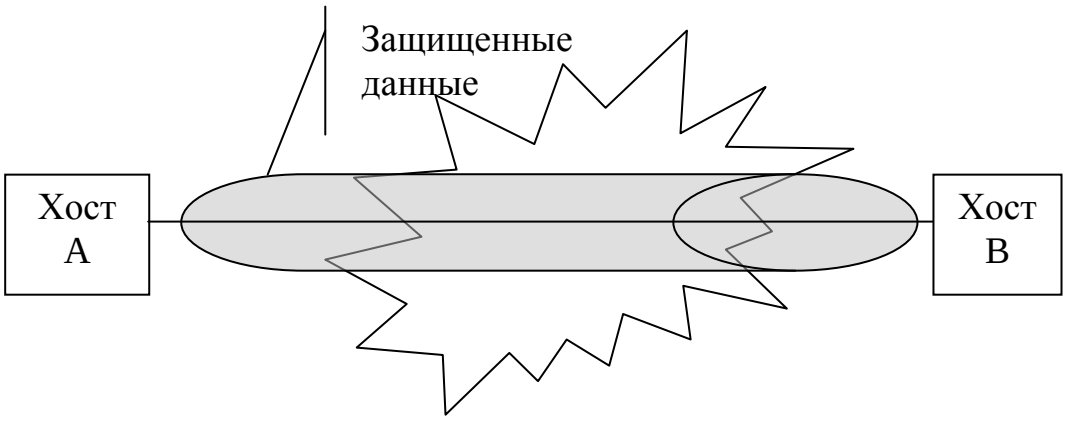


Рис. 3.2. Туннель безопасности

Архитектура IPSec является открытой, что позволяет использовать для защиты передаваемых данных новые криптографические алгоритмы, например, соответствующие национальным стандартам. Для этого необходимо, чтобы взаимодействующие стороны поддерживали эти алгоритмы, и они были бы стандартным образом зарегистрированы в описании параметров соединения.

Процесс защищенной передачи данных регулируется правилами безопасности, принятыми в системе. Параметры создаваемого туннеля описывает информационная структура, называемая контекст защиты или ассоциация безопасности (от англ. «Security Association», сокр. SA). Как уже отмечалось выше, IPSec является набором протоколов, и состав контекста защиты может различаться. В зависимости от конкретного протокола в него входит:

- IP-адрес получателя;

- указание на протоколы безопасности, используемые при передаче данных;

- ключи, необходимые для шифрования и формирования имитовставки (если это требуется);

- указание на метод форматирования, определяющий, каким образом создаются заголовки;

- индекс параметров защиты (от англ. «Security Parameter Index», сокр. SPI) – идентификатор, позволяющий найти нужный SA.

Обычно, контекст защиты является однонаправленным, а для передачи данных по туннелю в обе стороны задействуются два SA. Каждый хост имеет свою базу контекстов защиты, из которой выбирается нужный элемент либо на основании значения SPI, либо по IP-адресу получателя.

Два протокола, входящие в состав IPSec это:

1) протокол аутентифицирующего заголовка – AH (от англ. «Authentication Header»), обеспечивающий проверку целостности и аутентификацию передаваемых данных; последняя версия протокола описана в документе RFC 4302 (предыдущие – RFC 1826, 2402);

2) протокол инкапсулирующей защиты данных – ESP (от англ. «Encapsulating Security Payload»), обеспечивающий конфиденциальность и, опционально, проверку целостности и аутентификацию; описан в RFC 4303 (предыдущие версии – RFC 1827, 2406).

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. Более подробно режимы работы описаны ниже.

3.3.1. Протокол AH

В IP ver.4 аутентифицирующий заголовок располагается после IP-заголовка. Представим исходный IP-пакет как совокупность IP-заголовка, заголовка протокола следующего уровня (как правило, это

TCP или UDP, на рис. 3.3 он обозначен как ULP – от англ. «Upper-Level Protocol») и данных.

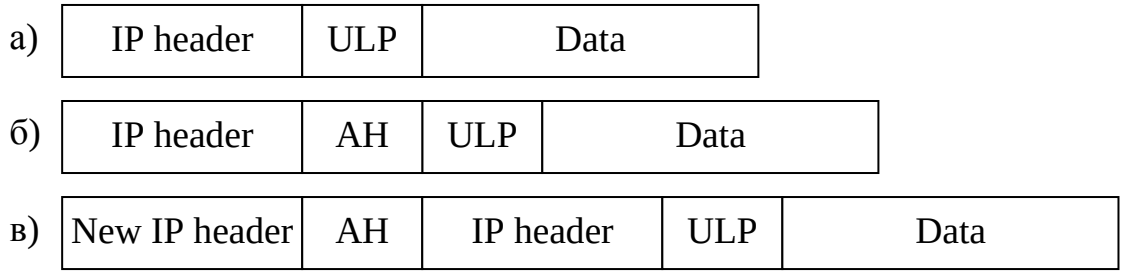


Рис. 3.3. а) исходный IP-пакет
б) IP-пакет при использовании AH в транспортном режиме
в) IP-пакет при использовании AH в туннельном режиме

На рисунке 3.3. представлен исходный пакет и варианты его изменения при использовании протокола AH в разных режимах. В транспортном режиме заголовок исходного IP-пакета остается на своем месте, но в нем модифицируются некоторые поля. Например, меняется поле Next Header, указывающее на то, заголовок какого протокола следует за IP-заголовком. В туннельном режиме создается новый IP-заголовок, после которого идет заголовок AH, а за ним – полностью исходный IP-пакет.

Аутентификация производится путем создания имитовставки (MAC) для чего используется хэш-функция и секретный ключ. Во всех реализациях AH обязательно должно поддерживаться использование алгоритмов HMAC-MD5-96 (используется по умолчанию) и HMAC-SHA-1-96, представляющих собой хэш-функции с ключом, основанные на хэш-функциях MD5 и SHA-1, соответственно. Но могут использоваться и другие, «факультативные» алгоритмы хэширования. Полученное значение, называемое в описании протокола ICV (от англ. «Integrity Check Value» – значение контроля целостности) помещается в поле *Authentication Data* (рис. 3.4). Это поле перемен-

ной длины, так как разные алгоритмы хеширования формируют разные по длине дайджесты.

При использовании AH в транспортном режиме, ICV рассчитывается для ULP, данных и неизменяемых полей IP-заголовка. Изменяемые поля, такие как поле TTL, указывающее на время жизни пакета и изменяемое при прохождении маршрутизаторов, при расчете значения хэш-функции принимаются равными 0. В туннельном режиме аутентифицируется весь исходный IP-пакет и неизменяемые поля нового заголовка. Рассмотрим формат заголовка AH (рис. 3.4).

Первые 8 бит заголовка (поле *Next Header*) содержат номер, соответствующий протоколу следующего уровня. Номер для каждого протокола назначает организация IANA (Internet Assigned Numbers Authority). Например, номер TCP – 6, ESP – 50, AH – 51 и т. д.

Поле *Payload Len* указывает длину заголовка AH в 32-битных словах. Далее 16 бит зарезервировано.

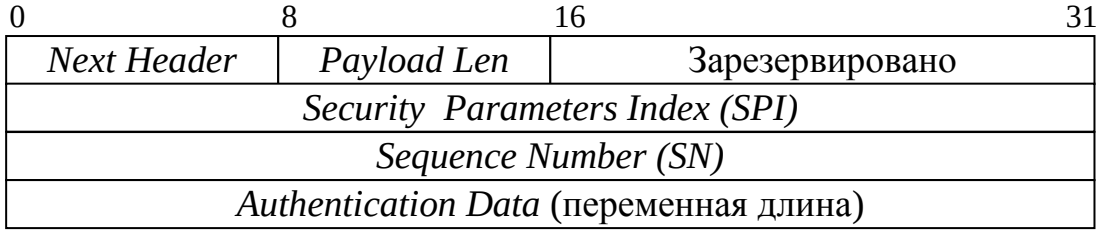


Рис. 3.4. Структура заголовка протокола AH

Поле *SPI* содержит значение индекса параметров защиты, по которому получатель сможет найти нужный контекст защиты (SA).

Поле *Sequence Number* было введено в RFC 2402. Значение счетчика, содержащееся в этом поле, может использоваться для защиты от атак путем повторных посылок перехваченных пакетов. Если функция защиты от повторов активирована (а это указывается в SA), отправитель последовательно наращивает значение поля для каждого пакета, передаваемого в рамках данной ассоциации (соединения, использующего единый SA).

Поле *Authentication Data*, как уже указывалось ранее, хранит значение ICV.

3.3.2. Протокол ESP

Если AH обеспечивает защиту от угроз целостности передаваемых данных, то ESP также может обеспечивать и конфиденциальность.

Так же как и AH, ESP может работать в транспортном и туннельном режимах. На рис. 3.5 изображены варианты его использования (штриховкой выделены фрагменты пакета, которые защищаются с помощью шифрования). Для ESP определен следующий перечень обязательных алгоритмов, которые должны поддерживаться во всех реализациях:

- для формирования имитовставки HMAC-MD5-96 (используется по умолчанию) и HMAC-SHA-1-96;

- для шифрования – DES (в режиме CBC; используется по умолчанию) и NULL (отсутствие шифрования).

Кроме того, зарегистрированы и могут быть реализованы еще ряд алгоритмов шифрования – Triple DES, CAST-128, RC5, IDEA, Blowfish, ARCFour (общедоступная версия RC4) [13].

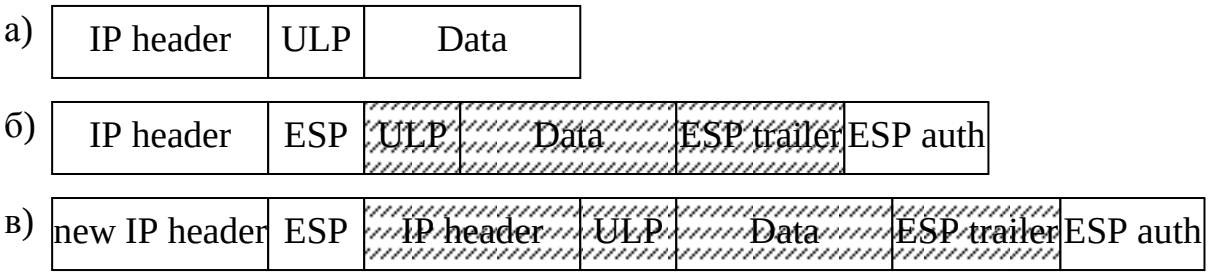


Рис. 3.5. а) исходный IP-пакет,
б) ESP в транспортном режиме,
в) ESP в туннельном режиме

Рассмотрим формат заголовка ESP (рис. 3.6). Он начинается с двух 32-разрядных значений – *SPI* и *SN*. Роль их такая же, как в протоколе AH – *SPI* идентифицирует контекст защиты, использующийся для создания данного туннеля; *SN* – позволяет защититься от повторов пакетов. *SN* и *SPI* не шифруются. Следующим идет поле, содержащее зашифрованные данные. После них - поле заполнителя, кото-

рый нужен для того, чтобы выровнять длину шифруемых полей до значения кратного размеру блока алгоритма шифрования.

После заполнителя идут поля, содержащие значение длины заполнителя и указание на протокол более высокого уровня. Четыре перечисленных поля (данные, заполнитель, длина, следующий протокол) защищаются шифрованием.

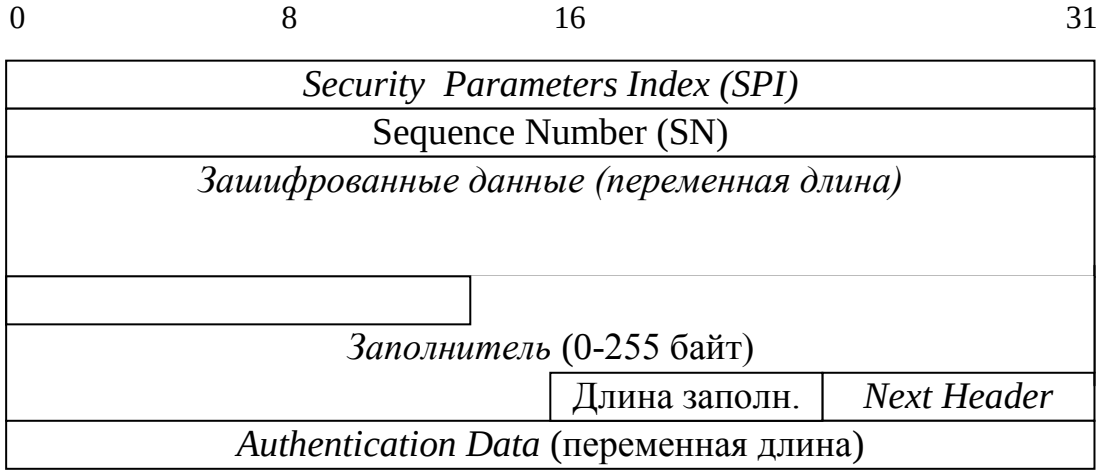


Рис. 3.6. Структура заголовка ESP

Если ESP используется и для аутентификации данных, то завершает пакет поле переменной длины, содержащее ICV. В отличие от AH, в ESP при расчете значения имитовставки, поля IP-заголовка (нового – для туннельного режима, модифицированного старого – для транспортного) не учитываются.

При совместном использовании протоколов AH и ESP, после IP-заголовка идет AH, после него – ESP. В этом случае, ESP решает задачи обеспечения конфиденциальности, AH – обеспечения целостности и аутентификации источника соединения.

Рассмотрим ряд дополнительных вопросов, связанных с использованием IPSec. Начнем с того, откуда берется информация о параметрах соединения – контекстах защиты или SA. Создание базы SA может производиться различными путями. В частности, она может создаваться администратором безопасности вручную, или формиро-

ваться с использованием специальных протоколов – SKIP, ISAKMP (Internet Security Association and Key Management Protocol) и IKE (Internet Key Exchange).

3.3.3. Протокол SKIP

Протокол SKIP (Simple Key management for Internet Protocol) был разработан корпорацией SUN Microsystems в 1994 году. Он создавался как IP-совместимый протокол, обеспечивающий управление ключами и криптозащиту передаваемых данных на сетевом уровне модели OSI [13]. По нумерации IANA этому протоколу присвоен номер 57.

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.

Отправитель генерирует временный ключ K_p, используемый для шифрования данных одного пакета или небольшой их группы. На нем зашифровывается исходный IP-пакет и инкапсулируется в SKIP-пакет (рис. 3.7). Шифрование данных на временном ключе, а не на общем секретном ключе K_{ij}, повышает надежность, так как в этом случае нарушителю труднее набрать нужный для реализации атаки на K_{ij} объем шифрованных данных.

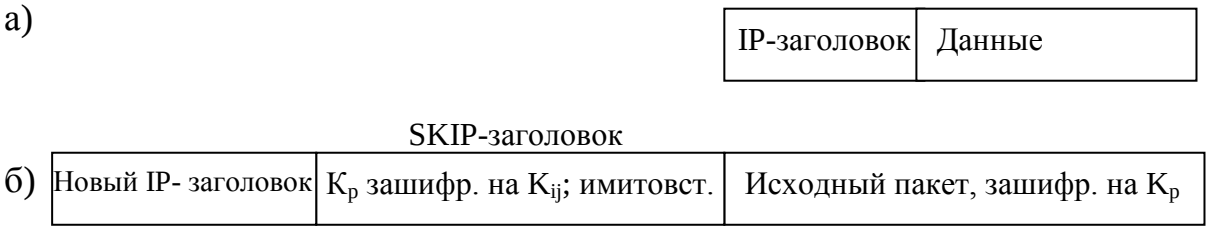


Рис. 3.7. Пакет до а) и после б) применения протокола SKIP.

Ключ K_p зашифровывается на ключе K_{ij} и криптограмма помещается в SKIP-заголовок, там же резервируется место под имитовставку.

Формируется новый IP-заголовок и с помощью хэш-функции с ключом для всего пакета рассчитывается значение имитовставки, которое помещается в SKIP-заголовок.

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. Рассмотрим теперь эту версию [14].

Итак, стороны распределили общий секретный ключ K_{ij} и отправитель сгенерировал временный ключ K_p. На его основе с помощью хэш-функции H будут выработаны ключ для шифрования пакета E_K_p и ключ для аутентификации A_K_p. В отличие от первоначального варианта, при передаче ключ K_p шифруется не на K_{ij}, а на модифицированном при помощи счетчика n и хеш-фукнции ключе K_{ijn}. Расчет производится в соответствии с формулой 3.1.

$$\begin{aligned} &E\_Kp=H(K_p|\text{Crypt Alg}|02h)|H(K_p|\text{Crypt Alg}|00h),\\ &A\_Kp= H(K_p|\text{MAC Alg}|03h)|H(K_p|\text{MAC Alg}|01h),\\ &K_{ijn}= H(K'_{ij}|n|01h)|H(K'_{ij}|n|00h), \end{aligned} \qquad (3.1)$$

где K'_{ij} – младшие 256 бит K_{ij}, Crypt Alg и MAC Alg – значения соответствующих полей заголовка SKIP (рис. 3.8).

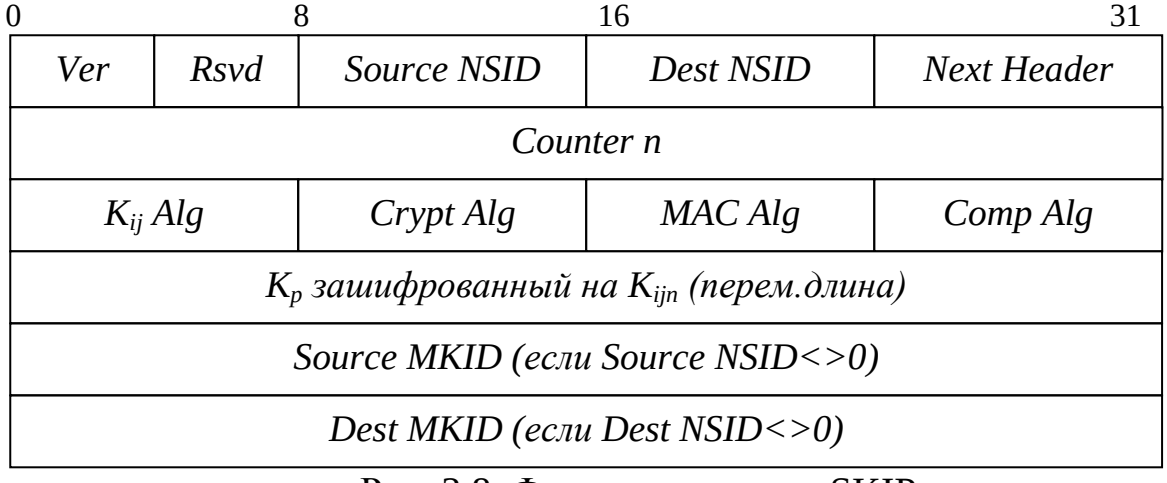


Рис. 3.8. Формат заголовка SKIP

Рассмотрим поля заголовка (рис. 3.8). *Ver* – номер версии протокола. Следующие за ним 4 бита зарезервированы (*Rsvd*). Далее – идентификаторы пространств имен источника и получателя *Source NSID и Dest NSID*. Если они равны 0, то в полях *Source MKID* и *Dest MKID* ставятся IP-адреса источника и получателя соответственно. После поля *Dest NSID* идет поле *Next Header*, содержащее номер протокола, следующего за SKIP. Далее идет 32-разрядное поле счетчика *Counter n*. Как отмечается в описаниях, правила для работы со счетчиком n отнесены на усмотрение разработчика, но для обеспечения совместимости версий предлагается считать, что *n* – время в часах, отсчитанное от 00:00 01.01.95. Как правило, если значение счетчика *n* пришедшего пакета отличается более чем на 1 от текущего, то пакет отбрасывается.

Далее в заголовке идут байтовые идентификаторы алгоритмов: шифрования ключа K_p – K_{ij} *Alg*, шифрования данных в пакете – *Crypt Alg*, аутентификации данных – *MAC Alg*, сжатия (если используется) – *Comp Alg*. После идентификаторов в SKIP-заголовок помещается ключ K_p, зашифрованный на ключе K_{ijn} (размер этого поля зависит от используемых алгоритмов шифрования ключа и данных). Далее идут идентификаторы отправителя и получателя в выбранном пространстве имен – *Source MKID* и *Dest MKID*. Наличие нескольких идентификаторов позволяет более гибко настраивать использование протоколов безопасности. Например, если на одном компьютере работают различные приложения, можно описать политику, указывающую какие алгоритмы и ключи использовать для защиты данных каждого из них.

В случае совместного использования протоколов SKIP и ESP, заголовок SKIP размещается после IP-заголовка перед заголовком ESP (рис. 3.9).

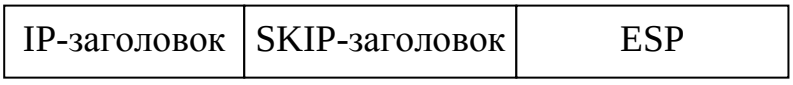


Рис. 3.9. Совместное использование SKIP и ESP

В этом случае, протокол ESP использует параметры соединения, определяемые протоколом SKIP, а указывает на это значение SPI в заголовке ESP: SKIP_SPI=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 [13]. 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.

Протокол ISAKMP определяет две фазы согласования параметров взаимодействия:

- согласование глобальных параметров защищенного канала (в терминах IPSec такие параметры называются управляющим контекстом);

- согласование параметров каждого защищенного соединения (они образуют контекст защиты – SA).

С точки зрения модели стека TCP/IP, протокол ISAKMP является протоколом прикладного уровня. В случае его использования совместно с IPSec, спецификация устанавливает использование в качестве протокола нижнего уровня UDP с номером порта 500.

Протокол ISAKMP определяет следующую последовательность действий по формированию управляющего контекста (стороны взаимодействия, например, это могут быть два шлюза безопасности, называются *«Инициатор»* и *«Партнер»*):

1) *«Инициатор»* → *«Партнер»*: заявка на контекст, включающая предлагаемые алгоритмы и их параметры;

2) *«Партнер»* → *«Инициатор»*: принимаемые алгоритмы и параметры (из списка, полученного на шаге 1; для каждой функции защиты – генерация и распределение ключей, шифрование, аутентификация – используется один алгоритм и его параметры);

3) *«Инициатор»* → *«Партнер»*: ключевой материал и одноразовый номер инициатора;

4) *«Партнер»* → *«Инициатор»*: ключевой материал и одноразовый номер партнера;

5) *«Инициатор»* → *«Партнер»*: подписанное инициатором зашифрованное сообщение, содержащее его идентификатор;

6) *«Партнер»* → *«Инициатор»*: подписанное партнером зашифрованное сообщение, содержащее его идентификатор.

Если используется протокол OAKLEY, на шаге 3 и 4 стороны отправляют друг другу свои открытые ключи вместе со случайными числами, служащими для защиты от повтора. Для обеспечения контроля подлинности открытых ключей могут быть использованы сертификаты X.509. В соответствии со схемой Диффи-Хеллмана, рассчитывается общий секретный ключ. На его основе вырабатываются значения:

SKEYID_d – ключевой материал, используемый для генерации временных ключей для защищенных соединений;

SKEYID_a – сеансовые ключи, используемые для аутентификации сторон и согласовываемых параметров;

SKEYID_e – сеансовые ключи, используемые для шифрования согласовываемых параметров.

Пятый и шестой шаги служат для обмена идентификационной информацией, защищенной и заверенной на ключах *SKEYID_e* и *SKEYID_a*.

Такой порядок взаимодействия реализуется при использовании основного или базового режима (англ. «Main Mode»). Он более медленный, но и более безопасный. Существует также «агрессивный» режим (англ. «Aggressive Mode») при котором, для увеличения скорости взаимодействия ряд параметров передается в открытом виде и уменьшено число шагов взаимодействия (с 6 до 3 за счет того, что на первом и втором шаге сразу передается больше параметров) [11].

Сообщение ISAKMP состоит из заголовка и следующих за ним полей данных. Формат заголовка приведен на рис. 3.10.

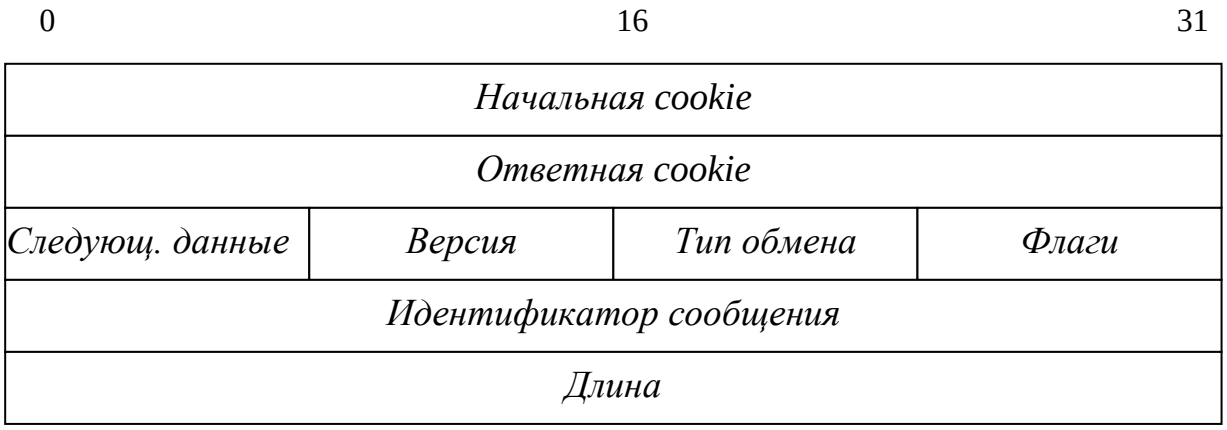


Рис. 3.10. Заголовок ISAKMP

Исходя из значения текущего времени, стороны формируют идентификаторы (*cookie*), которые включают в заголовок пакета (на шаге 1 присутствует только идентификатор стороны-инициатора соединения, на последующих – идентификаторы обеих сторон). Присутствие этих идентификаторов позволяет защититься от атак путем повторных посылок перехваченных сообщений.

Поле «*Следующие данные*» содержит идентификатор, указывающий на тип содержимого области данных. Например, 1 – контекст

защиты (SA), 2 – предложение (используется при согласовании параметров); 6 – сертификат, 7 – запрос сертификата и т. д.

Тип обмена – указывает на тип информационного обмена (режим, в котором работает протокол). Например, 0 – нет обмена, 1 – базовый, 4 – агрессивный и т. д.

Флаги позволяют указать дополнительные настройки. Например, один из флагов указывает, на то, что все данные, идущие за заголовком, зашифрованы.

Идентификатор сообщения – уникальный идентификатор сообщения.

Длина – длина сообщения (заголовка и данных).

Итак, описанная фаза протокола позволяет сформировать общий защищенный канал и согласовать его параметры.

После создания общего защищенного канала, параметры каждого защищенного соединения, создаваемого в рамках этого канала, согласуются на основе сформированных глобальных параметров канала и образуют контекст защиты. Каждое защищенное соединение является однонаправленным и в нем может использоваться один из двух протоколов – ESP или AH (кроме того, на базе общего защищенного канала могут быть созданы защищенные соединения, использующие отличный от IPSec протокол). Если предполагается организовать защищаемое ESP двустороннее взаимодействие, понадобятся два соединения (и два SA), если использовать и протокол AH, и ESP, то нужны четыре SA.

В состав согласуемых параметров, образующих SA, входят [13]:

- номер протокола криптозащиты (AH, ESP, другой);
- номера алгоритмов криптозащиты и их параметры;
- режим протокола (транспортный или туннельный);
- сеансовые ключи (действующие для текущего соединения);
- срок существования защищенного соединения (может задаваться временем или объемом переданного трафика; например, срок

существования канала - 6 часов, соединения в рамках этого канала – 1 час);

- максимальный размер пакетов;

- дополнительные параметры (параметры счетчика и т. д.) для защиты от повтора, задержки, удаления пакетов сообщения.

Пользователями защищенных соединений, как правило, являются прикладные процессы. И между двумя узлами сети может существовать произвольное число соединений, сформированных в рамках одного защищенного канала.

Защищенное соединение, соответствующее спецификациям протокола IPSec, идентифицируется целевым IP-адресом, используемым протоколом криптозащиты (ESP или AH) и индексом SPI.

Для выработки параметров «*Инициатор*» и «*Партнер*» обмениваются следующими сообщениями.

1) «*Инициатор*» → «*Партнер*» (защищенное сообщение):

- заявка на создание защищенного соединения (предлагаемые алгоритмы и их параметры);

- одноразовый номер инициатора.

2) «*Партнер*» → «*Инициатор*» (защищенное сообщение):

- принимаемые алгоритмы и параметры;

- одноразовый номер партнера.

3) «*Инициатор*» → «*Партнер*» (защищенное сообщение):

- одноразовый номер инициатора;

- одноразовый номер партнера.

Для защиты передаваемых сообщений используются ключи, выработанные при формировании защищенного канала: *SKEYID_a* – для аутентификации, *SKEYID_e* – для шифрования. Для аутентификации используется хэш-функция, в качестве аргументов которой выступают аутентифицируемое сообщение и ключ *SKEYID_a*.

Временные ключи защищенного соединения генерируются путем применения хэш-функции к значению *SKEYID_d* с дополнитель-

ными параметрами, в число которых входят одноразовые идентификаторы инициатора и партнера.

Принимающая сторона соединения задает для формируемого SA номер SPI, по которому он будет идентифицироваться.

3.3.5. Протоколы IPSec и трансляция сетевых адресов

При подключении сетей организаций к Интернет, часто используется механизм трансляции сетевых адресов – NAT (от англ. «Network Address Translation»). Это позволяет уменьшить число зарегистрированных IP-адресов, используемых в данной сети. Внутри сети используются незарегистрированные «частные» адреса (как правило, из диапазонов, специально выделенных для этой цели, например, адреса вида 192.168.x.y для сетей класса C). Если пакет из такой сети передается в Интернет, то маршрутизатор, внешнему интерфейсу которого назначен по крайней мере один зарегистрированный ip-адрес, модифицирует ip-заголовки сетевых пакетов, подставляя вместо частных адресов зарегистрированный адрес. Порядок, по которому производится подстановка, описывается в специальной таблице. При получении ответа, в соответствии с таблицей делается обратная замена, и пакет переправляется во внутреннюю сеть.

Рассмотрим пример использования NAT рис. 3.11. В данном случае, во внутренней сети используются частные адреса 192.168.0.x. С компьютера, с адресом 192.168.0.2 обращаются во внешнюю сеть к компьютеру с адресом 195.242.2.2. Пусть это будет подключение к web-серверу (протокол HTTP, который использует TCP порт 80).

При прохождении пакета через маршрутизатор, выполняющий трансляцию адресов, ip-адрес отправителя (192.168.0.2) будет заменен на адрес внешнего интерфейса маршрутизатора (195.201.82.146), а в таблицу трансляции адресов будет добавлена запись, аналогичная приведенной в таблице 3.1.

Получив представление о механизме работы NAT, разберемся, какие сложности могут возникнуть, в случае использования IPSec.

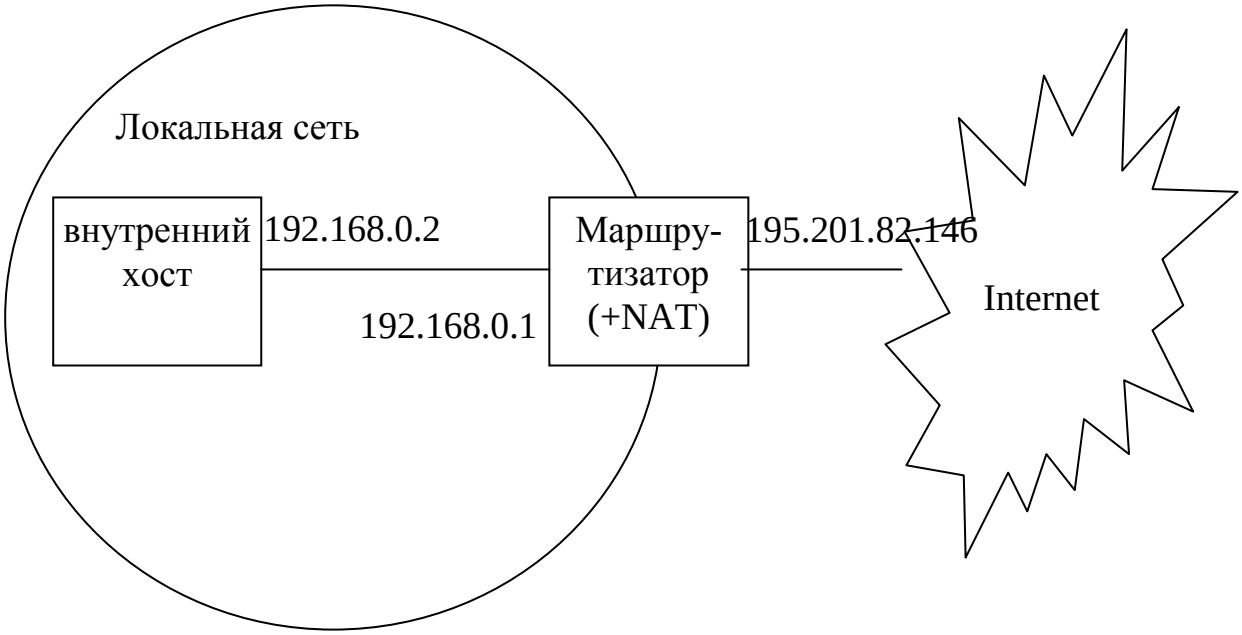


Рис. 3.11. Пример использования механизма NAT

Таблица 3.1.

Таблица трансляции адресов

Протокол	Внутренний локальный ip-адрес : порт	Внутренний зарегистрированный ip-адрес: порт.	Внешний ip-адрес: порт
TCP	192.168.0.2: 2001	195.201.82.146: 2161	195.242.2.2 : 80

Предположим, с хоста с адресом 192.168.0.2 пытаются установить защищенное соединение с внешним хостом 195.242.2.2, используя протокол аутентифицирующего заголовка (AH). При прохождении маршрутизатора, ip-адрес отправителя меняется, как было описано выше. Протокол AH определяет, что значение имитовставки рассчитывается, включая неизменяемые поля IP-заголовка, в частности – адрес отправителя. Сторона-получатель, обнаружит неверное (из-за трансляции адресов) значение имитовставки и отбросит пакет.

Таким образом, механизм NAT и протокол AH несовместимы. В то же время, протокол ESP, который не контролирует целостность ip-заголовка, может использоваться совместно с трансляцией адресов.

Кроме того, RFC 2709 определяет расширение NAT - IPC-NAT (англ. «IPSec policy Controlled NAT» – NAT управляемый правилами IPSec). Оно позволяет решить указанную проблему путем создания IP-IP туннеля, одной из конечных точек которого является узел NAT. В этом случае, вместо модификации IP-адреса отправителя в заголовке исходного пакета, NAT-устройство помещает без изменений весь исходный пакет (который аутентифицирован AH), в новый IP-пакет, в заголовке которого в качестве адреса отправителя ставится адрес NAT-устройства. На стороне получателя из полученного пакета изымают исходный пакет и далее обрабатывают его как обычно.

Отдельно необходимо решать вопрос с распределением ключей. Если для этой цели используется протокол IKE (а он использует транспортный протокол UDP, порт 500), потребуется специально организовать пересылку соответствующих данных во внутреннюю сеть. В том, случае, если задействовать UDP-порт 500 не представляется возможным, можно использовать описываемый в документах RFC 3947, 3948 механизм NAT-T (от англ. «NAT traversal»), определяющий инкапсуляцию IKE и IPSec трафика в пакеты UDP. При этом задействуется порт 4500.

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. Но бывают и другие схемы подключения, которые будут рассмотрены ниже.

Английский термин, используемый для обозначения МЭ – «firewall». Поэтому в литературе межсетевые экраны иногда также называют файервол или брандмауэр (немецкий термин, аналог firewall).

Как уже было отмечено, 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 [13]:

- 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. Соответственно, в дополнение к перечисленным

выше возможностям, подобный МЭ может контролировать процесс установки соединения и проводить проверку проходящих пакетов на принадлежность разрешенным соединениям.

Шлюз прикладного уровня может анализировать пакеты на всех уровнях модели OSI от сетевого до прикладного, что обеспечивает наиболее высокий уровень защиты. 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.

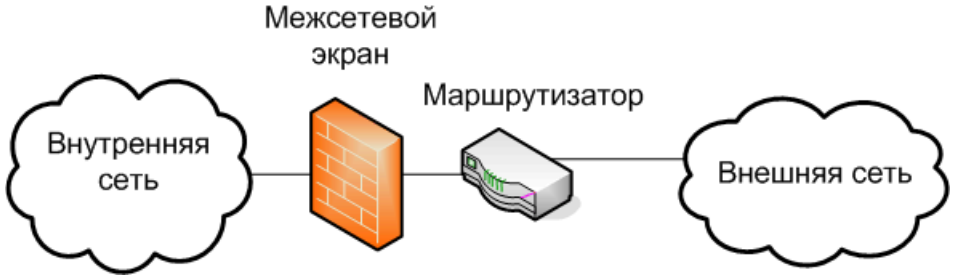
Рассмотрим теперь вопросы, связанные с установкой МЭ. На рис. 3.12 представлены типовые схемы подключения МЭ. В первом случае (рис. 3.12 а), МЭ устанавливается после маршрутизатора и защищает всю внутреннюю сеть. Такая схема применяется, если требования в области защиты от несанкционированного межсетевого доступа примерно одинаковы для всех узлов внутренней сети. Например, «разрешать соединения, устанавливаемые из внутренней сети во внешнюю, и пресекать попытки подключения из внешней сети во внутреннюю».

В том случае, если требования для разных узлов различны (например, нужно разместить почтовый сервер, к которому могут подключаться «извне»), подобная схема установки межсетевого экрана не является достаточно безопасной. Если в нашем примере нарушитель, в результате реализации сетевой атаки, получит контроль над указанным почтовым сервером, через него он может получить доступ и к другим узлам внутренней сети.

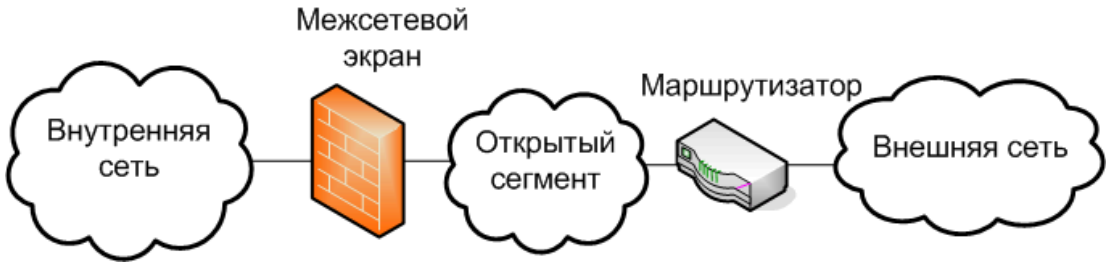
В подобных случаях иногда перед МЭ создается открытый сегмент сети предприятия (рис. 3.12 б), а МЭ защищает остальную внутреннюю сеть. Недостаток данной схемы заключается в том, что подключения к узлам открытого сегмента МЭ не контролирует.

Более предпочтительным в данном случае является использование МЭ с тремя сетевыми интерфейсами (рис. 3.12 в).

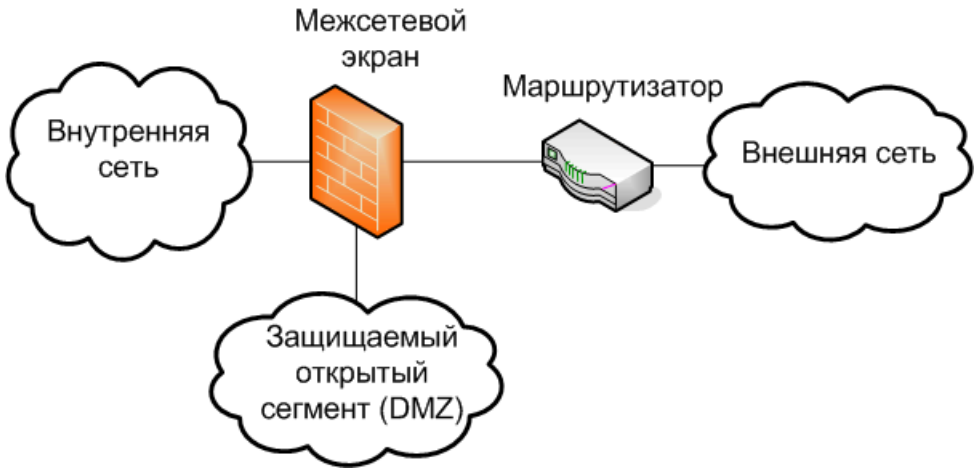
а)



б)



в)



г)

Рис. 3.12. Типовые схемы подключения межсетевых экранов

а) подключение МЭ с двумя сетевыми интерфейсами;

б) подключение МЭ с двумя сетевыми интерфейсами при выделении открытого сегмента внутренней сети;

в) подключение МЭ с тремя сетевыми интерфейсами;

г) подключение двух МЭ

МЭ с тремя сетевыми интерфейсами конфигурируется таким образом, чтобы правила доступа во внутреннюю сеть были более строгими, чем в открытый сегмент. В то же время, и те, и другие соединения могут контролироваться МЭ. Открытый сегмент в этом случае иногда называется «демилитаризованной зоной» – DMZ.

Еще более надежной считается схема, в которой для защиты сети с DMZ задействуются два независимо конфигурируемых МЭ (рис. 3.12 г). В этом случае, МЭ 2 реализует более жесткий набор правил фильтрации по сравнению с МЭ1. И даже успешная атака на первый МЭ не сделает внутреннюю сеть беззащитной.

В последнее время стал широко использоваться вариант установки программного МЭ непосредственно на защищаемый компьютер. Иногда такой МЭ называют «персональным». Подобная схема позволяет защититься от угроз исходящих не только из внешней сети, но из внутренней. Особенно актуально применение персональных МЭ при непосредственном подключении компьютера к потенциально опасной сети. Например, при подключении домашнего компьютера к Интернет.

Завершая учебное пособие, хочется отметить, что обеспечение информационной безопасности информационных и управляющих систем является сейчас очень актуальной задачей. Дополнительные сведения по данной тематике можно получить из специальной литературы, в частности из изданий, перечисленных в библиографическом списке.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. ГОСТ Р 50922-2006. Защита информации. Основные термины и определения. – М.: Стандартинформ, 2008. – 12 с.

2. ГОСТ Р 51275-2006. Защита информации. Объект информатизации. Факторы, воздействующие на информацию. Общие положения. – М.: Стандартинформ, 2007. – 11 с.

3. Девянин П.Н., Михальский О.О., Правиков Д.И., Щербаков А.Ю. Теоретические основы компьютерной безопасности: Учеб. пособие для вузов. – М.: Радио и связь, 2000. – 192 с.

4. ГОСТ Р ИСО/МЭК 15408-1-2002. Информационная технология. Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Часть 1. Введение и общая модель. – М.: Госстандарт России, 2002. – 40 с.

5. Зегжда Д.П., Ивашко А.М. Основы безопасности информационных систем. – М.: Горячая линия – Телеком, 2000. – 452 с.

6. Молдовян Н.А. Проблематика и методы криптографии. – СПб.: Изд-во СПбГУ, 1998. – 212 с.

7. Ященко В.В. и др. Введение в криптографию. / Под общ. ред. В.В.Ященко. – М.: МЦНМО, «ЧеРо», 1998. – 272 с.

8. Грунтович М.М. Основы криптографии с открытыми ключами. Учебное пособие. – Пенза: Изд-во Пензен. госуд. Ун-та, 2000. – 65 с.

9. Романец Ю.В., Тимофеев П.А., Шаньгин В.Ф. Защита информации в компьютерных системах и сетях. - М.: Радио и связь, 1999. – 328 с.

10. Зима В.М., Молдовян А.А., Молдовян Н.А. Компьютерные сети и защита передаваемой информации. - СПб.: Изд-во СПбГУ, 1998. – 328 с.

11. Конеев И.Р., Беляев А.В. Информационная безопасность предприятия. – СПб.: БХВ-Петербург, 2003. – 752 с.

12. Василенко О.Н. Теоретико-числовые алгоритмы в криптографии. – М.: МЦНМО, 2003. – 328 с.

13. Зима В.М., Молдовян А.А., Молдовян Н.А. Безопасность глобальных сетевых технологий. – 2-е изд. – СПб.: БХВ-Петербург, 2003. – 368 с.

14. Рудаков О.И., Грунтович М.М. Протоколы защиты информации в сети: IPSEC и SKIP. Специальная техника средств связи. Серия: Системы, сети и технические средства конфиденциальной связи. Пенза, ПНИЭИ, вып. № 1, 1999 г. С.79-85.

Нестеров Сергей Александрович

ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ И ЗАЩИТА ИНФОРМАЦИИ

Учебное пособие

Лицензия ЛР № 020593 от 07.08.97

Налоговая льгота — Общероссийский классификатор продукции
ОК 005-93, т. 2; 95 3005 - учебная литература

Подписано в печать 23.12.2008. Формат 60x84/16 Печать цифровая
Усл. печ. л. 8,0. Уч.-изд. л. 8,0. Тираж 100. Заказ 4620b.

Отпечатано с готового оригинал-макета, предоставленного автором, в цифровом типографском центре Издательства Политехнического университета:
195251, Санкт-Петербург, Политехническая ул., 29.
Тел. (812) 540-40-14
Тел./факс: (812) 927-57-76