Федеральное агентство по образованию

Государственное образовательное учреждение высшего профессионального образования

СЕВЕРО-ЗАПАДНЫЙ ГОСУДАРСТВЕННЫЙ ЗАОЧНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Р.Р. ХАМИДУЛЛИН, И.А. БРИГАДНОВ, А.В. МОРОЗОВ

МЕТОДЫ И СРЕДСТВА ЗАЩИТЫ КОМПЬЮТЕРНОЙ ИНФОРМАЦИИ

Учебное пособие

Санкт-Петербург

2005

Утверждено редакционно-издательским советом университета
УДК681.322.
ББК32.973

Хамидуллин Р.Р., Бригаднов И.А., Морозов А.В. Методы и средства защиты компьютерной информации: Учеб. пособие. – СПб.: СЗТУ, 2005. – 178 с.

Учебное пособие разработано в соответствии с требованиями государственных образовательных стандартов высшего профессионального образования по направлению подготовки дипломированного специалиста 230100 (специальность 230101 – вычислительные машины, комплексы, системы и сети) и направлению подготовки бакалавра 552800.

Учебное пособие также может быть использовано аспирантами и представителями смежных специальностей.

В пособии изложены вопросы, связанные с алгоритмами криптографических преобразований данных, алгоритмы аутентификации пользователей, требования к системам защиты информации, а также приведены практические примеры, способствующие успешному освоению теоретического материала.

Рецензенты: Г.А. Леонов, д-р физ.-мат. наук, проф., декан мат-мех. факультета СПбГУ; Д.А. Индейцев, д-р физ.-мат. наук, проф., директор института проблем машиноведения РАН.

ПРЕДИСЛОВИЕ

Как известно, всеобщая компьютеризация помимо очевидных выгод несёт с собой и многочисленные проблемы, наиболее сложной из которых является информационная безопасность. Широкое внедрение автоматизированных систем обработки данных, связанных с вводом, хранением, обработкой и выводом информации являются чрезвычайно уязвимыми по отношению к деструктивным воздействиям.

В связи с этим, важнейшей характеристикой любой компьютерной системы независимо от её сложности и назначения становится безопасность циркулирующей в ней информации.

В данном пособии излагаются основные понятия и разделы, позволяющие получить представления о задачах и направлениях современной криптографии.

В учебном пособии материал изложен в соответствии с Государственным Образовательным Стандартом на дисциплину «Методы и средства защиты компьютерной информации». Весь материал учебного пособия базируется только на открытых публикациях в отечественной и зарубежной печати.

В учебном пособии использованы примеры, взятые из отдельных литературных источников по данной тематике, способствующие успешному освоению теоретического материала. Учебное пособие снабжено списком литературы, необходимым для освоения данной дисциплины в процессе самостоятельной работы студентов, обучающихся без отрыва от производства.

ВВЕДЕНИЕ

Криптография — наука, связанная с созданием математических методов и средств защиты информации, обеспечивающих конфиденциальность, целостность, имитозащиту и ряд других функций. Вопросы безопасности — важная часть концепции внедрения новых информационных технологий во все сферы жизни общества. Поэтому широкомасштабное использование вычислительной техники и телекоммуникационных систем приводит к качественно новым возможностям несанкционированного доступа к ресурсам и данным информационной системы, т.е. к их высокой уязвимости. Таким образом, обеспечение целостности, достоверности и доступности информации – важные составные успеха деятельности любой организации. Формула успеха любой деятельности гласит [22]: кто владеет достоверной и полной информацией — тот владеет ситуацией, кто владеет ситуацией — тот способен управлять ею в своих интересах, кто способен управлять — тот способен побеждать. Эффективность механизмов защиты информации в значительной степени зависит от реализации ряда принципов. Во-первых, механизмы защиты следует проектировать одновременно с разработкой информационной системы, что позволяет обеспечить их бесконфликтность, своевременную интеграцию в вычислительную среду и сокращение затрат. Во-вторых, вопросы защиты следует рассматривать комплексно в рамках единой системы защиты информации.

Таким образом, управление безопасностью вычислительных систем охватывает широкий круг вопросов, в число которых входит: обеспечение целостности, конфиденциальности и аутентичности информации; разграничение прав пользователей по доступу к ресурсам автоматизированной системы; защита автоматизированной системы и

ее элементов от несанкционированного доступа; обеспечение юридической значимости электронных документов. Поэтому информация, как совокупность знаний о фактических данных и зависимостях между ними, стала стратегическим ресурсом; она — основа для выработки любого решения. В связи с этим защита информации, будучи сложной, наукоемкой и многогранной проблемой, по сути, в условиях внедрения современных информационных технологий, создания распределенных вычислительных систем и сетей связи, приобретает особую остроту.

Как известно [3, 21], современные вычислительные системы могут работать в мультипрограммном режиме (одновременно решается несколько задач), в мультипроцессорном режиме (создаются условия для решения задачи несколькими параллельно работающими процессорами), а также в режиме разделения времени, когда к информационным ресурсам одновременно может обращаться большое количество абонентов. При таких режимах работы в памяти компьютеров одновременно могут храниться программы и массивы данных различных пользователей ПК.

В то же время, циркулирующая в территориально распределенных системах и сетях информация становится уязвимой в связи с возрастанием многообразия угроз несанкционированного ее получения и использования. В свою очередь, основу обеспечения информационной безопасности в информационно-телекоммуникационных системах составляют криптографические методы и средства защиты информации, которые представляют собой совокупность организационно-технических и криптографических мероприятий, обеспечивающих решение следующих методов защиты, как служебной информации, так и информационных ресурсов в целом:

• шифрование всего информационного трафика, передающегося через открытые сети передачи данных и отдельных сообщений;

• криптографическая аутентификация устанавливающая связь разноуровневых объектов;

• защита несущего данные трафика средствами имитозащиты (защиты от навязывания ложных сообщений) и цифровой подписи с целью обеспечения целостности и достоверности передаваемой информации;

• шифрование данных, представленных в виде файлов либо хранящихся в базе данных;

• контроль целостности программного обеспечения путем применения криптографически стойких контрольных сумм;

• применение цифровой подписи для обеспечения юридической значимости документов; применение затемняющей цифровой подписи для обеспечения неотслеживаемости действий клиента в платежных системах, основанных на понятии электронных денег.

В данном учебном пособии рассмотрены не только методы и средства защиты компьютерной информации, но и основные задачи обеспечения информационной безопасности, которые решаются с помощью криптографических протоколов:

• **обмен ключевой информации с последующей установкой защищенного обмена данными;**

• аутентификация сторон, устанавливающих связь;

• авторизация пользователей при доступе к телекоммуникационным и информационным службам.

ГЛАВА 1. ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ КОМПЬЮТЕРНЫХ СИСТЕМ

1.1. Основные понятия и определения

Криптография **[25,26] – это раздел прикладной математики, изучающий методы преобразования информации в целях сокрытия ее содержания.**

Криптоанализ **– это раздел прикладной математики, изучающий методы, алгоритмы, программные и аппаратные средства анализа криптосистем с целью извлечения конфиденциальных данных.**

Отправитель – это субъект, посылающий сообщение.

Получатель – это субъект, получающий данное сообщение.

Текст – упорядоченный набор из элементов выбранного алфавита.

Открытым текстом называется исходное сообщение.

Зашифрование - процесс маскировки сообщения способом, позволяющим скрыть его суть.

Шифр – совокупность обратимых преобразований множества открытых данных на множество зашифрованных данных, заданных алгоритмом криптографического преобразования.

Шифртекстом (криптограммой) называется зашифрованное сообщение.

Расшифрование - процесс преобразования шифртекста в открытый текст. Процедура криптографического преобразования представлена на (рис.1.1).

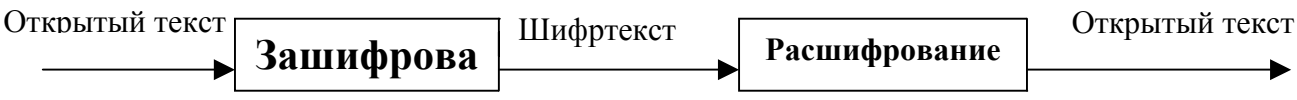

Рис.1.1. Схема криптографического преобразования

Шифртекст в дальнейшем изложении будем обозначать буквой

C, а открытый текст буквой M. Если шифрование сочетается со сжатием, то размер $C < M$. Функция зашифрования E, оперируя с M, создает C. Математически это можно записать, как $E(M) = C$.

В процессе расшифрования функция расшифрования D, оперируя с C, восстанавливает M по формуле $D(C) = M$.

Так как смысл зашифрования и последующего расшифрования сообщения заключается в восстановлении исходного открытого текста, то верно следующее тождество: $D(E(M)) \equiv M$.

На рис.1.2 приведена классификация существующих алгоритмов шифрования.

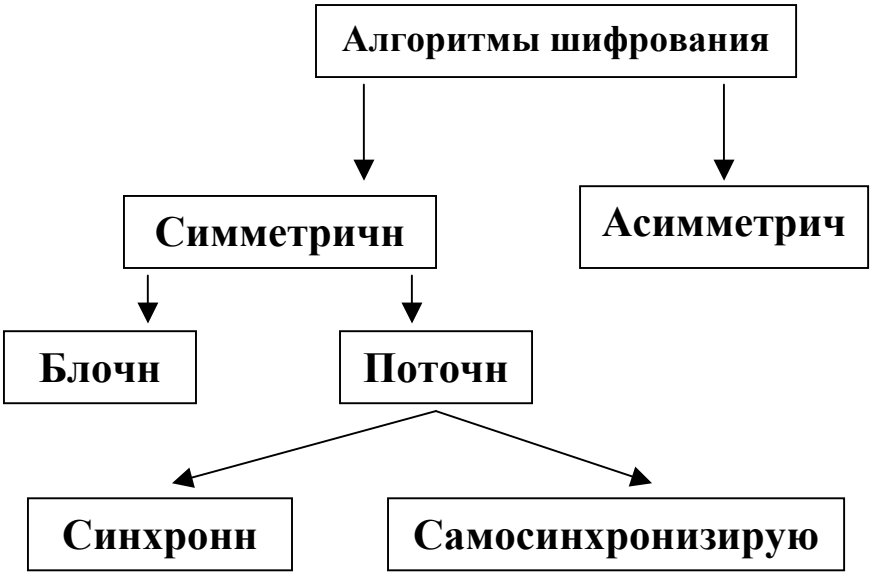

Рис.1.2. Классификация алгоритмов шифрования

В основе **криптографических алгоритмов** лежат два основных математических преобразования, а именно **замены** и **перестановки**, все остальные являются лишь комбинацией этих двух методов.

В перестановочных шифрах символы открытого текста изменяют своё местоположение.

В шифрах замены один символ открытого текста замещается символом зашифрованного текста. Различают четыре типа шифров

замены: простой, сложный, блочный, а также полиалфавитные шифры замены. Криптография позволяет производить [1, 14, 26]:

- шифрование передаваемых сообщений (хранимых данных) для защиты от утечки информации;
- контроль целостности передаваемых сообщений и хранимых данных с целью обнаружения случайных или преднамеренных искажений;
- проверку подлинности передаваемых сообщений. В этом процессе участвуют, по крайней мере две стороны: претендент, доказывающий аутентичность, и верификатор, проверяющий аутентичность.
- защиту программ от несанкционированного копирования и распространения;
- организацию парольных систем.

Криптография связана с различными областями математики (алгеброй, теорией чисел, теорией вероятностей, теорией сложности, вычислительной математикой, теорией связи и т.д.), а также с многочисленными техническими дисциплинами, создающими фундамент для построения как средств защиты данных, так и аппаратуры взлома шифров [2, 4, 23]. Некоторые вопросы теории чисел, используемые в криптографии, рассмотрены в приложении [П1,…,П6] данного пособия.

Отдельные разделы теории связи тесно связаны с криптографией, например теория корректирующего кодирования, которая также решает задачу обеспечения целостности передаваемых данных. Одно из основных различий между криптографией и теорией кодирования заключается в том, что первая защищает от целенаправленных действий нарушителя, а вторая – от случайных изменений, передаваемых данных.

Криптосистема – это система, реализованная программно, аппаратно или программно – аппаратно и осуществляющая

криптографическое преобразование информации. Она состоит из пространства ключей, открытых текстов, шифртекстов и алгоритмов зашифрования и расшифрования.

Ключ (K) это сменный элемент шифра, применяемый для зашифрования отдельного сообщения, обеспечивающий выбор одного варианта преобразования из совокупности возможных. Ключом определяется в первую очередь криптостойкость защищаемой информации. Множество возможных ключей называют пространством ключей. Ключи используются в операциях зашифрования и расшифрования.

Таким образом, функции зашифрования и расшифрования можно представить в следующем виде:

$$E_K(M) = C, \qquad D_K(C) = M, \qquad D_K(E_K(M)) \equiv M.$$

Эти равенства справедливы для симметричных криптосистем.

В асимметричных криптосистемах для зашифрования и расшифрования используют различные ключи, то есть ключ зашифрования K_1 отличается от ключа расшифрования K_2.

В данном случае, функции зашифрования и расшифрования приобретают следующий вид:

$$E_{K_1}(M) = C,$$

$$D_{K_2}(E_{K_1}(M)) \equiv M.$$

Гаммирование — процесс наложения по определенному закону гаммы шифра на открытые данные. Под гаммой шифра понимается псевдослучайная двоичная последовательность, вырабатываемая по заданному алгоритму и предназначенная как для зашифрования открытых данных, так и для расшифрования.

Синхропосылка - исходные открытые параметры алгоритма криптографического преобразования.

Криптографический протокол – алгоритм, выполняемый не менее чем двумя сторонами, задаваемый последовательностью операций, выполняемых каждой из сторон.

Конфиденциальность – свойство информации быть доступной только ограниченному кругу пользователей информационной системы, в которой циркулирует данная информация.

При передаче данных от отправителя к получателю могут быть две ситуации:

- знание ключа любой стороны позволяет вычислить ключ другой стороны;

- знание ключа одной из сторон не позволяет вычислить ключ другой стороны.

В первом случае используется **симметричное преобразование** данных, при этом само преобразование данных может не быть обратимым. Во втором используется **асимметричное преобразование** данных, при этом преобразование может быть обратимым. Таким образом, понятие симметричности означает двунаправленную вычислимость ключа.

В **симметричных шифрах** секретными являются ключи зашифрования и расшифрования. В **асимметричных шифрах** ключ зашифрования не позволяет вычислить секретный ключ расшифрования и поэтому может быть опубликован.

Дешифрование - это нарушение конфиденциальности шифртекста, достигнутое методами криптоанализа.

Стойкость криптоалгоритма (**криптостойкость**) заключается в способности шифра противостоять попыткам его расшифрования. Криптостойкость зависит от сложности алгоритма преобразования, длины ключа и объема ключевого пространства.

Атака - это реализация угрозы безопасности для криптосистемы. Известны несколько типов атак на криптографические алгоритмы:
- атака с известным шифртекстом; при этом предполагается, что

противник знает криптосистему, то есть алгоритмы шифрования, имеет набор перехваченных криптограмм, но он не знает секретный ключ;

- атака с известным открытым ключом, когда противник имеет доступ не только к шифртекстам нескольких сообщений, но и к открытому тексту этих сообщений;
- простая атака с выбором открытого текста; в этом случае у противника есть доступ не только к шифртекстам и открытым текстам нескольких сообщений, но и возможность выбирать открытый текст для шифрования;
- адаптивная атака с выбором открытого текста; криптоаналитик имеет возможность выбирать открытые тексты с учетом того, что криптограммы всех предыдущих открытых текстов ему известны;
- атака с выбором шифртекста; при этом криптоаналитик имеет возможность выбрать необходимое количество криптограмм и получить соответствующие им открытые тексты;
- адаптивная атака с выбором шифртекста, когда криптоаналитик, выбирая очередную криптограмму, знает все открытые тексты, соответствующие предыдущим криптограммам;
- атака с выбором текста, когда противник может выбирать как криптограммы (и дешифровывать их), так и открытые тексты (и зашифровывать их);
- атака с выбором ключа; при этом криптоаналитик знает не сами ключи, а связи между различными ключами;

Теоретически существуют стойкие алгоритмы зашифрования. Для того чтобы алгоритм считался таковым, он должен удовлетворять следующим требованиям:

- длина ключа и длина открытого сообщения должны быть одинаковыми;
- ключ должен использоваться только один раз;

- выбор ключа из ключевого пространства должен происходить равновероятно.

Перечисленные требования труднореализуемы. В результате применение современной аппаратно-программной базы приводит к неабсолютной стойкости используемых алгоритмов шифрования. Соответствующие криптографические алгоритмы можно классифицировать по степени доказуемости уровня стойкости: безусловно стойкие, доказуемо стойкие [18, 19].

Безусловно стойкие криптоалгоритмы гарантировано не позволяют раскрыть ключ.

Безопасность **доказуемо стойких** криптоалгоритмов определяется сложностью решения хорошо исследованных задач, например разложения числа на множители. Отличительной особенностью таких криптоалгоритмов является их «жёсткость», то есть невозможность модификации (усиления) путём незначительных изменений. Поэтому падение сложности задачи разложения приводит к соответствующему снижению стойкости криптографических алгоритмов, основанных на этой задаче.

Целостность – свойство информации или программного обеспечения сохранять свою структуру и содержание в процессе передачи и хранения.

Достоверность – свойство информации, выражающееся в строгой принадлежности объекту, который является её источником, либо тому объекту, от которого эта информация принята.

Оперативность – способность информации или некоторого информационного ресурса быть доступным для конечного пользователя в соответствии с его временными потребностями.

Хэш-функция – это функция, отображающая аргумент произвольной конечной длины в образ фиксированной длины. Функция, для которой по данному аргументу вычислить её значение легко, а по данному значению функции аргумент найти

сложно, называется хэш-функцией, вычислимой в одну сторону. Будем считать все хэш-функции вычислимыми в одну сторону. Если хэш-функция зависит от секретного ключа, она называется ключевой, в противном случае – бесключевой.

Коллизией называется пара аргументов M и M', которым соответствует одно и то же значение хэш-функции: $h(M) = h(M')$. Если число таких аргументов равно r, то говорят о коллизии кратности r. Если хэш-функция не имеет коллизий при заданных ограничениях на длину аргумента, то она называется свободной от коллизий.

Аутентификация – процесс проверки соответствия полученной информации заявленной. Различают аутентификацию данных, источника данных и опознавание участника протокола.

Имитозащита является частным случаем аутентификации. Если криптоалгоритм обеспечивает защиту получателя от навязывания ложной информации, то говорят, что криптоалгоритм реализует имитозащиту данных. Имитозащита включает в себя следующие понятия:

- контроль целостности данных;
- контроль подлинности данных;
- контроль неповторяемости данных;
- контроль того, что данные предназначены именно тому адресату, который их получил.

Имитозащита может быть обеспечена с помощью ключевой хэш-функции. Если аргументом хэш-функции являются защищаемые данные, то значение хэш-функции передается вместе с данными. Контроль подлинности и целостности осуществляется путем сравнения вычисленного и полученного значений хэш-функций. При совпадении этих значений при некоторых условиях можно утверждать, что источником данных является владелец секретного ключа и в ходе передачи, данные не были искажены.

Стеганография применяется для скрытия факта передачи секретных сообщений в другие сообщения, при этом скрывается даже само существование секрета. Отправитель может написать ничего не значащее сообщение и на том же листке бумаги скрыть секретное. В настоящее время скрывают сообщения в графических изображениях, при этом младший бит каждого байта изображения заменяется битом сообщения. Графическое изображение меняется от таких действий совершенно незначительно, так как большинство графических стандартов определяют намного большее число цветовых градаций, чем способен различить глаз человека. На принимающей стороне сообщение извлекается. Например, в черно-белом рисунке размером 1024x1024 пикселей можно скрыть сообщение размером 64 Кбайт.

Для сокрытия сообщения можно также использовать имитационные функции Питера Уэйнера [22]. Эти функции изменяют сообщение так, что его статистические параметры приобретают сходство с каким-либо текстом (статьёй газеты, литературным произведением и т.п.).

1.2. Основные криптографические методы защиты информации

Защита данных с помощью зашифрования - это одно из решений важнейшей проблемы обеспечения их безопасности. Зашифрованные данные становятся доступными только тому, кто знает, как их расшифровать, и поэтому похищение зашифрованных данных бессмысленно для несанкционированных пользователей.

Для зашифрования данных с незапамятных времен использовались коды и шифры. Под шифрованием понимается процесс, в котором криптографическому преобразованию подвергается каждый символ открытого текста, а под кодированием — процесс замены элементов

открытого текста кодами. Коды оперируют лингвистическими элементами, разделяя шифруемый текст на такие смысловые элементы, как слова и слоги.

В шифре всегда различают два элемента: алгоритм и ключ. Алгоритм позволяет использовать сравнительно короткий ключ для шифрования сколь угодно большого текста. Для защиты данных в вычислительных системах в основном используются шифры.

В настоящее время на практике используются следующие криптографические методы защиты:

-перестановки

-замены

-гаммирование

1.2.1. *Шифры перестановки*

Шифр, преобразования которого изменяют только порядок следования символов исходного текста, называется шифром перестановки (ШП).

Рассмотрим преобразование из ШП, предназначенное для зашифрования сообщения длиной n символов. Его можно представить с помощью таблицы

$$\begin{pmatrix} 1 & 2 & \dots & n \\ i_1 & i_2 & \dots & i_n \end{pmatrix},$$

где i_1 – номер места шифртекста, на которое попадает первая буква исходного сообщения при выбранном преобразовании, i_2 – номер места для второй буквы и т.д. В верхней строке таблицы выписаны по порядку числа от 1 до n, а в нижней – те же числа, но в произвольном порядке. Такая таблица называется подстановкой степени n.

Зная подстановку, задающую преобразование, можно осуществить как зашифрование, так и расшифрование текста. Например, если для преобразования используется подстановка типа

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 5 & 4 & 1 & 2 & 6 & 3 \end{pmatrix},$$

то слово ПРИКАЗ зашифровывается в АКПРЗИ.

Примером ШП, предназначенного для зашифрования сообщений длины *n*, является шифр, в котором в качестве множества ключей взято множество всех подстановок степени *n*, а число ключей такого шифра равно *n*!.

1.2.2. *Шифры замены*

Шифрование методом замены [25] основано на алгебраической операции, называемой подстановкой - взаимно однозначное отображение некоторого конечного множества M на себя. Число N элементов этого множества называется степенью подстановки. Природа множества M роли не играет, поэтому можно считать, что

$$M = 1,2,\ldots,N.$$

Две подстановки называются независимыми, если они не имеют общих действительно перемещаемых чисел.

Количество m чисел, действительно перемещаемых подстановкой S, называется длиной цикла подстановки.

В криптографии рассматриваются четыре типа подстановки: моноалфавитная, гомофоническая, полиалфавитная и полиграммная.

В примерах, приведенных ниже, использовано кодирование букв русского 32 буквенного алфавита (без буквы ё), приведенного в табл. 1.1, где знак «□» означает пробел. При моноалфавитной замене каждой букве алфавита открытого текста ставится в соответствие одна буква шифротекста из этого же алфавита. Простым и самым древним из известных подстановочных шифров является шифр, который использовал Гай Юлий Цезарь [1,14]. В шифре Цезаря каждая буква

алфавита заменяется буквой, находящейся на три позиции дальше в этом же алфавите, т.е. $k = 3$.

Таблица 1.1.

Буква	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л
Код	01	02	03	04	05	06	07	08	09	10	11	12
Буква	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч
Код	13	14	15	16	17	18	19	20	21	22	23	24
Буква	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	□ (пробел)			
Код	25	26	27	28	29	30	31	32	33			

Данный алгоритм зашифрования можно выразить следующими **формулами, где каждая буква открытого текста *M* заменяется буквой шифрованного текста.**

В общем виде при любом сдвиге

$$C = E(M) = (M + k)\ mod(N),$$

где $k = 1,\ldots,32$ для русского алфавита.

Алгоритм расшифрования имеет вид

$$M = D(C) = (C - k)\ mod\ (N).$$

Если русский текст зашифрован с помощью шифра Цезаря, то с помощью простого перебора 32 возможных вариантов ключей можно легко раскрыть шифр. Применение метода последовательного перебора всех возможных вариантов оправдано следующими тремя важными характеристиками данного шифра:

- **известны алгоритмы зашифрования и расшифрования;**
- **необходимо перебрать всего 32 варианта;**
- **язык открытого текста известен и легко узнаваем.**

Существуют шифры, которые базируются на методе многобуквенного шифрования (шифр Плейфейера) [15, 17]. В нем комбинации, состоящие из двух букв (биграммы), открытого текста *M* рассматриваются как самостоятельные единицы, преобразуемые в биграммы шифрованного текста.

Не вдаваясь в подробности алгоритма шифра Плейфейера [23], отметим, что поскольку в алфавите 32 буквы, а биграмм 32 х 32 =1024, то идентифицировать биграммы сложнее, чем отдельные буквы.

Поэтому шифр Плейфейера надёжнее простых моноалфавитных шифров.

Относительная частота появления отдельных букв колеблется гораздо в более широком диапазоне, чем частота появления биграмм, поэтому анализ частотности употребления биграмм тоже оказывается сложнее анализа частотности употребления букв.

В основу многобуквенных шифров (шифр Хилла) [17,22] положен принцип замены каждых m_i - последовательных букв открытого текста C_i - буквами шифрованного текста.

Подстановка определяется m линейными уравнениями, в которых каждому символу присваивается определенное числовое значение. Пусть $m = 3$, тогда получаем следующую систему уравнений:

$$C_1 = (k_{11}p_1 + k_{12}p_2 + k_{13}p_3) \bmod N,$$
$$C_2 = (k_{21}p_1 + k_{22}p_2 + k_{23}p_3) \bmod N,$$
$$C_3 = (k_{32}p_1 + k_{32}p_2 + k_{33}p_3) \bmod N.$$

В векторной форме записи эта система уравнений выглядит следующим образом:

$$\begin{pmatrix} C_1 \\ C_2 \\ C_3 \end{pmatrix} = \begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix},$$

где C и P – векторы размерности 3, представляющие шифрованный и открытый текст соответственно; K – матрица размерности 3 x 3, является ключом шифрования по $mod\ N$.

В общем виде систему Хилла можно записать в следующем виде:

$$C = E_K(P) = KP,$$
$$P = D_K(C) = K^{-1}C = K^{-1}KP = P,$$

где K^{-1} – матрица, обратная матрице K, то есть матрица, для которой выполняется равенство $KK^{-1} = K^{-1}K$=I - единичная матрица.

Преимущество шифра Хилла состоит в том, что он полностью маскирует частоту вхождения отдельных букв; чем больше размер матрицы, тем больше в шифрованном тексте скрывается информации о различиях в значениях частоты появления других комбинаций символов. Например, шифр с матрицей 3 x 3 скрывает частоту появления не только отдельных букв, но и двухбуквенных комбинаций.

Шифры, использующие несколько моноалфавитных подстановок, применяемых в ходе шифрования открытого текста в зависимости от определённых условий, называются полиалфавитными шифрами.

Примером полиалфавитного шифра замены является система Виженера [25, 26]. Шифрование происходит по таблице, которая представляет собой квадратную матрицу размерностью *n* x *n*, где *n* – число букв используемого алфавита.

На рис. 1.3 показана таблица Виженера для русского языка (алфавит Z_{32} –32 буквы и пробел). Первая строка содержит все буквы алфавита. Каждая следующая строка получается из предыдущей циклическим сдвигом последней на одну букву влево.

На первом этапе шифрования выбирается ключ (ключевая фраза). На втором этапе под каждой буквой исходного сообщения последовательно записываются буквы ключа (если ключ оказался короче сообщения, то его используют несколько раз). Каждая буква шифртекста находится на пересечении столбца таблицы, определяемого буквой открытого текста, и строки, определяемой буквой ключа. Пусть, например, требуется зашифровать сообщение ШИФРЫ ЗАМЕНЫ с помощью ключа Х А К Е Р. Запишем строку исходного текста с расположенной под ней строкой с циклически повторяемым ключом:

Ш И Ф Р Ы З А М Е Н Ы

Х А К Е Р Х А К Е Р Х

Полученный шифртекст выглядит следующим образом:

М И Ю Х Л Ы А Ц К Э Р

Расшифрование полученной криптограммы производится следующим образом.

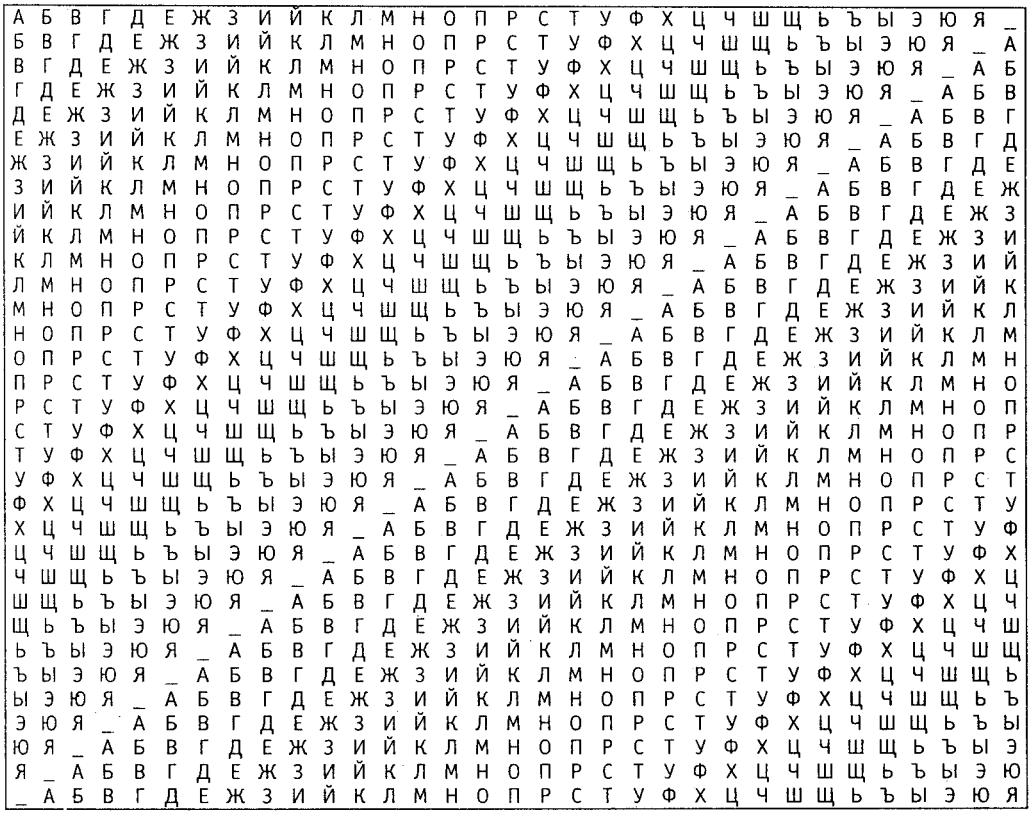

Рис. 1.3. Таблица Виженера

Под буквами шифртекста последовательно записываются буквы ключа; в строке таблицы, соответствующей очередной букве ключа, производят поиск соответствующей буквы шифртекста. Находящаяся над ней в первой строке таблицы буква является соответствующей буквой исходного текста, то есть первая буква текста определяется по схеме (Х $\xrightarrow{\text{по строке}}$ М $\xrightarrow{\text{по столбцу}}$ Ш).

1.2.3. *Шифр Вернама*

Шифр Вернама [25] является частным случаем системы шифрования Виженера при значении модуля равным 2.

Сообщение m обычно записывается в виде последовательности нулей и единиц. Длина ключа k равна длине сообщения.

Шифрование состоит в применении к m и k операции сложения по модулю 2 (*XOR)*, $E_k(m) = m \oplus k$.

Шифр Вернама считается практически не раскрываемым, так как данное сообщение с помощью подбора слишком большого ключа можно преобразовать в любое другое. Основная проблема состоит в хранении и передаче ключа.

К. Шеннон [25] доказал, что если ключ является истинно случайной двоичной последовательностью с равномерным законом распределения, при этом длина ключа равна длине исходного сообщения и используется этот ключ только один раз, то такой шифр является абсолютно стойким.

1.2.4. *Поточные шифры*

Поточные шифры [14, 22], в отличие от блочных, осуществляют поэлементное шифрование потока данных без задержки в криптосистеме, их важнейшим достоинством является высокая скорость преобразования, соизмеримая со скоростью поступления входной информации. Таким образом, обеспечивается шифрование практически в реальном масштабе времени вне зависимости от объема и разрядности потока преобразуемых данных.

В синхронных поточных шифрах гамма формируется независимо от входной последовательности, каждый элемент (бит, символ, байт и т. п.) которой таким образом шифруется независимо от других элементов. В синхронных поточных шифрах отсутствует эффект размножения ошибок, то есть число искаженных элементов в расшифрованной последовательности равно числу искаженных элементов зашифрованной последовательности, пришедшей из канала связи.

Вставка или выпадение элемента зашифрованной последовательности недопустимы, так как из-за нарушения синхронизации это приведет к неправильному расшифрованию

всех последующих элементов. В самосинхронизирующихся поточных шифрах элементы входной последовательности зашифровываются с учетом N предшествующих элементов, которые принимают участие в формировании ключевой последовательности. В самосинхронизирующихся шифрах имеет место эффект размножения ошибок, в то же время в отличие от синхронных восстановление синхронизации происходит автоматически через N элементов зашифрованной последовательности.

1.2.5. *Шифрование методом гаммирования*

Для зашифрования входной последовательности по этому методу отправитель производит побитовое сложение по модулю 2 ключа k (известный получателю и отправителю) и m-разрядной двоичной последовательности, соответствующей пересылаемому сообщению:

$$c_i = m_i \oplus k_i \ , \ i = \overline{1,m},$$

где m_i, k_i, c_i- очередной i-й бит соответственно исходного сообщения m, ключа k и зашифрованного сообщения c. Процесс расшифрования сводится к повторной генерации ключевой последовательности и наложению ее на зашифрованные данные. Уравнение расшифрования имеет вид:

$$m_i = c_i \oplus k_i \ , \ i=\overline{1,m}$$

Как известно [1, 22], если ключ является фрагментом истинно случайной двоичной последовательности с равномерным законом распределения, причем его длина равна длине исходного сообщения и используется этот ключ только один раз, после чего уничтожается, такой шифр является абсолютно стойким, его невозможно раскрыть, даже если криптоаналитик располагает неограниченным запасом времени и неограниченным набором вычислительных ресурсов. Действительно, противнику известно

только зашифрованное сообщение c, при этом все различные ключевые последовательности k возможны и равновероятны, а значит, возможны и любые сообщения m, то есть криптоалгоритм не дает никакой информации об открытом тексте.

Необходимые и достаточные условия абсолютной стойкости шифра:

- полная случайность ключа;
- равенство длин ключа и открытого текста;
- однократное использование ключа.

Основным недостатком данной схемы является равенство объема ключевой информации и объема передаваемых сообщений, поэтому гаммирование используется в каналах связи для шифрования только исключительно важных сообщений.

Если период гаммы превышает длину всего зашифрованного сообщения и неизвестна никакая часть исходного текста, то шифр можно раскрыть только прямым перебором (подбором ключа). В этом случае криптостойкость определяется только размером ключа.

Таким образом, процессом гаммирования называется процедура наложения на входную информационную последовательность гаммы шифра, то есть последовательности с выходов генератора псевдослучайной последовательности (ПСЧ). Последовательность называется псевдослучайной, если по своим статистическим свойствам она отличима от истинно случайной последовательности, но в отличие от последней является детерминированной, то есть значение алгоритма её формирования дает возможность повторения гаммы необходимое число раз.

Чтобы получить линейные последовательности элементов гаммы, длина которых превышает размер шифруемых сообщений, используются генераторы ПСЧ.

Надёжность шифрования методом гаммирования определяется качеством генератора гаммы. Различают гаммирование с конечной

и бесконечной гаммами. В качестве конечной гаммы может использоваться фраза (пример 1), в качестве бесконечной - последовательность, вырабатываемая генератором псевдослучайных чисел (пример 2).

В том случае, если множеством используемых для шифрования знаков сообщения является текст, отличный от двоичного кода, то его символы и символы гаммы заменяются цифровыми эквивалентами, которые затем суммируются по модулю *N*. Процесс зашифрования в этом случае определяется соотношением

$$c_i = (m_i + r_i) \; mod \; N, \quad i = 1, m, \qquad (1.1)$$

где m_i, r_i, c_i – очередной *i*-й знак исходного сообщения, гаммы и шифртекста соответственно;

N– число символов в алфавите сообщения; *m* – число знаков открытого текста.

Пример 1. Открытый текст: «Г А М Б И Т» («04 01 13 02 09 19») Гамма: «М О Д Е Л Ь» («13 15 05 06 12 29»). Открытый текст и гамма оцифрованы в соответствии с табл. 1.1). Для зашифрования используем соотношение (1.1) и операцию сложения по *mod* 33.

$$c_1 = 4 + 13(mod\ 33) = 17;$$
$$c_2 = 1 + 15(mod\ 33) = 16;$$
$$c_3 = 13 + 5\ (mod\ 33) = 18;$$
$$c_4 = 2 + 6(mod\ 33) = 8;$$
$$c_5 = 9 + 12(mod\ 33) = 21;$$
$$c_6 = 19 + 29(mod\ 33) = 15.$$

Шифртекст: «Р П С З Ф О» («17 16 18 08 21 15»).

Пример 2. Открытый текст: «ГАМБИТ» «04 01 13 02 09 19»

Гамма «0 7 0 6 0 9 0 4 0 5 0 8 07 09 . . . ».

Для зашифрования используем сложение по *mod* 2.

00100	00001	01101	00010	01001	10011
$\oplus$					
00111	00110	01001	00100	00101	01000
00011	00111	00100	00110	01100	11011

Шифртекст: «В Ж Г Е Л Ъ ».

1.2.6. *Блочные составные шифры*

Пусть блочный составной шифр [1,21] определяется семейством преобразований E следующим образом: $E = (P,C,K,f)$, где P - множество входных значений; C - множество выходных значений; K- пространство ключей; f- функция зашифрования $f: P \cdot K \to C$.

На основе этого семейства с помощью операции композиции можно реализовать блочные составные шифры.

Преобразование f_i называется i-м раундом шифрования, $i = \overline{1, r}$, ключ k_i - раундовым ключом. Если ключевые пространства K_i и преобразования f_i для всех раундов совпадают, то такой составной шифр называется итерационным, представляющим собой композицию одной и той же криптографической функции, используемой с разными ключами. Таким образом, идея, лежащая в основе композиционных шифров, состоит в построении криптостойкой системы путем многократного применения относительно простых криптографических преобразований, в качестве которых К. Шеннон [23] предложил использовать преобразования подстановки (substitution) и перестановки (permutation). Схемы, реализующие эти преобразования, называются *SP*-сетями.

Многократное использование этих преобразований позволяет обеспечить два свойства, которые должны быть присущи стойким шифрам: рассеивание (diffusion) и перемешивание (confusion) [2,7]. Рассеивание предполагает распространение влияния одного знака

открытого текста, а также одного знака ключа на значительное количество знаков шифртекста. Наличие у шифра этого свойства:

- позволяет скрыть статистическую зависимость между знаками открытого текста, иначе говоря, перераспределить избыточность исходного языка посредством распространения ее на весь текст;
- не позволяет восстанавливать неизвестный ключ по частям.

Например, обычная перестановка символов позволяет скрыть частоты появления биграмм, триграмм и т. д.

Цель перемешивания - сделать как можно более сложной зависимость между ключом и шифртекстом. Криптоаналитик на основе статистического анализа перемешанного текста не должен получить сколь-нибудь значительного количества информации об использованном ключе. Обычно перемешивание осуществляется при помощи подстановок. Как будет видно ниже, применение к каждому элементу открытого текста своей собственной подстановки приводит к появлению абсолютно стойкого шифра. Применение рассеивания и перемешивания порознь не обеспечивает необходимую стойкость (за исключением вышеупомянутого предельного случая), стойкая криптосистема получается только в результате их совместного использования. В современных блочных криптосистемах раундовые шифры строятся в основном с использованием операций замены двоичных кодов небольшой разрядности (схемы, реализующие эту нелинейную операцию, называются *S*-блоками; как правило, именно от их свойств в первую очередь зависит стойкость всей системы), перестановки элементов двоичных кодов, арифметических и логических операций над двоичными кодами. Важным достоинством многих составных шифров является их симметричность относительно операций зашифрования и расшифрования, которые по этой причине могут быть реализованы на одном устройстве. Переход от одного режима к другому

обеспечивается заменой последовательности раундовых ключей на обратную.

Если представить шифруемый блок данных (открытого p_i или закрытого c_i, текста) длиной n в виде пары полублоков

$$p_i = c_i = (L, R) = |L| = |R| = n/2,$$

то раундовая функция зашифрования над $n/2$-битовыми блоками данных имеет следующий вид :

$$f_i(p_i) = f_i(L, R) = (R, L \oplus f_i(R)),$$

для которого легко построить обратную функцию расшифрования $f_i^{-1}(c)$:

$$f^{-1}(c) = f_i^{-1}(L, R) = (R, L \oplus f_i(R)),$$

Шифр, использующий раундовые функции такого вида, называется шифром Фейстеля (рис. 1.4).

В подавляющем большинстве шифров рассматриваемой структуры используется разрядность блока, равная 64 битам, а в качестве операции $\oplus$ - поразрядное сложение по модулю 2.

Практической реализацией итерационного блочного шифра были разработанные Х. Фейстелем и другими сотрудниками фирмы *IBM* криптоалгоритмы Lucifer и *DES* (Data Encryption Standard). Алгоритм работы *DES* подробно рассмотрен в главе 2.

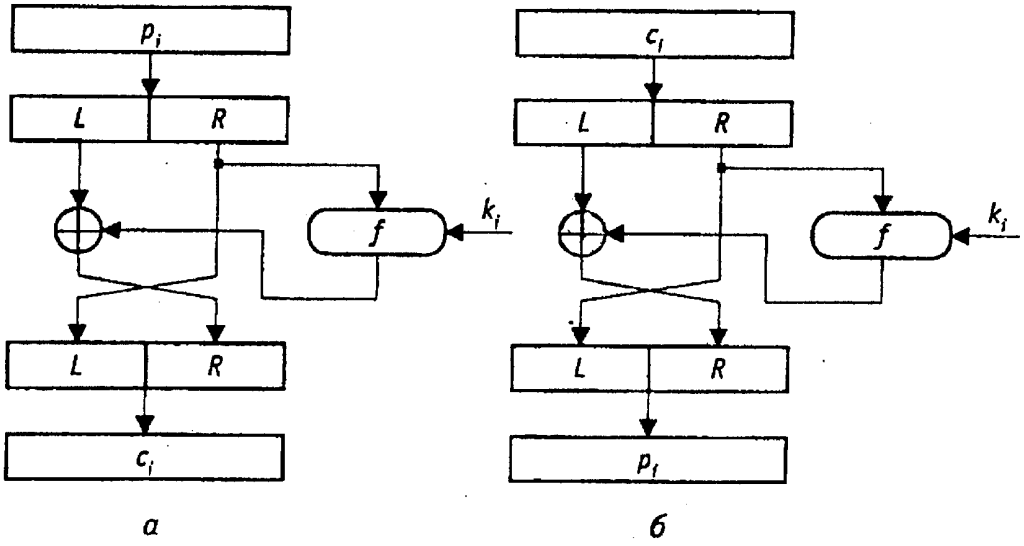

Рис.1.4. Схема петли Фейстеля:
а- зашифрование; б - расшифрование

ГЛАВА 2. СИММЕТРИЧНЫЕ КРИПТОСИСТЕМЫ

2.1. Алгоритм шифрования данных *DES*

Алгоритм *DES* (Data Encryption Standard) [1,22] является типичным представителем семейства блочных шифров, допускающим эффективную аппаратную и программную реализацию при достижении скоростей шифрования до нескольких мегабайт в секунду. Алгоритм *DES* предназначен для шифрования данных 64-битовыми блоками. Обобщенная схема шифрования в алгоритме *DES* показана на рис. 2.1. Алгоритм *DES* представляет собой комбинацию двух основных методов шифрования подстановки и перестановки. Основным комбинационным блоком *DES* является применение к тексту единичной комбинации этих двух методов. Такой блок называется раундом. *DES* включает 16 раундов, то есть одна и та же комбинация методов применяется к открытому тексту 16 раз.

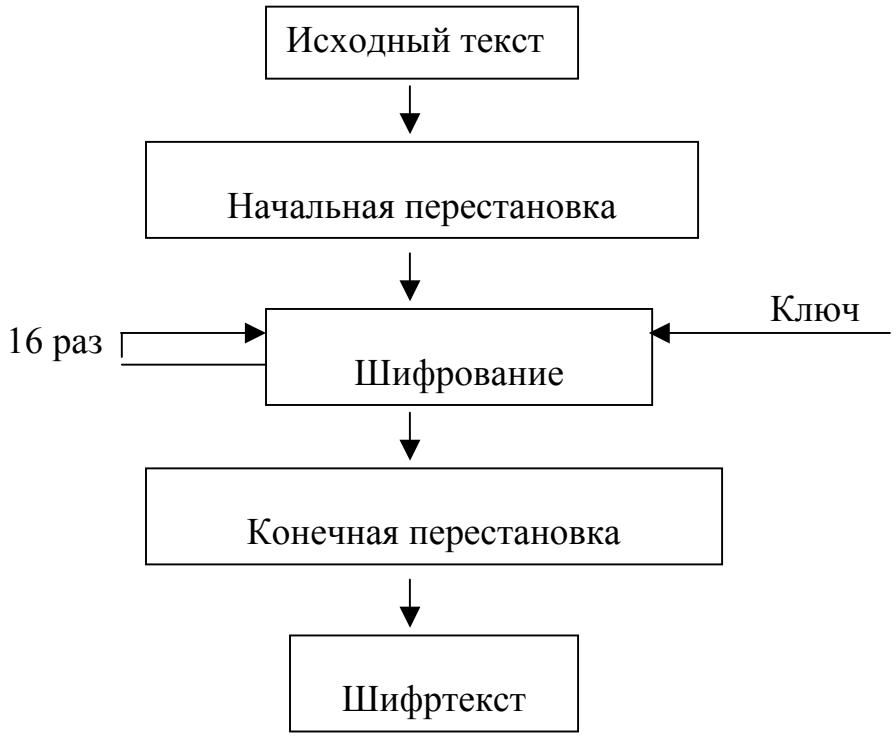

Рис.2.1. Обобщенная схема шифрования в алгоритме *DES*

При описании алгоритма *DES* (рис. 2.2) применены следующие обозначения:

L и *R* – последовательности битов (левая (left) и правая (right));

LR – конкатенация последовательностей *L* и *R*, т.е. такая последовательность битов, длина которой равна сумме длин *L* и *R*; в последовательности *LR* биты последовательности *R* следует за битами последовательности *L*;

XOR - операция побитового сложения по модулю 2.

Согласно рис. 2.2 64-битовый блок исходного текста преобразуется с помощью матрицы начальной перестановки *IP*, т.е биты входного блока переставляются в соответствии с матрицей *IP*.

Полученная последовательность битов разделяется на две последовательности: L_0 – левые или старшие биты, R_0 – правые или младшие биты, каждая из которых содержит 32 бита. Далее выполняется итеративный процесс шифрования, состоящий из 16 циклов. Результат *i*-й итерации можно описать следующими соотношениями:

$$L_i = R_{i-1},\ i = 1, 2,\ldots, 16;\ R_i = L_{i-1} \oplus f(R_{i-1}, K_i),\ i = 1,2,\ldots, 16.$$

Функция *f* называется функцией шифрования. Её аргументами являются последовательность R_{i-1}, полученная на предыдущем шаге итерации, и 48-битовый ключ K_i, который является результатом преобразования 64-битового ключа *K*. Таким образом, из приведенной выше схемы следует, что после первоначальной перестановки 64-битовый блок разбивается на правую и левую половины длиной по 32 бита каждая. Затем выполняется 16 раундов одинаковых преобразований с помощью функции *f*, в которых данные объединяются с соответствующим подключом. После 16 раунда правая и левая половины объединяются и алгоритм завершается заключительной обратной перестановкой IP^{-1}
По отношению к процессу зашифрования процесс расшифрования

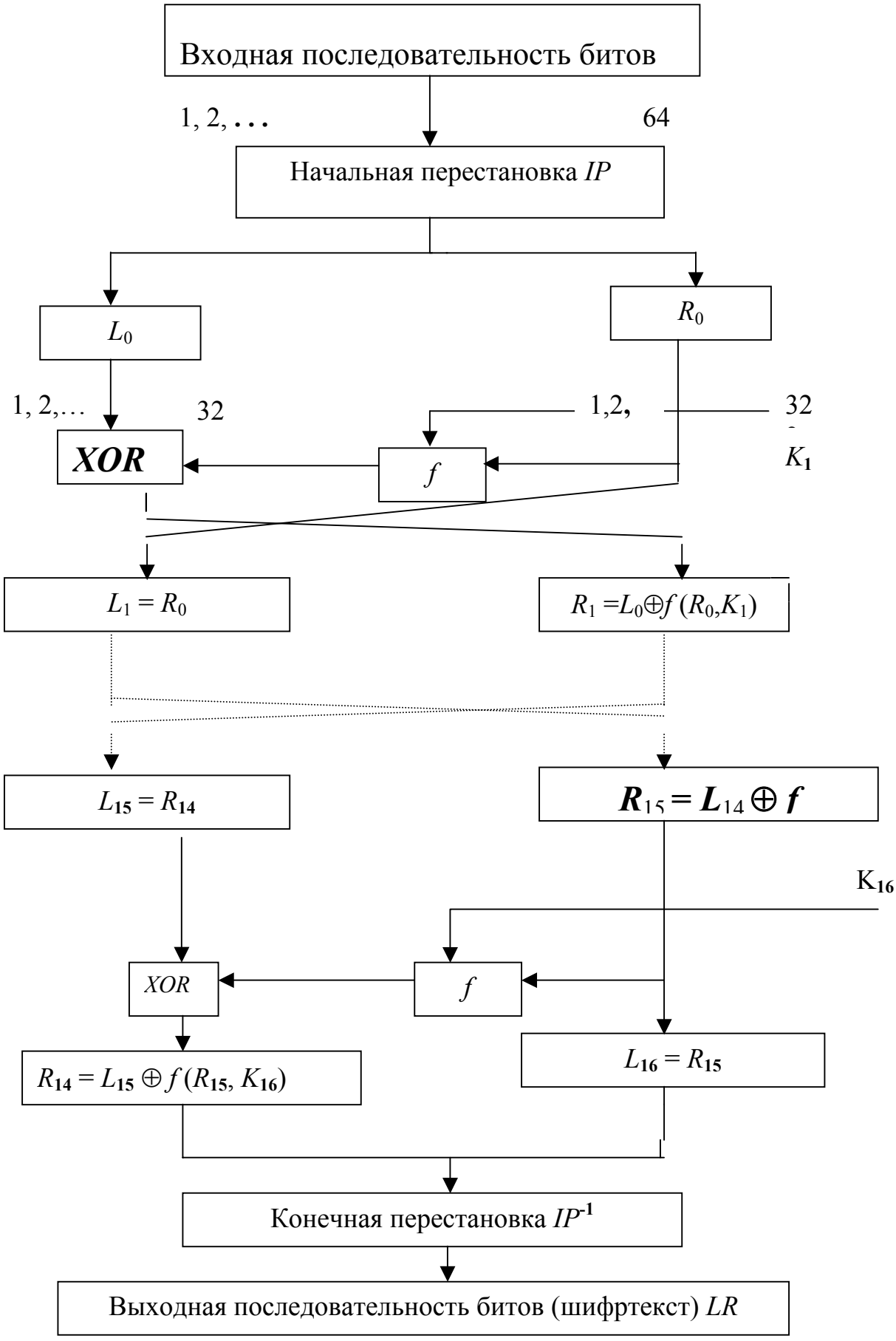

Рис. 2.2. Структурная схема алгоритма *DES*

является инверсным, то есть все действия должны быть выполнены в обратном порядке. Это означает, что расшифровываемые данные сначала переставляются в соответствии с матрицей IP^{-1}, а затем над последовательностью битов $R_{16}\ L_{16}$ выполняются те же действия, что и в процессе шифрования, но в обратном порядке.

Итеративный процесс расшифрования может быть описан следующими соотношениями:

$$R_{i-1} = L_i,\ \ i = 1, 2, \ldots,16;$$

$$L_{i-1} = R_i \oplus f(L_i, K_i),\ \ i = 1, 2, \ldots, 16.$$

Откуда следует, что для процесса расшифрования с переставленным входным блоком $R_{16}\ L_{16}$ на первой итерации используется ключ K_{16}, на второй итерации – K_{15} и т.д.

На последнем шаге итерации будут получены последовательности L_0 и R_0, которые конкатенируются в 64-битовую последовательность L_0R_0. Затем в этой последовательности 64 бита переставляются в соответствии с матрицей *IP*. Результатом такого преобразования является расшифрованное 64-битовое значение.

Таким образом, алгоритм *DES* позволяет использовать для зашифрования или расшифрования блока одну и ту же функцию.

Единственное отличие состоит в том, что ключи должны использоваться в обратном порядке. Иными словами, если в раундах зашифрования использовались ключи K_1, K_2, K_3,…, K_{16}, то ключами расшифрования будут K_{16}, K_{15}, K_{14},…,K_1.

2.2. Режимы использования блочных шифров

Режимы применения блочных шифров рассмотрим на примере алгоритма *DES* [7, 22], который, являясь базовым строительным блоком защиты передачи данных, подходит как для шифрования, так и для аутентификации данных. Для применения *DES* в

различных приложениях были определены четыре режима его работы:

- электронная кодовая книга *ECB* (Electronic Code Book);
- сцепление блоков шифра *CBC* (Cipher Block Chaining);
- обратная связь по шифротексту *CFB* (Cipher Feed Back);
- обратная связь по выходу *OFB* (Output Feed Back).

Считается, что этих четырёх режимов вполне достаточно, для того, чтобы использовать *DES* практически в любой области, для которых этот алгоритм подходит. Он позволяет непосредственно преобразовывать 64-битовый входной открытый текст в 64-битовый выходной шифрованный текст.

2.2.1. *Режим электронной кодовой книги*

В режиме *ECB* открытый текст обрабатывается блоками по 64 бита, и каждый блок шифруется одним и тем же ключом (рис. 2.3). При заданном ключе каждый 64-битовый блок открытого текста представляется уникальным блоком шифрованного текста, т.е. для каждой 64-битовой последовательности открытого текста указана соответствующая последовательность шифрованного текста. При длине сообщения, превышающей 64 бита, отправитель делит это сообщение на 64-битовые блоки с добавлением при необходимости заполнителей к последнему блоку. На рис. 2.3 открытый текст обозначен как M_1, M_2, …, M_n, а соответствующий шифрованный текст – C_1, C_2,…, C_n. Основным достоинством данного метода является простота реализации. Метод идеален для небольших объемов данных. Важной особенностью режима *ECB* является то, что одинаковые 64-битовые блоки открытого текста, если таковые встречаются в исходном сообщении, в шифрованном тексте тоже

будут представлены одинаковыми блоками.

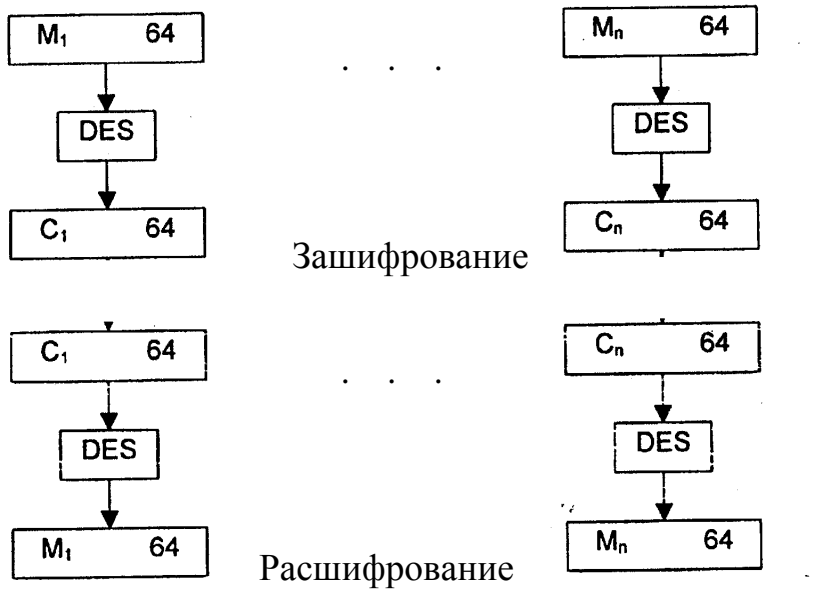

Рис. 2.3. Схема работы алгоритма в режиме электронной кодовой книги

Недостатком метода *ECB* является относительно низкая криптостойкость, так как при передаче достаточно длинных сообщений данный режим может не обеспечить необходимый уровень защиты.

2.2.2. *Режим сцепления блоков шифра*

Режим сцепления блоков шифра свободен от недостатков режима *ECB*. В этом режиме входное значение алгоритма зашифрования задаётся равным *XOR* - разности текущего блока открытого текста и полученного на предыдущем шаге блока шифрованного текста (рис. 2.4). В процессе шифрования все блоки открытого текста оказываются связанными, так как шифрование любого блока выполняется одним и тем же ключом, а входные данные,

поступающие на вход функции зашифрования, уже не жестко связаны с блоками открытого типа. В связи с этим повторяющиеся 64-битовые последовательности в шифрованном тексте не проявляются. При дешифровании текст, как и при режиме *ECB*, проходит через алгоритм дешифрования поблочно.

В результате соответствующий блок открытого текста получается как *XOR* – разность выходного блока алгоритма дешифрования и предыдущего блока зашифрованного текста. В этом режиме исходный файл *M* разбивается на 64-битовые блоки:

$$M = M_1, M_2, \ldots, M_n.$$

Первый блок M_1 складывается по модулю 2 с 64-битовым начальным вектором *IV*, который меняется ежедневно и держится в секрете. Полученная сумма затем шифруется с использованием ключа *DES*, известного и отправителю, и получателю информации.

Полученный 64-битовый шифр C_1 складывается по модулю 2 со вторым блоком текста, результат шифруется и получается второй 64-битовый шифр C_2 и т.д. Процедура повторяется до тех пор, пока не будут обработаны все блоки текста. Процесс зашифрования можно представить следующими соотношениями:

$$C_n = E_k(C_{n-1} \oplus M_n). \quad \text{Отсюда} \qquad D_k(C_n) = D_k(E_k(C_{n-1} \oplus M_n)),$$

$$D_k(C_n) = (C_{n-1} \oplus M_n), \qquad C_{n-1} \oplus D_k(C_n) = C_{n-1} \oplus C_{n-1} \oplus M_n = M_n.$$

При расшифровании для восстановления первого блока открытого текста необходимо выполнить операцию *XOR* по отношению к тому же вектору *IV* и первому блоку на выходе алгоритма дешифрования.

Значение вектора *IV* должно быть известно и отправителю, и получателю сообщения.

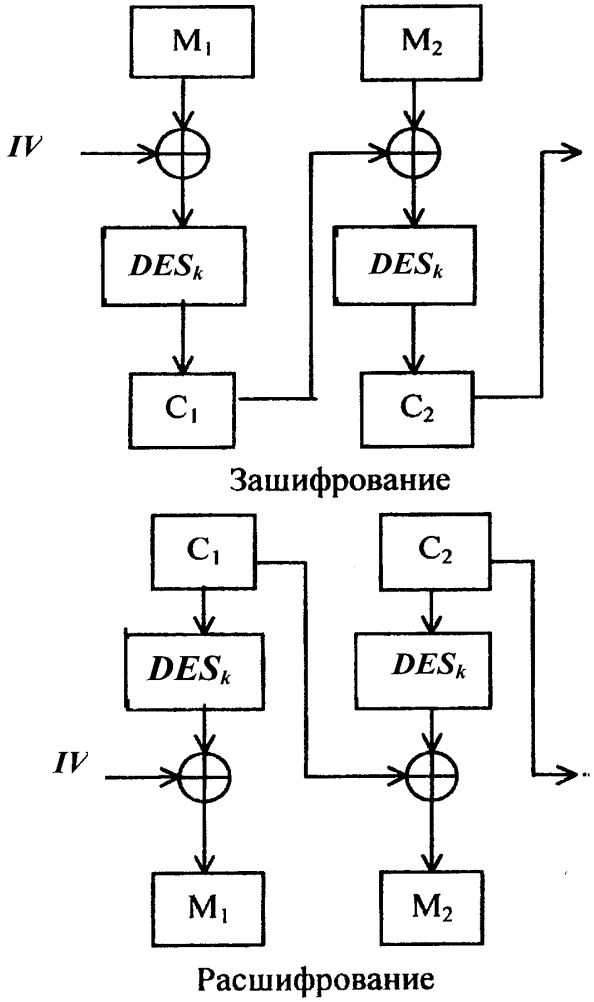

Рис. 2.4. Режим работы блочного алгоритма в режиме сцепления блоков шифра

2.2.3. *Обратная связь по шифртексту*

Схему *DES* можно преобразовать в потоковый шифр, используя режим шифрованной обратной связи (*CFB*), который избавляет от необходимости дополнять сообщение до целого числа блоков и позволяет работать в режиме реального времени, т.е. каждый

символ можно зашифровать и сразу же передавать получателю, не дожидаясь окончания шифрования остальной части сообщения.

Следует отметить, что для поточного шифра необходимо, чтобы длина зашифрованного текста в точности соответствовала длине открытого. На рис. 2.5 показана схема зашифрования в режиме *CFB*. При этом шифрованный текст, соответствующий любому элементу открытого текста, будет зависеть от всех предыдущих элементов открытого текста, так как происходит сцепление элементов открытого текста, как и в режиме *CBC*.

Процесс зашифрования происходит следующим образом. На входе функции шифрования размещается 64-битовый регистр сдвига, в котором размещается некоторое значение вектора инициализации *IV*. Главные *j* битов этого значения связываются операцией *XOR* с первой порцией открытого текста M_1 для получения первой порции шифрованного текста C_1, который подается на линию передачи данных. Содержимое регистра сдвига смещается влево на *j* битов, а в наименее значимые *j* битов помещается значение C_1. Весь процесс повторяется до тех пор, пока не будут зашифрованы все элементы открытого текста. Операция дешифрования происходит по той же схеме, однако для того, чтобы получить очередную порцию открытого текста с помощью операции *XOR*, необходимо связать полученные по обратной связи порции зашифрованного текста с порцией на выходе функции шифрования данных. Следует заметить, что в данном случае используется функция шифрования, а не расшифрования. Это можно объяснить следующим образом. Обозначим крайние слева *j* битов значения *X* через $S_j(X)$. Тогда

$$C_1 = M_1 \oplus S_j\,(E(IV)),$$

поэтому

$$M_1 = C_1 \oplus S_j\,(E\,(IV)).$$

Для последующих шагов расшифрования используются те же выкладки. Режим *CFB* может использоваться как для обеспечения конфиденциальности, так и для аутентификации.

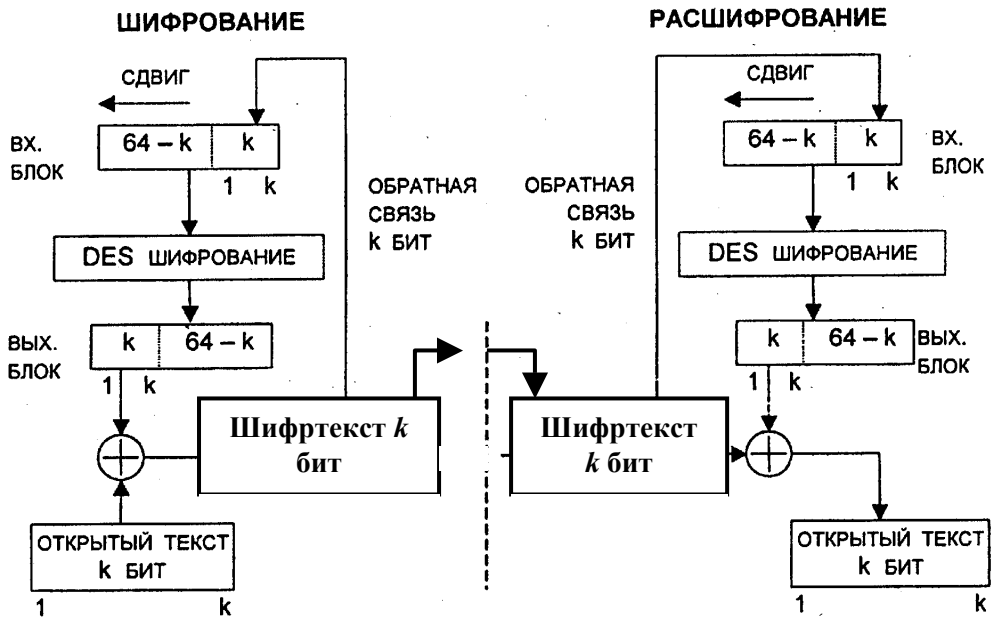

Рис. 2.5. Схема работы блочного алгоритма в режиме обратной связи по шифртексту

2.2.4. *Режим обратной связи по выходу*

В режиме обратной связи по выходу (*OFB*) в регистр сдвига подается порция зашифрованного текста (рис. 2.6). Входной блок вначале содержит вектор инициализации *IV*, выровненный по правому краю. При этом для каждого сеанса шифрования данных необходимо использовать новое начальное состояние регистра, которое должно пересылаться по каналу открытым текстом.

Преимущество *OFB* состоит в том, что влияние возможных искажений битов при передаче данных не распространяется на последующие порции данных. Если искаженные биты появились при передаче C_1, это влияет только на восстановленное из C_1 значение M_1, а все последующие порции открытого текста передачи данных повреждены не будут.

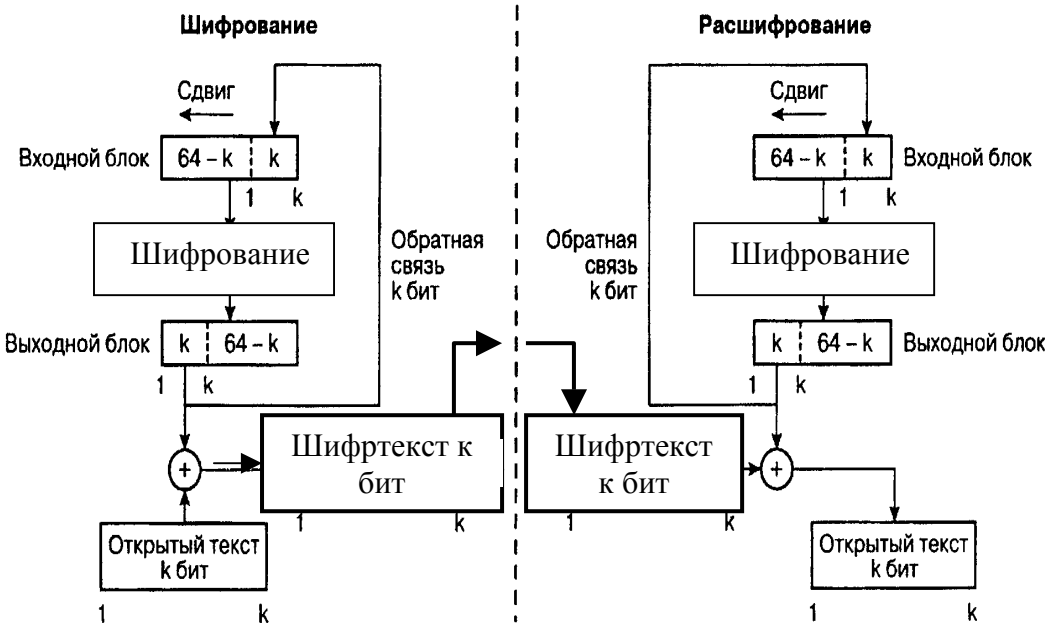

Рис. 2.6. Схема работы блочного алгоритма в режиме обратной связи по выходу

2.3. Алгоритм шифрования *AES*

Алгоритм шифрования *AES* [7] разработали два специалиста по криптографии - Дж. Деймен (J. Daemen) и В. Риджмен (V. Rijmen) из Бельгии. Этот алгоритм является нетрадиционным блочным шифром, поскольку не использует сеть Фейстеля для криптопреобразований. Алгоритм представляет каждый блок кодируемых данных в виде двумерного массива байтов размером 4 x 4, 4 x 6 или 4 x 8 в зависимости от установленной длины блока. Далее на соответствующих этапах производятся преобразования либо над

независимыми столбцами, либо над независимыми строками, либо вообще над отдельными байтами в таблице.

Алгоритм состоит из определенного количества раундов (от 10 до 14 - это зависит от размера блока и длины ключа), в которых последовательно выполняются преобразования Sub Bytes, Shift Rows, Mix Columns, Add Round Key. Они воздействуют на массив State, который адресуется с помощью указателя State. Преобразование AddRoundKey использует дополнительный указатель для адресации ключа раунда Round Key.

Преобразование Sub Bytes - нелинейная байтовая подстановка (рис. 2.7), которая воздействует независимо на каждый байт массива State, используя таблицу подстановок *S*-box. Эта таблица является обратимой.

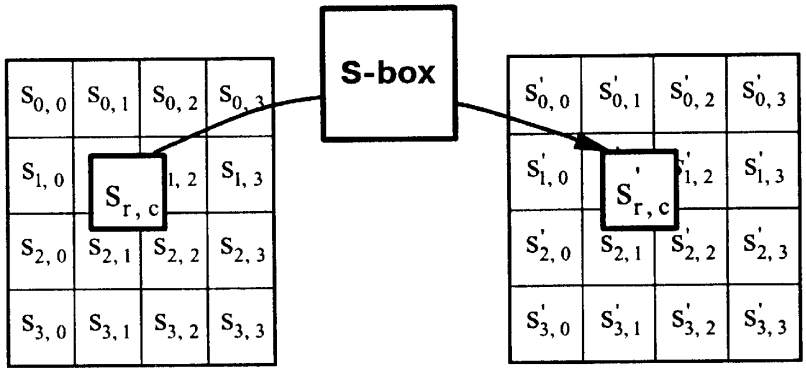

Рис. 2.7. Преобразование Sub Bytes, использующее таблицу подстановок S-box для обработки каждого байта массива State

При преобразовании Shift Rows байты в трех последних строках двумерного массива State циклически сдвигаются на различное число байтов. При этом первая строка не сдвигается (рис. 2.8.).

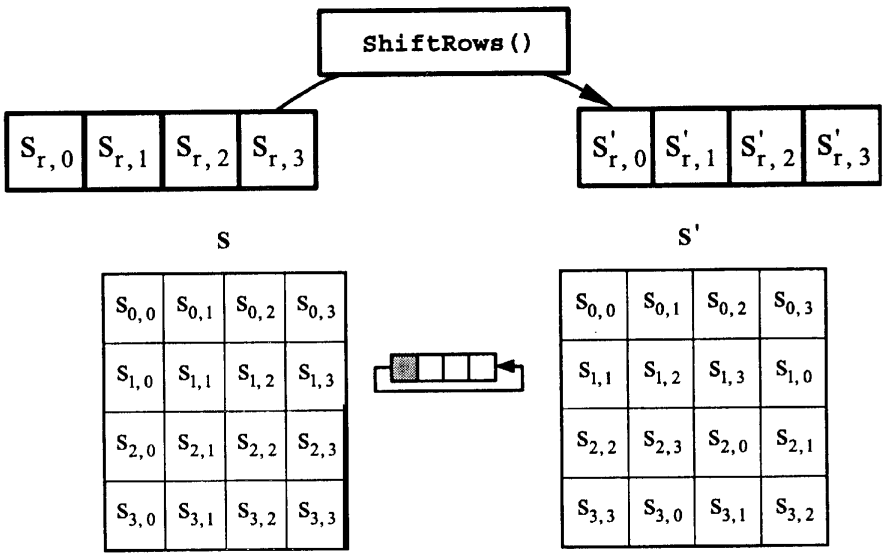

Рис. 2.8. Преобразование Syift Rows, использующее циклический сдвиг трёх последних строк в массиве State

Mix Columns – это математическое преобразование, перемешивающее данные внутри каждого столбца массива State (рис. 2.9). Преобразование Mix Columns воздействует поочередно на столбцы массива State, обращаясь с каждым из них как с четырехчленным полиномом.

Эти столбцы рассматриваются как полиномы над полем $GF\ (2^8)$ и умножаются на фиксированный полином $a(x)$ с приведением результата умножения по модулю $(x^4 +1)$.

При преобразовании Add Round Key ключ раунда Round Key прибавляется к массиву State с помощью операции простого побитового сложения *XOR* (сложения по модулю 2) - рис. 2.10. Каждый ключ раунда Round Key состоит из Nb слов, взятых из набора ключей (key schedule).

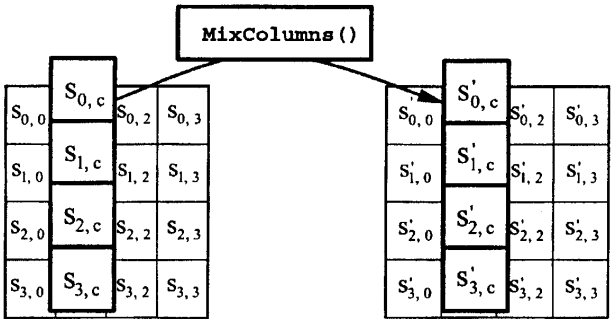

Рис. 2.9. Преобразование Mix Columns, поочерёдно обрабатывающее столбцы массива State

Перечисленные выше преобразования выполняются алгоритмом в каждом раунде, кроме последнего (в последнем раунде операция перемешивания столбцов отсутствует). Отсутствие операции перемешивания столбцов в последнем раунде делает всю последовательность операций алгоритма симметричной.

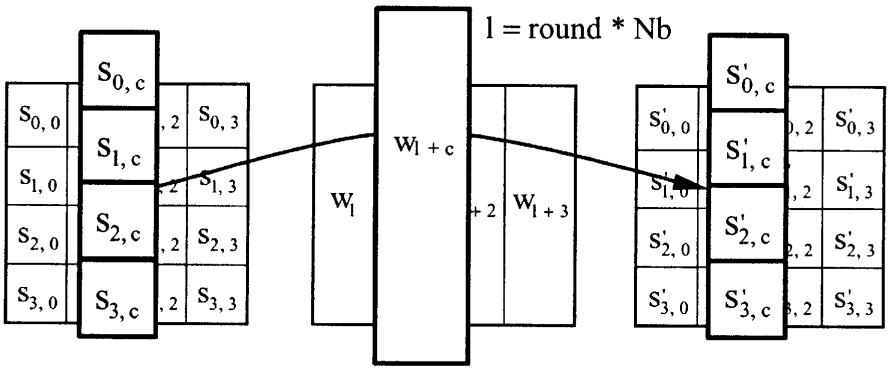

Рис. 2.10. Преобразование Add Round Key, производящее сложение *XOR* каждого массива State со словами из ключевого набора

Все преобразования в шифре *AES* имеют строгое математическое обоснование. Сама структура и последовательность операций позволяют выполнять данный алгоритм эффективно как на 8-битовых, так и на 32-битовых процессорах. В структуре алгоритма

заложена возможность параллельного исполнения некоторых операций, что может повысить скорость шифрования на многопроцессорных рабочих станциях в четыре раза.

Рассмотрим особенности применения алгоритмов симметричного шифрования.

Алгоритмы симметричного шифрования используют ключи относительно небольшой длины и могут быстро шифровать большие объемы данных. При симметричной методологии шифрования отправитель и получатель применяют для осуществления процессов зашифрования и расшифрования сообщения один и тот же секретный ключ. Алгоритмы симметричного зашифрования строятся исходя из предположения, что зашифрованные данные не сможет прочитать никто из тех, кто не обладает ключом для их расшифрования. Если ключ не был скомпрометирован, то при расшифровании автоматически выполняется аутентификация отправителя, так как только он имеет ключ, с помощью которого можно зашифровать информацию, и только получатель имеет ключ, с помощью которого можно ее расшифровать.

Алгоритмы симметричного шифрования применяются для абонентского шифрования данных, то есть для шифрования информации, предназначенной для отправки кому-либо, например, через Internet. Использование только одного секретного ключа для всех абонентов сети, конечно, недопустимо по соображениям безопасности: в случае компрометации (утери, хищения) ключа под угрозой будет находиться документооборот всех абонентов сети. В этом случае часто используется так называемая матрица ключей [7, 12].

Матрица ключей представляет собой таблицу, содержащую ключи парной связи абонентов. Каждая *i*-я строка матрицы представляет собой набор ключей конкретного абонента *i* для связи с остальными

N - 1 абонентами. Каждый элемент таблицы K_y предназначен для связи абонентов i и j и доступен только двум данным абонентам. Для всех элементов матрицы ключей соблюдается равенство $K_y = K_{ij}$. Наборы ключей (сетевые наборы) распределяются между всеми абонентами криптографической сети. Такие сетевые наборы должны распределяться по защищенным каналам связи или «из рук в руки».

Порядок использования систем с симметричными ключами таков:

1. Безопасно создается, распространяется и сохраняется симметричный секретный ключ.

2. Для получения зашифрованного текста отправитель применяет к исходному сообщению симметричный алгоритм шифрования вместе с секретным симметричным ключом. Таким образом, неявно подготавливается аутентификация отправителя и получателя, так как только отправитель знает симметричный секретный ключ и может зашифровать этот текст, только получатель знает симметричный секретный ключ и может расшифровать текст.

3. Отправитель передает зашифрованное сообщение. Симметричный секретный ключ никогда не передается в открытой форме по незащищенным каналам связи.

4. Получатель применяет к зашифрованному тексту тот же самый симметричный алгоритм шифрования/расшифрования вместе с тем же самым симметричным ключом (который уже есть у получателя) для восстановления исходного текста. Его успешное восстановление идентифицирует того, кто знает секретный ключ.

Для симметричных криптосистем актуальна проблема безопасного распределения симметричных секретных ключей. Всем системам симметричного шифрования присущи следующие недостатки:

- принципиальным является требование защищенности и надежности канала передачи секретного ключа для каждой пары участников информационного обмена;

- предъявляются повышенные требования к службе генерации и распределения ключей, обусловленные тем, что для n абонентов при схеме взаимодействия «каждый с каждым» требуется $n \bullet (n - 1) / 2$ ключей, то есть зависимость числа ключей от числа абонентов является квадратичной. Например, для $n = 1000$ абонентов требуемое количество ключей будет равно $n \bullet (n - 1) / 2 = 499500$ ключей.

Поэтому без эффективной организации защищенного распределения ключей широкое использование обычной системы симметричного шифрования в больших сетях, в частности глобальных, практически невозможно.

2.4. Российский стандарт шифрования данных ГОСТ 28147-89

В России установлен единый алгоритм криптографического преобразования данных для систем обработки информации в отдельных вычислительных комплексах и сетях ЭВМ. Он определяется положениями ГОСТ 28147-89 [27], в дальнейшем обозначаемый ГОСТ. Этот алгоритм предназначен для аппаратной и программной реализации, удовлетворяет необходимым криптографическим требованиям и не накладывает ограничений на степень секретности защищаемой информации. Алгоритм реализует шифрование 64-битовых блоков данных с помощью 256-битового ключа, состоящего из восьми 32-битовых подключей.

Использование подключей в 32 раундах алгоритма ГОСТ приведено в табл. 2.1. На каждом i-м раунде используется K_i - й подключ.

ГОСТ может применяться в следующих рабочих режимах:

- простая замена;
- гаммирование;
- гаммирование с обратной связью и выработка имитовставки.

Таблица 2.1.

Раунд:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Подключ:	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
Раунд:	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Подключ:	1	2	3	4	5	6	7	8	8	7	6	5	4	3	2	1

В процессе работы алгоритма последовательно на 32 раундах, выполняется следующий алгоритм зашифрования.

В начале шифруемый текст разбивается на левую L и правую R половины. На каждом i-м раунде алгоритма ГОСТ выполняются следующие операции:

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

Общая схема раунда алгоритма ГОСТ представлена на рис. 2.11.

На первом этапе над правой половиной шифруемого текста и i-м подключом выполняется операция сложения по модулю 2^{32}. Результат операции разбивается на восемь 4-битовых фрагмента; каждый из этих фрагментов поступает на вход своего S-блока. В алгоритме ГОСТ используются восемь различных S-блоков. Первый 4-битовый фрагмент поступает в первый S-блок, следующий 4-битовый фрагмент - во второй S-блок и т.д. Каждый из S-блоков является перестановкой чисел от 0 до 15. Например, один из S-блоков может выглядеть таким образом: 4, 10, 9, 2, 13, 8, 0, 14, 6, 11, 1, 12, 7, 15, 5, 3.

В этом случае, если на вход S-блока поступает 0, то на выходе появится 4. Если на вход поступает 1, то на выходе будет 10 и т.д. Все восемь S-блоков перестановки различны, фактически они служат дополнительным ключевым материалом и должны храниться в секрете. ГОСТ [27] не определяет метод генерации S-блоков, пользователь самостоятельно создает перестановки для S-блоков с помощью генератора случайных чисел.

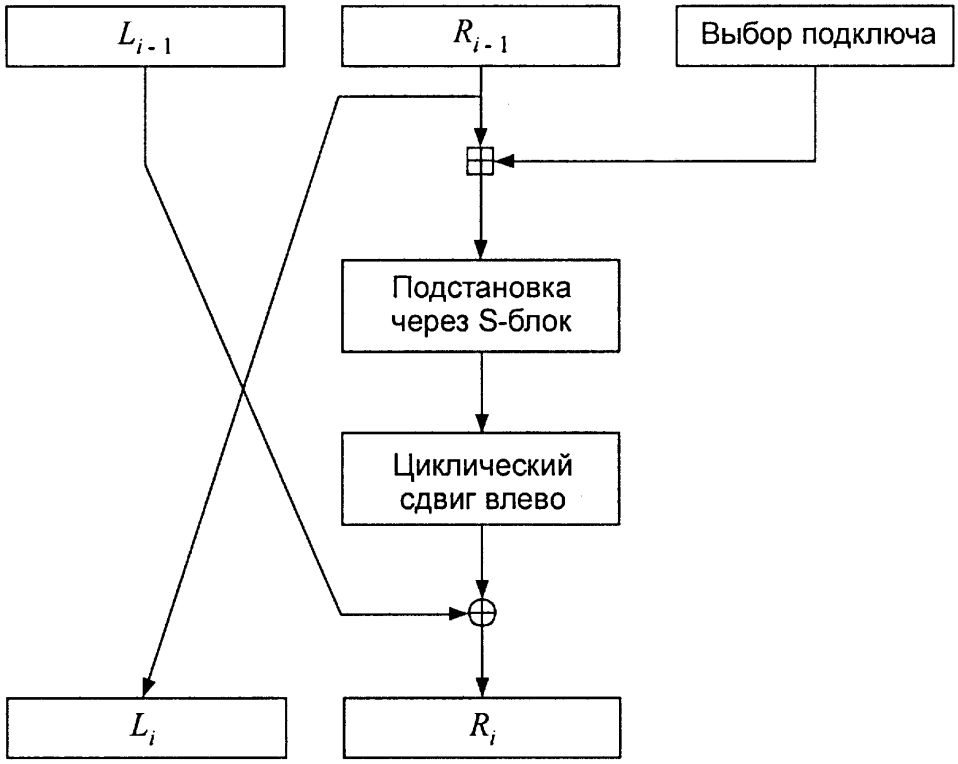

Рис. 2.11. Общая схема алгоритма ГОСТ

Для каждого *S*-блока задается своя подстановка, которая является долговременным секретным ключом. Генерация подстановок в *S*-блоках определяет криптостойкость этого алгоритма. Подобный прием служит также для развязывания различных сетей передачи данных - очевидно, что расшифрование будет успешным, если принимающая и передающая стороны используют одинаковые подстановки. Таким образом, в сети передачи данных абоненты с одинаковыми подстановками имеют возможность обмениваться зашифрованной информацией, что позволяет создавать выделенные группы пользователей с засекреченной связью.

Пример перестановки данных *S*-блоков [27] алгоритма ГОСТ приведен в табл. 2.2.

На следующем этапе выходы всех восьми *S*-блоков объединяются в одно 32-х битовое слово; затем это слово сдвигается циклически влево на 11 битов. На завершающем этапе результат объединяется операцией *XOR* с левой половиной шифруемого текста, создавая

новую правую половину, а правая половина становится новой левой половиной.

Таблица 2.2.

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

После выполнения этих 32 операций реализация алгоритма шифрования будет завершена. Расшифрование выполняется точно так же, как зашифрование, просто инвертируется порядок подключей K_i. Криптостойкость алгоритма ГОСТ обеспечивается за счет:

- большой длины ключа (256 бит); если учесть, что подстановки в *S*-блоках могут являться секретными, то общий объем секретной информации окажется равным 610 битам;
- 32 раундов преобразований, используемых в ГОСТ. Например, уже после восьми раундов единичное изменение входной последовательности отразится на изменении всех битов выхода.

ГЛАВА 3. АСИММЕТРИЧНЫЕ КРИПТОСИСТЕМЫ

3.1. Концепция криптосистемы с открытым ключом

Асимметричные криптосистемы являются эффективными системами криптографической защиты данных, их также называют криптосистемами с открытым ключом. Без создания открытых ключей и построения на их основе асимметричных алгоритмов шифрования было бы невозможно развитие основных типов криптографических протоколов (ключевой обмен, электронно-цифровая подпись, аутентификация и т.п.).

В асимметричных системах для зашифровывания данных используется один ключ, а для расшифрования - другой (поэтому их и называют асимметричными). Ключ, используемый для зашифрования, является открытым, поэтому может быть опубликован для использования всеми пользователями системы, которые зашифровывают данные. Для расшифрования данных получатель пользуется вторым ключом, являющимся секретным, и он не может быть определен из ключа зашифрования.

На рис. 3.1 показана обобщенная схема асимметричной криптосистемы с открытым ключом. В этой криптосистеме K_o – открытый ключ, K_c – секретный ключ получателя. Генератор ключей располагается на стороне получателя, так как это дает возможность не пересылать секретный ключ K_c по незащищенному каналу.

Расшифрование данных с помощью открытого ключа невозможно.

В зависимости от приложения отправитель использует либо секретный ключ, либо открытый ключ получателя, либо же оба, если требуется выполнить какую-то специальную криптографическую функцию. В практике шифрования

использование криптосистем с открытым ключом можно отнести к следующим категориям:

- зашифрование - расшифрование; отправитель шифрует сообщение с использованием открытого ключа получателя;

- цифровая подпись; отправитель «подписывает» сообщение с помощью своего секретного ключа. Подпись получается в результате применения криптографического алгоритма к сообщению или небольшому блоку данных, являющемуся хэш-функцией сообщения.

Асимметричные криптосистемы имеют следующие особенности:

- открытый ключ K_o и криптограмма C могут быть отправлены по незащищенным каналам, т.е. противнику известны открытый ключ и криптограмма;

- открытыми являются алгоритмы зашифрования и расшифрования

$$E : M \to C,$$
$$D : C \to M.$$

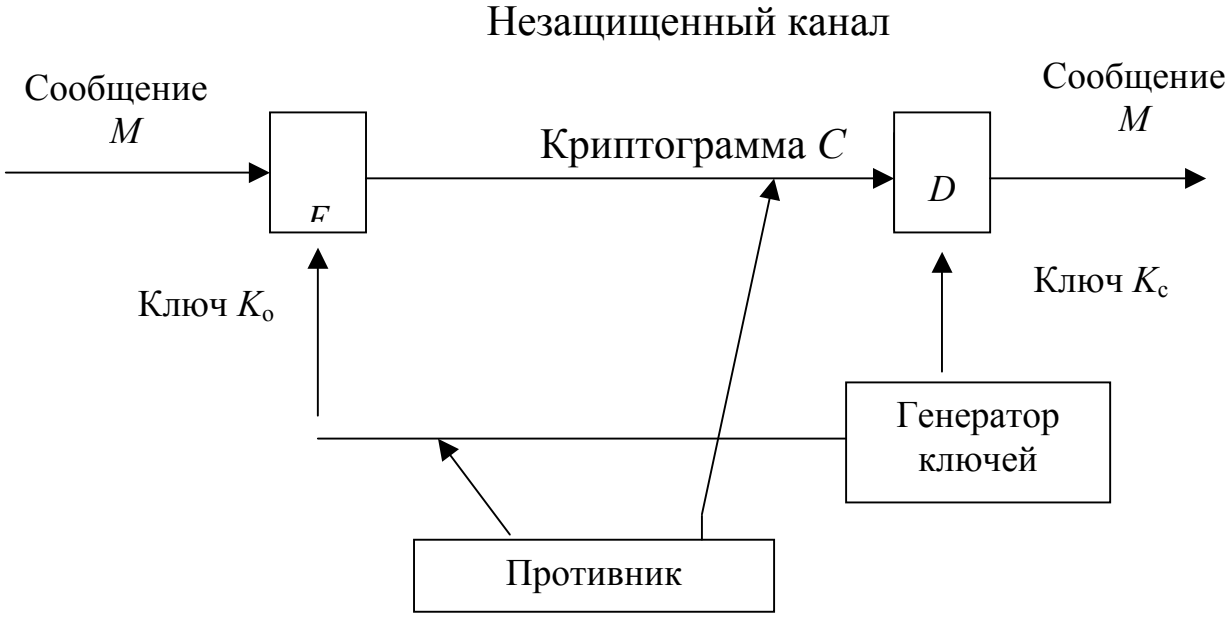

Рис. 3.1. Обобщенная схема асимметричной криптосистемы с открытым ключом

Защита информации в асимметричной криптосистеме основана на секретности ключа K_c.

Безопасность асимметричной криптосистемы обеспечивается выполнением следующих требований:

- вычисление пары ключей (K_o, K_c) должно быть простым;
- отправитель может легко вычислить криптограмму, зная открытый ключ K_o и сообщение M;

$$C = E_{K_o}(M) \qquad (3.1)$$

- получатель может легко восстановить исходное сообщение, используя секретный ключ K_c и криптограмму C

$$M = D_{K_c}(C) \qquad (3.2)$$

- при попытке вычислить секретный ключ K_c противник наталкивается на непреодолимую вычислительную проблему, даже зная открытый ключ K_o;
- при попытке вычислить исходное сообщение M противник также наталкивается на непреодолимую вычислительную проблему, даже зная пару (K_o, C).

3.2. Однонаправленные функции

Идея асимметричных криптографических систем с открытым ключом основана на применении однонаправленных функций. Однонаправленную функцию [17,22] можно определить как f: $X \rightarrow Y$, где X и Y – некоторые произвольные множества.

Функция является однонаправленной, если для всех $x \in X$ можно вычислить функцию $y = f(x)$, где $y \in Y$. Для большинства $y \in Y$ достаточно сложно получить значение $x \in X$ такое, что $f(x) = y$, т.е. эта функция отображает свои аргументы в некотором диапазоне значений так, что каждое значение функции имеет уникальное обратное значение, при этом значения функции вычислить легко, а обратные практически невозможно. Отсутствие эффективных

алгоритмов обратного преобразования $Y \to X$ является основным критерием отнесения функции f к классу однонаправленных функций.

Рассмотрим целочисленное умножение в качестве первого примера однонаправленной функции. Прямой задачей является вычисление произведения двух больших целых чисел P и Q, т.е. нахождение значения

$$N = P\,Q\,. \qquad (3.3)$$

Обратной задачей является разложение на множители большого целого числа, то есть определение делителей P и Q большого целого числа $N = P \cdot Q$, что является практически неразрешимой задачей при достаточно больших значениях N. По современным оценкам теории чисел при целом $N \approx 2^{664}$ и $P \approx Q$ для разложения числа N потребуется 10^{23} операций, что для современных вычислительных систем практически нереализуемо.

Модульная экспонента с фиксированными основанием и модулем является следующим характерным примером однонаправленной функции. Допустим, что a и N – целые числа такие, что $1 \le a < N$. Требуется определить множество $Z_N = \{0, 1, 2, \ldots, N\text{-}1\}$.

Модульная экспонента с основанием a по модулю N представляет собой функцию

$$f_{a,N}(X) = a^X (mod\ N), \qquad (3.4)$$

где X – целое число, $1 \le X \le N\text{-}1$.

Задача вычисления значения функции $f_{a,N}(x)$ называется задачей дискретного логарифмирования.

Сформулируем задачу дискретного логарифмирования следующим образом. Для известных целых a, N, Y найти целое число X такое, что

$$a^X\ mod\ N = Y.$$

Модульная экспонента считается условно однонаправленной функцией, так как алгоритм вычисления дискретного логарифма за

приемлемое время пока не найден, хотя не доказано, что не существует эффективного алгоритма вычисления дискретного логарифма за указанное время.

По современным оценкам теории чисел при целых числах $a \approx 2^{664}$ и $N \approx 2^{664}$ решение задачи дискретного логарифмирования (нахождение показателя степени X для известного Y) потребует около 10^{26} операций, т.е. эта задача имеет в 10^3 раз большую вычислительную сложность, чем задача разложения на множители. Разница в оценках сложности задач возрастает при увеличении длины чисел.

Кроме того, в настоящее время при построении криптосистем с открытым ключом широко используются так называемые однонаправленные функции с «потайным ходом». Такая функция предполагается легко вычислимой в одном направлении и практически не вычислимой в другом при отсутствии дополнительной информации. При наличии такой информации обратная функция может быть вычислена за полиномиальное время, рассматриваемой как функция длины вводимого значения.

Функция $f: X \to Y$ относится к классу однонаправленных функций с «потайным ходом» в том случае, когда она является однонаправленной и возможно эффективное вычисление обратной функции, если известен «потайной ход», то есть секретное число, строка или другая информация, ассоциирующаяся с данной функцией.

В качестве практического применения однонаправленной функции рассмотрим алгоритм шифрования данных по схеме *RSA*.

3.3. Криптосистема шифрования данных *RSA*

В настоящее время наиболее изученным методом криптографической защиты, основанным на трудности факторизации больших чисел и трудности вычисления дискретных логарифмов, является алгоритм *RSA* (названый по начальным буквам фамилий ее изобретателей Rivest, Shamir, Adleman) [25]. Этот алгоритм может работать как в режиме шифрования данных, так и в режиме электронной цифровой подписи. Схема *RSA* представляет собой блочный шифр, в котором открытый и зашифрованный тексты представляются целыми числами из диапазона от 0 до $N - 1$ для некоторого N, т.е. открытый текст шифруется блоками,

каждый из которых содержит двоичное значение, меньшее некоторого заданного числа *N*. Это значит, что длина блока должна быть меньше или равна $\log_2(N)$.

Трудность факторизации больших чисел и трудность вычисления дискретных логарифмов определяют надежность алгоритма *RSA*. В данной криптосистеме открытый ключ K_o, секретный ключ K_c, сообщение *M* и криптограмма *C* принадлежат множеству целых чисел

$$Z_N = \{0, 1, 2. \ldots, N\text{-}1\}, \qquad (3.5)$$

$$N = P \cdot Q, \qquad (3.6)$$

где *N* – модуль, *P* и *Q* являются случайными большими взаимно простыми числами. Их выбирают равной длины и хранят в секрете для обеспечения максимальной безопасности.

Множество Z_N с операциями сложения и умножения по модулю *N* образует арифметику по модулю *N*.

Открытый ключ K_o выбирают случайным образом так, чтобы выполнялись условия

$$1 < K_o \le \varphi(N), \quad \text{НОД}\,(K_o, \varphi(N)) = 1 \qquad (3.7)$$

$$\varphi(N) = (P - 1)\,(Q - 1), \qquad (3.8)$$

где $\varphi(N)$ – функция Эйлера, которая указывает количество положительных целых чисел в интервале от 1 до *N*, которые взаимно просты с *N*.

Кроме того, открытый ключ K_o и функция Эйлера $\varphi(N)$ также должны быть взаимно простыми.

Затем, используя расширенный алгоритм Евклида [17], вычисляют секретный ключ K_c такой, что

$$K_c \cdot K_o \equiv 1 (mod\ \varphi(N)) \text{ или } K_c = K_o^{-1}\ (mod\ (P - 1)\ (Q - 1)). \qquad (3.9)$$

Данное вычисление возможно, поскольку получатель знает пару простых чисел (*P*, *Q*) и может легко найти $\varphi(N)$, при этом K_c и *N* должны быть взаимно простыми. Задачу зашифрования открытого текста *M* в криптограмму *C* можно решить, используя открытый ключ K_o, по следующей формуле:

$$C = E_{K_o}(M) = M^{K_o} (mod\ N). \qquad (3.10)$$

Определение значения M по известным значениям C, K_o и N т.е. обращение функции $C = M^{K_o} (mod\ N)$, практически не осуществимо при $N \approx 2^{512}$.

Задачу расшифрования криптограммы C можно решить, используя секретный ключ K_c, по следующей формуле:

$$M = D_{K_c}(C) = C^{K_c} (mod\ N). \qquad (3.11)$$

Процесс расшифрования можно записать так:

$$D(E(M)) = M. \qquad (3.12)$$

Подставляя в (3.12) значения (3.10) и (3.11), получаем

$$(M^{K_o})^{K_c} = M\ (mod\ N)$$

или

$$M^{K_o K_c} = M\ (mod\ N). \qquad (3.13)$$

Согласно теореме Эйлера, утверждающей, что если НОД $\varphi(N) = 1$, то

$$X^{\varphi(N)} \equiv 1\ (mod\ N)$$

или в несколько более общей форме

$$X^{n\varphi(N)+1} \equiv X\ (mod\ N). \qquad (3.14)$$

Сравнивая выражения (3.13) и (3.14), получаем

$K_o \cdot K_c = n \cdot \varphi(N) + 1$ **или** $K_o \cdot K_c \equiv 1 (mod\ \varphi(N))$.

Поэтому для вычисления секретного ключа K_c применяют последнее соотношение. Откуда следует, что если криптограмму

$C = M^{K_o} (mod\ N)$ **возвести в степень K_c, то в результате восстанавливается исходный открытый текст M, так как**

$$(M^{K_o})^{K_c} = M^{K_o K_c} = M^{n\varphi(N)+1} \equiv M\ (mod\ N).$$

Таким образом, получатель, который создает криптосистему, защищает два параметра: секретный ключ K_c и пару чисел P и Q, произведение которых даёт модуль N. Одновременно публикует значения модуля N и открытого ключа K_o, поэтому противнику известны только значения K_o и N. Если бы криптоаналитик смог разложить число N на множители P и Q, то он узнал бы тройку чисел (P,Q, K_o), значение функции Эйлера $\varphi(N) = (P-1)(Q-1)$ и определил значение секретного ключа K_c.

Однако, как отмечалось выше, разложение большого числа N на множители вычислительно не осуществимо при длине выбранных P и Q не менее 100 десятичных знаков.

Рассмотрим более подробно схему алгоритма *RSA*.

Чтобы использовать алгоритм *RSA*, надо сначала сгенерировать открытый и секретный ключи, выполнив следующие шаги:

- **выбрать два больших простых числа *P* и *Q*;**
- **определить *N* как результат умножения *P* на *Q* ($N = P \cdot Q$);**
- **выбрать большое число K_c. Оно должно быть взаимно простым с результатом умножения $\varphi(N) = (P-1)(Q-1)$;**
- **определить такое число K_o, для которого выполняется следующее соотношение: $K_o K_c \ (mod\ \varphi(N)) = 1$;**
- **объявить открытым ключом числа (K_o и *N*), а секретным ключом (K_c и *N*).**

Затем, чтобы зашифровать данные по известному ключу (K_c , *N*), необходимо разбить шифруемый текст на блоки, каждый из которых может быть представлен в виде числа M_i, (i=1…N-1); зашифровать текст, рассматриваемый как последовательность чисел M_i по формуле

$$C_i = M_i^{K_o} mod\ (N).$$

Чтобы расшифровать эти данные, используя секретный ключ (K_c, *N*), необходимо выполнить вычисления:

$$M_i = C_i^{K_c} mod\ (N).$$

В результате будет получено множество чисел M_i, которое представляет собой исходный текст.

Рассмотрим пример [17,24] использования метода *RSA* для шифрования сообщения 651.

Для удобства расчетов будем использовать числа небольшой разрядности.

1. **Выберем $P = 3$ и $Q = 11$.**
2. **Определим $N = 3 \cdot 11 = 33$.**
3. **Найдем $\varphi(N) = (P-1)(Q-1) = 20$.**

Следовательно, в качестве K_c выберем любое число, являющееся взаимно простым с 20, например $K_c = 3$.

4. **Выберем число K_o. В качестве такого числа может быть взято любое число, для которого справедливо соотношение**

$K_o \cdot 3 mod(20) = 1$, например $K_o = 7$.

5. **Представим шифруемое сообщение как последовательность целых чисел в диапазоне 0, . . . ,32.**

Например, в виде 651. Зашифруем данное сообщение, используя ключ K_o =7 и N=33 по формуле

$C_i = M_i (mod\ N)$:

$C_1 = 6^7\ mod\ (33) = 279936\ mod\ (33) = 30;$

$C_2 = 5^7\ mod\ (33) = 78125\ mod\ (33) = 14;$

$C_3 = 1^7\ mod\ (33) = 1\ mod\ (33) = 1.$

Получаем криптограмму (30,14,1).

Расшифровать полученное сообщение можно, используя секретный ключ K_c по формуле

$M_i = C_i^{K_c} (mod\ N)$

$M_1 = 30^3\ mod\ (33) = 27000\ mod\ (33) = 6;$

$M_2 = 14^3\ mod\ (33) = 2744\ mod\ (33) = 5;$

$M_3 = 1^3\ mod\ (33) = 1\ mod\ (33) = 1.$

Таким образом, в результате расшифрования криптограммы C получено исходное сообщение «651».

Криптостойкость алгоритма *RSA* основывается на предположении исключительной трудоёмкости определения секретного ключа по открытому, так как для этого необходимо решить задачу существования делителей целого числа.

Эта задача не имеет до настоящего времени эффективного решения. Для чисел, состоящих из 200 и более цифр (а именно такие числа рекомендуется использовать), традиционные методы требуют выполнения огромного числа операций (около 10^{23}).

Быстродействие аппаратной реализации *RSA* примерно в 1000 раз ниже, чем быстродействие аппаратной реализации симметричной криптосистемы *DES*.

Шифрование *RSA* выполняется намного эффективнее, если правильно выбрать значение K_o. Чаще всего используются 3, 17 или $65537 = 2^{16} + 1$ – двоичное представление этого числа содержит только две единицы, поэтому для возведения в степень нужно выполнить лишь 17 умножений.

Использование в качестве K_0 любого из указанных значений не влияет на криптостойкость, если даже одно и то же значение K_0 используется группой пользователей.

3.4. Асимметричные криптосистемы на базе эллиптических кривых

3.4.1. *Основные понятия и определения*

На базе эллиптических кривых *E* можно реализовать не только криптоалгоритмы асимметричного шифрования, но и выработки общего секретного ключа для симметричного шифрования. Криптосистемы на базе эллиптических кривых [18,19,21,23] позволяют использовать существенно меньшие размеры ключей по сравнению с другими криптоалгоритмами при сохранении одинакового уровня криптостойкости. Для перечисленных выше реализаций используются эллиптические кривые над полями Галуа *GF* (*p*) с конечным числом *p* элементов двух видов:

- эллиптическая кривая над конечным полем типа $E(GF(p))$, где p - некоторое простое число;
- эллиптическая кривая над конечным полем типа $E(GF(2^m))$, где $p = 2^m$.

Эллиптическая кривая E в конечном поле E $(GF(p))$ по модулю p определяется соотношением

$$y^2 = x^3 + ax + b \ (mod\ p), \qquad (3.15)$$

при этом a и b должны удовлетворять неравенству

$$4a^3 + 27b^2 (mod\ p) \neq 0 \ \text{ и } \ p > 3.$$

Представление эллиптической кривой E в виде уравнения (3.15) носит название эллиптической кривой в форме Вейерштрасса.

Эллиптической кривой $E(GF(p))$ является группа решений (x, y), $x \in GF(p)$, $y \in GF(p)$ соотношения (3.15) при определенных значениях a и b, а также дополнительная точка неопределенности 0.

К данной группе применимы следующие правила:

1. $0 + 0 = 0$.

2. $(x, y) + 0 = (x, y)$ для всех $(x, y) \in E(GF(p))$, то есть 0 является нулевым элементом группы.

3. $(x, y) + (x, -y) = 0$, то есть точки (x, y) и $(x, -y)$ являются взаимно обратными.

4. Правило сложения двух разных и не взаимно обратных точек.

Для всех $(x_1, y_1) \in E(GF(p))$ и $(x_2, y_2) \in E(GF(p))$, удовлетворяющих условию $x_1 \neq x_2$:

$$(x_1, y_1) + (x_2, y_2) = (x_3, y_3),$$

где значения x_3 , y_3 и λ в дальнейшем вычисляются по модулю p:

$$x_3 = \lambda^2 - x_1 - x_2;$$

$$y_3 = \lambda (x_1 - x_3) - y_1;$$

$$\lambda = (y_2 - y_1) / (x_2 - x_1).$$

5. Правило удвоения точки.

Для всех $(x_1, y_1) \in E(GF(p))$, удовлетворяющих условию $y_1 \neq 0$,
Результат удвоения любой точки определяется по формулам:

$$2(x_1, y_1) = (x_3, y_3);$$

$$x_3 = \lambda^2 - 2x_1;$$

$$y_3 = \lambda (x_1 - x_3) - y_1;$$

$$\lambda = (3x_1^2 + a) / 2y_1.$$

Операции сложения коммутативны, поэтому $E(GF(p))$ представляет собой абелеву группу.

Эллиптическая кривая характеристики 2 в конечном поле $GF(2^m)$ определяется соотношением по модулю p:

$$y^2 + xy = x^3 + ax^2 + b \ (mod\ p) \text{ при } b \neq 0. \qquad (3.16)$$

Эллиптической кривой $E(GF(2^m))$ является группа решений (x, y), $x \in GF(2^m)$, $y \in GF(2^m)$ соотношения (3.16) при определенных a и b, а также дополнительная точка неопределенности 0.

К данной группе применимы следующие правила:

1. $0 + 0 = 0$.

2. $(x, y) + 0 = (x, y)$ для всех $(x, y) \in E(GF(2^m))$.

3. $(x, y) + (x, x + y) = 0$, то есть точки (x, y) и $(x, x + y)$ являются взаимно обратными.

4. Правило сложения двух разных и не взаимно обратных точек.

Для всех $(x_1, y_1 \in E(GF(2^m))$ и $(x_2, y_2) \in E(GF(2^m))$, удовлетворяющих условию $x_1 \neq x_2$:

$$(x_1, y_1) + (x_2, y_2) = (x_3, y_3),$$

где значения x_3, y_3, λ вычисляются по *mod p*

$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a;$$

$$y_3 = \lambda (x_1 + x_3) + x_3 + y_1;$$

$$\lambda = (y_1 + y_2) / (x_1 - x_2).$$

5. Правило удвоения точки.

Для всех $(x_1, y_1) \in E(GF(2^m))$, удовлетворяющих условию $x_1 \neq 0$, результат удвоения любой точки определяется по формулам:

$$2(x_1 y_1) = (x_3, y_3);$$

$$x_3 = \lambda^2 + \lambda + a;$$

$$y_3 = x_1^2 + (\lambda + 1) x_3;$$

$$\lambda = x_1 + y_1 / x_1.$$

Аналогично группе $E(GF(p))$ операции сложения являются коммутативными, поэтому $E(GF(2^m))$ представляет собой абелеву группу. С помощью описанных выше правил сложения можно вычислить точку kP для любого целого числа k и любой точки P эллиптической кривой. Однако решение обратной задачи - нахождение числа k по известным точкам P и kP - является трудноразрешимой проблемой. Решение проблемы дискретного логарифма эллиптической кривой является значительно более сложным, чем проблема дискретного логарифмирования (т.е.

нахождение числа x по заданному числу $y = g^x \pmod n$ при известном основании g и модуле n), на котором базируются *RSA* и другие подобные асимметричные криптосистемы. Сложность решения данной проблемы обусловлена ресурсоемкостью операций сложения и дублирования точек, с помощью которых вычисляется kP. Откуда следует возможность применения более коротких ключей.

3.5. Алгоритм асимметричного шифрования на базе эллиптических кривых *ECES*

В алгоритме *ECES* (Elliptic Curve Encryption Scheme) на первом этапе должны быть определены параметры, являющиеся общей открытой информацией для всех пользователей системы [8,15,22]:

- конечное поле $GF(p)$;
- эллиптическая кривая $E(GF(p))$;
- большой простой делитель количества точек кривой n;
- точка G, координаты которой имеют порядок, что и число n.

Каждый пользователь системы генерирует пару ключей следующим образом:

- выбирает случайное целое число K_c, $1 < K_c < n$ -1;
- вычисляет точку $K_o = K_c\, G$.

При этом секретным ключом пользователя является число K_c, открытым ключом - точка K_o. Кроме того, сообщение M разбивается на блоки длиной $2L$ - 16 бит, где L равно ближайшему большему целому от $\log_2 p$;

- выбирается случайное целое число k, $1 < k < n$ - 1;
- вычисляется точка $(x_1, y_1) = kG$;
- вычисляется точка $(x_2, y_2) = k\, K_o$;

Известно несколько подходов [22, 23] к зашифрованию - расшифрованию информации, предполагающих использование

эллиптических кривых. В этом разделе мы рассмотрим наиболее простой из этих подходов. Как было изложено выше, зашифрованное сообщение пересылается в виде значения (x, y) для точки P_m . Здесь точка P_m будет представлять зашифрованный текст и впоследствии будет расшифроваться. В качестве параметров данной криптосистемы рассматривается точка G и эллиптическая группа $E_p(a, b)$.

Пользователи A и B выбирают секретные ключи K_{cA} и K_{cB}, а также генерируют открытые ключи $K_{oA} = K_{cA}\ G$ и $K_{oB} = K_{cB}\ G$ соответственно. Чтобы зашифровать сообщение P_m, пользователь A выбирает случайное положительное целое число k и вычисляет криптограмму C_m с помощью открытого ключа стороны B, - K_{oB}, состоящую из двух точек

$$C_m = \{(kG),\ (P_m + k\ K_{oB}\ )\}.$$

Чтобы расшифровать эту криптограмму, пользователь B умножает первую точку (kG) на свой секретный ключ K_{cB} и вычитает результат из второй точки:

$$(P_m + k\ K_{oB}) - K_{cB}\ (kG) = P_m + k(K_{cB}\ G) - K_{cB}\ (kG) = P_m\ .$$

Пользователь A замаскировал сообщение P_m с помощью добавления к нему маски $k\ K_{oB}$. Однако следует заметить, что никто не сможет убрать маску $k\ K_{oB}$, кроме пользователя, который знает значение k и имеет личный ключ K_{cB}. Противнику для восстановления сообщения придется вычислить k по данным G и kG, что является трудной задачей. В качестве примера [22] возьмем $p = 751$, $E_P = (-1, 188)$ и $G = (0, 376)$.

Все расчеты в данном примере выполняются по модулю p. Предположим, что пользователь A отправляет пользователю B сообщение, которое кодируется точкой $P_m = (562, 201)$, и выбирает случайное число $k = 386$. Открытым ключом B является $K_{oB} = (201, 5)$. Мы имеем $386(0, 376) = (676, 558)$ и $(562, 201) + 386(201, 5) = (385, 328)$.

Таким образом, пользователь *A* должен послать зашифрованный текст {(676, 558), (385, 328)}.

ГЛАВА 4. ИДЕНТИФИКАЦИЯ И ПРОВЕРКА ПОДЛИННОСТИ СООБЩЕНИЯ

4.1. Основные понятия и определения

Любой объект компьютерной системы (КС) может быть идентифицирован числом, словом или строкой символов. Если объект имеет некоторый идентификатор, зарегистрированный в сети, его называют законным (легальным), другие объекты относятся к незаконным (нелегальным).

Идентификация объекта выполняется в том случае, когда объект делает попытку войти в сеть. Если идентификация прошла успешно, то данный объект является законным для данной сети.

Аутентификация объекта (проверка подлинности) устанавливает, является ли данный объект именно тем, кем себя объявляет.

Предоставление полномочий (авторизация) устанавливает сферу деятельности объекта и доступные ему ресурсы КС, после идентификации и подтверждения его подлинности.

Приведенные выше процедуры инициализации являются процедурами защиты и относятся к одному объекту КС [20,21].

Аутентификация объектов при защите каналов передачи данных означает взаимное установление подлинности объектов, связанных между собой по линии связи. Эта процедура выполняется обычно в начале сеанса в процессе установления логической связи между двумя объектами сети (соединение), так называемых абонентов. В конечном итоге обеспечивается уверенность в том, что соединение установлено с законным объектом и вся информация дойдет до места назначения.

При обмене сообщениями необходимо обеспечить выполнение следующих требований защиты:

- получатель должен быть уверен в подлинности источника данных;
- получатель должен быть уверен в подлинности передаваемых данных;
- отправитель должен быть уверен в доставке данных получателю;
- отправитель должен быть уверен в подлинности доставленных данных.

Для выполнения первых двух требований средством защиты является цифровая подпись. Для выполнения последующих требований отправитель должен получить уведомление о вручении с помощью удостоверяющей почты. Средством защиты в такой процедуре является цифровая подпись подтверждающего ответного сообщения, которое, в свою очередь, является доказательством пересылки исходного сообщения.

При реализации всех четырех требований в КС гарантируется защита данных при их передаче по каналу связи и обеспечивается функция защиты, которая называется функцией подтверждения (неоспоримости) передачи. Тогда отправитель не может отрицать ни факта посылки сообщения, ни его содержания, а получатель не может отрицать ни факта получения сообщения, ни подлинности его содержания.

4.2. Идентификация и аутентификация пользователя

Чтобы получить допуск к ресурсам КС, пользователь должен пройти идентификацию (то есть сообщить системе по её запросу свое имя), затем аутентификацию (то есть вводить в КС уникальную, не известную другим пользователям информацию о

себе). Для этого необходимо наличие соответствующего субъекта (модуля) аутентификации и аутентифицирующего объекта, хранящего уникальную информацию для аутентификации пользователя [3,20,21].

Различают несколько форм представления объектов, аутентифицирующих пользователя, а именно на основе знания каких-либо данных. Примером могут служить стандартные пароли, персональные идентификационные номера (*PIN*), а также секретные и открытые ключи, знание которых демонстрируется в протоколах типа «запрос-ответ»:

- на основе средств хранения; обычно это магнитные карты, смарт-карты, touch memory и персональные генераторы, которые используются для создания одноразовых паролей;

- на основе биометрических характеристик; здесь включаются методы, базирующиеся на проверке пользовательских биометрических характеристик (голос, сетчатка глаза, отпечатки пальцев и т.д.). В данном случае не используются криптографические методы и средства. Аутентификация на основе биометрических характеристик применяется для контроля доступа в помещения либо к какой-либо технике.

Протоколы аутентификации можно классифицировать по уровню обеспечиваемой безопасности или по возможности противостоять определенному классу атак: простая аутентификация (на основе использования паролей), строгая аутентификация (на основе использования криптографических методов и средств) и протоколы, обладающие свойством доказательства с нулевым значением.

Каждый из перечисленных типов аутентификации способствует решению своих специфических задач, поэтому все они используются на практике, за исключением протоколов аутентификации, обладающих свойством доказательства с нулевым

знанием, интерес к которым носит пока чисто теоретический характер.

Оценка уровня безопасности [3], обеспечиваемого протоколом аутентификации, может быть определена по отношению к конкретным типам атак, каковыми являются:

- самозванство, когда один пользователь пытается выдать себя за другого;
- повторная передача, заключающаяся в повторной передаче аутентифицированных данных каким-либо пользователем;
- подмена стороны аутентификационного обмена, при этом злоумышленник в ходе атаки участвует в процессе аутентификационного обмена между двумя сторонами и имеет возможность изменения проходящего через него трафика;
- отражение передачи, в ходе которой злоумышленник в рамках данного протокола пересылает обратно перехваченную информацию;
- вынужденная задержка, когда злоумышленник перехватывает информацию и передает её через некоторое время;
- атака с выборкой текста, при которой злоумышленник перехватывает аутентификационный трафик и пытается получить информацию о долговременных ключах.

Наиболее опасной угрозой протоколам аутентификации является случай, когда злоумышленник выдает себя за какую-либо другую сторону, обладающую существенными привилегиями в системе. Кроме того, существует атака такого вида, когда после успешного прохождения аутентификации между двумя пользователями и установления соединения нарушитель «убирает» какого-либо пользователя из соединения и продолжает работу от его имени.

Подобного рода атаки могут быть устранены следующими приемами:

- периодическое выполнение процедур аутентификации в рамках уже установленного сеанса связи;

- привязка результата аутентификации к последующим действиям пользователей в рамках системы (например, аутентификационный обмен секретными сеансовыми ключами, с использованием которых осуществляется дальнейшее взаимодействие пользователей).

Протоколы аутентификации для атак, описанных выше, имеют следующие свойства:

- взаимная аутентификация, отражающая необходимость обоюдной аутентификации между сторонами аутентифицированного обмена;

- вычислительная эффективность – это количество операций, необходимых для выполнения протоколов;

- коммуникационная эффективность, отражающая количество сообщений и их длину, необходимую для осуществления аутентификации;

-наличие третьей стороны; например, доверенный сервер распределения симметричных ключей или сервер, реализующий дерево сертификатов для распределения открытых ключей;

- основа гарантий безопасности; этому могут служить протоколы, обладающие свойством доказательства с нулевым знанием;

- хранение секрета; определяется способ реализации хранения критичной ключевой информации.

4.3. Типовые схемы идентификации и аутентификации пользователя

Пусть в КС зарегистрировано n пользователей и i-й аутентифицирующий объект i-го пользователя содержит два информационных поля:

Id_i – неизменный идентификатор i-го пользователя, который является аналогом имени и используется для идентификации пользователя;

K_i – аутентифицирующая информация пользователя, которая может изменяться и служит для аутентификации (например, пароль $P_i = K_i$).

Подобная структура соответствует практически любому ключевому носителю информации, который используется для опознания пользователя. Так, для носителей типа пластиковых карт выделяется неизменяемая информация Id_i первичной персонализации пользователя и объект в файловой структуре карты, содержащий K_i.

Первичной аутентифицирующей информацией i-го пользователя можно назвать совокупную информацию в ключевом носителе. Внутренний аутентифицирующий объект не должен существовать больше времени работы конкретного пользователя. Для длительного хранения следует использовать данные в защищённой форме.

Ниже приведены две типовые схемы идентификации и аутентификации [3,17,21].

Согласно алгоритму по схеме 1 в КС выделяется объект-эталон для идентификации и аутентификации пользователей. Структура объекта-эталона для схемы 1 показана в табл. 4.1. Здесь $E_i = F(Id_i, K_i)$, где F – функция, которая обладает свойством «невосстановимости» значения K_i по E_i и Id_i.

«Невосстановимость» K_i оценивается некоторой трудоемкостью T_o решения задачи восстановления аутентифицирующей информации K_i по E_i и Id_i.

Для пары K_i и K_j возможно совпадение соответствующих значений E. Поэтому вероятность ложной аутентификации пользователя не должна быть больше некоторого порогового

значения P_o. На практике задают $T_o = 10^{20},\ldots,10^{30}$, $P_o = 10^{-7},\ldots,10^{-9}$ [15,21].

Таблица 4.1

Номер пользователя	Информация для идентификации	Информация для аутентификации
1	Id_1	E_1
2	Id_2	E_2
. . .	. . .	. . .
N	Id_n	E_n

4.3.1. *Протокол идентификации и аутентификации по схеме 1*

1. Сначала пользователь предъявляет свой идентификатор Id.
2. При несовпадении Id ни с одним из Id_i, которые зарегистрированы в КС, идентификатор отвергается и пользователь не допускается к работе. В случае $Id = Id_i$, пользователь прошёл идентификацию.
3. Субъект аутентификации запрашивает у пользователя его аутентификатор K и вычисляет значение $Y = F(Id_i, K)$.
4. Субъект аутентификации производит сравнение значений Y и E_i. При равенстве этих значений устанавливается, что пользователь успешно аутентифицирован в системе. Информация об этом пользователе передаётся в программные модули, использующие ключи пользователей. В противном случае пользователь не допускается к работе.

Данную схему идентификации и аутентификации пользователя можно модифицировать по схеме 2.

Согласно схеме 2 в КС выделяется модифицированный объект-эталон, структура которого показана в табл. 4.2.

Таблица 4.2

Номер пользователя	Информация для идентификации	Информация для аутентификатора
1	Id_1, S_1	E_1
2	Id_2, S_2	E_2
. . .	. . .	. . .
N	Id_n, S_n	E_n

В схеме 2 значение E_i равно $F(S_i, K_i)$, где S_i – случайный вектор, который задаётся при создании идентификатора пользователя; F - функция, обладающая свойством «невосстановимости» значения K_i по E_i и S_i.

4.3.2. *Протокол идентификации и аутентификации по схеме 2*

1. Пользователь предъявляет свой идентификатор *Id*.
2. При несовпадении *Id* ни с одним из Id_i, которые зарегистрированы в КС, идентификатор отвергается и пользователь не допускается к работе. В случае $Id = Id_i$, пользователь прошёл идентификацию.
3. По идентификатору Id_i выделяется вектор S_i.
4. Субъект аутентификации запрашивает у пользователя его аутентификатор K и вычисляет значение $Y = F(S, K_i)$.
5. Субъект аутентификации производит сравнение значений Y и E_i. При равенстве этих значений устанавливается, что пользователь успешно аутентифицирован в системе. В противном случае пользователь не допускается к работе.

Необходимым требованием устойчивости схем аутентификации к восстановлению информации K_i является случайный равновероятный выбор K_i из множества возможных значений.

Особенности применения пароля для аутентификации пользователя.

Каждый пользователь КС получает идентификатор и пароль, который в начале сеанса он предъявляет системе. На рис. 4.1 проиллюстрирован метод подтверждения подлинности с использованием пароля, основанный на сравнении представляемого пользователем пароля P_A с исходным значением P'_A, хранящимся в компьютерном центре. Так как пароль храниться в тайне, то перед пересылкой по незащищенному каналу он должен шифроваться. Если P_A и P'_A совпадают, то пароль P_A считается подлинным, а пользователь – законным [3,17,21].

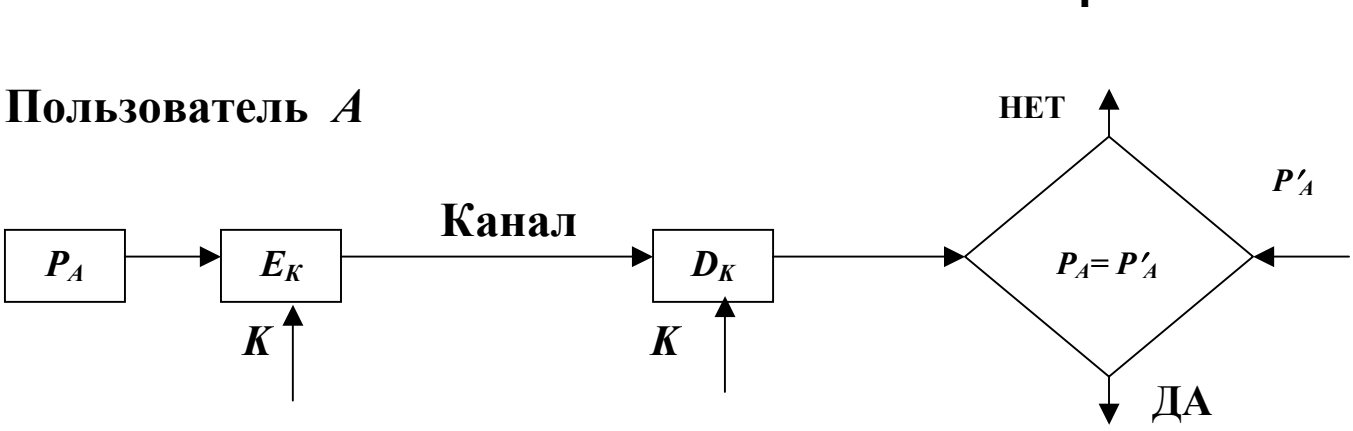

Рис. 4.1. Схема простой аутентификация с помощью пароля

Доступ в такую систему может получить нарушитель, если узнает каким-либо образом пароль и идентификатор законного пользователя.

Если получатель не должен раскрыть исходную открытую форму пароля, то отправитель должен переслать вместо открытой формы пароля отображение пароля, получаемое с использованием односторонней функции $\alpha(\cdot)$ пароля, которое должно гарантировать

невозможность раскрытия противником пароля по его отображению и нарушитель наталкивается на неразрешимую задачу.

Функция $\alpha(\cdot)$ может быть рассчитана по формуле

$$\alpha(P) = E_P\,(Id),$$

где P – пароль отправителя; Id – идентификатор отправителя; E_P – процедура шифрования, выполняемая с использованием пароля P в качестве ключа.

Если длина ключа и пароля одинаковы, то такие функции особенно удобны, так как в случае подтверждения подлинности с помощью пароля, то процесс идентификации состоит из пересылки получателю отображения $\alpha(P)$ и сравнения его с предварительно вычисленным и хранимым эквивалентом $\alpha'(P)$.

Чтобы пользователь запомнил пароль, он должен быть коротким, однако это делает пароль уязвимым к атаке полного перебора всех вариантов. Для предотвращения такой атаки функцию $\alpha(P)$ определяют как

$$\alpha(P) = E_{P \oplus K}\,(Id),$$

где K и Id – соответственно ключ и идентификатор отправителя.

Подтверждение подлинности пароля состоит из сравнения двух отображений $\alpha(P_A)$ и $\alpha'(P_A)$ и признания пароля P_A, если эти отображения равны.

4.3.3. *Идентификация и аутентификация пользователя по биометрическим признакам*

В настоящее время всё большее распространение получает биометрическая идентификация и аутентификация пользователя, при которых успешно идентифицируется потенциальный пользователь путем измерения физиологических параметров и характеристик человека, особенностей его поведения.

По сравнению с традиционными биометрические методы имеют следующие основные достоинства [5, 22, 5]:

- высокая степень достоверности идентификации по биометрическим признакам из-за их уникальности;
- неотделимость биометрических признаков от дееспособности личности;
- трудность фальсификации биометрических признаков.

Биометрическими признаками, используемыми при идентификации потенциального пользователя, являются: узор радужной оболочки и сетчатки глаз; отпечатки пальцев; геометрическая форма руки; форма и размеры лица; особенности голоса; биомеханические характеристики рукописной подписи; биомеханические характеристики «клавиатурного почерка».

При регистрации пользователь демонстрирует один или несколько своих характерных биометрических признаков, которые регистрируются системой как контрольный «образ» законного пользователя. Этот образ хранится в электронной форме и используется для проверки идентичности каждого, кто выдает себя за законного пользователя. В зависимости от совпадения или несовпадения совокупности предъявленных признаков с зарегистрированными в контрольном образе, предъявивший их считается законным пользователем или нет.

Системы идентификации по узору радужной оболочки и сетчатки глаз разделяются на два класса:

- использующие рисунок радужной оболочки глаза;
- использующие рисунок кровеносных сосудов сетчатки глаза.

Эти системы считаются наиболее надёжными среди всех биометрическим систем и используются там, где требуется высокий уровень безопасности, так как вероятность повторения данных параметров равна 10^{-78}.

Самыми распространенными являются системы идентификации по отпечаткам пальцев, так как имеются большие банки данных по отпечаткам пальцев. Основными пользователями подобных систем во всём мире являются полиция, различные государственные и некоторые банковские организации.

Системы идентификации по геометрической форме руки используют сканеры формы руки, устанавливаемые на стенах. Большинство пользователей предпочитает именно системы идентификации такого типа.

Системы идентификации по лицу и голосу являются наиболее доступными из-за их дешевизны, так как современные компьютеры имеют видео- и аудиосредства. Такие системы применяются при удалённой идентификации субъекта доступа в телекоммуникационных сетях.

Системы идентификации личностей по динамике рукописной подписи учитывают интенсивность каждого усилия подписывающего, частотные характеристики каждого элемента подписи и начертания подписи в целом.

Системы идентификации по биомеханическим характеристикам «клавиатурного почерка» основываются на том, что моменты нажатия и отпускания клавиш при наборе текста на клавиатуре существенно различаются у разных пользователей. Этот «клавиатурный почерк» позволяет построить достаточно надёжные средства идентификации. В случае обнаружения изменения клавиатурного почерка пользователя ему автоматически запрещается работа на ЭВМ.

Применение систем биометрической идентификации не получило надлежащего нормативно-правового обеспечения в виде стандартов, поэтому они используются только в автоматизированных системах, обрабатывающих и хранящих

персональные данные, составляющие коммерческую или служебную тайну.

4.4. Взаимная проверка подлинности пользователей

Процесс взаимной аутентификации выполняется в начале сеанса связи двумя способами либо механизмом запроса-ответа, либо механизмом отметки времени («временный штемпель»). В обоих случаях для защиты механизма контроля следует применять шифрование.

Если проверка подлинности проводится методом запроса-ответа, то пользователь *A* включает в посылаемое для пользователя *B* сообщение непредсказуемый элемент – запрос *X* (например, случайное число). При ответе *B* должен выполнить над *X* некоторую операцию (например, вычислить функцию $f(X)$). Заранее это осуществить невозможно, так как пользователю *B* неизвестно, какое случайное число *X* придёт в запросе. Получив ответ с результатом действий *B*, пользователь *A* может быть уверен, что пользователь *B* – подлинный. Недостатком данного метода является возможность установления закономерности между запросом и ответом.

При механизме отметки времени регистрируется время для каждого сообщения, при этом каждый пользователь сети может определить, насколько «устарело» пришедшее сообщение, и решить не принимать его, поскольку оно может быть ложным. Возникает проблема допустимого временного интервала задержки для подтверждения подлинности сеанса, поскольку сообщение с «временным штемпелем» не может быть передано мгновенно, а компьютерные часы отправителя и получателя не могут быть абсолютно синхронизированы. Поэтому для взаимной проверки подлинности обычно используют процедуру «рукопожатия»,

заключающуюся во взаимной проверке ключей, используемых сторонами.

Процедура рукопожатия обычно выполняется в самом начале сеанса связи между двумя сторонами в компьютерных сетях. Достоинством такой процедуры является то, что ни один из участников сеанса связи не получает никакой секретной информации во время процедуры подтверждения подлинности.

В некоторых случаях пользователи хотят иметь непрерывную проверку подлинности отправителей в течение всего сеанса связи. Один из простейших способов непрерывной проверки подлинности показан на рис. 4.2 [3, 21]. Передаваемая криптограмма имеет вид

$$E_K (Id_A, M),$$

где Id_A – идентификатор отправителя A; M – сообщение.

Получатель B, принявший эту криптограмму, расшифровывает её и раскрывает пару (Id_A, M). Если идентификатор Id_A совпадает с хранимым значением Id'_A, получатель B признает эту криптограмму.

Другой вариант непрерывной проверки подлинности использует вместо идентификатора отправителя его секретный пароль. Пароли подготовлены заранее и известны обеим сторонам. Пусть P_A и P_B – пароли пользователей A и B соответственно. Пользователь A создает криптограмму $C = E_K (P_A, M)$.

Получатель расшифровывает криптограмму и сравнивает пароль, извлечённый из этой криптограммы, с исходным значением. При их равенстве получатель признаёт эту криптограмму.

Рассмотренная выше процедура рукопожатия дана в предположении, что пользователи A и B уже имеют общий секретный ключ.

На самом деле реальные процедуры предназначены для распределения ключей между подлинными партнёрами и включают

этап распределения ключей и этап собственно подтверждения подлинности партнёра по информационному обмену.

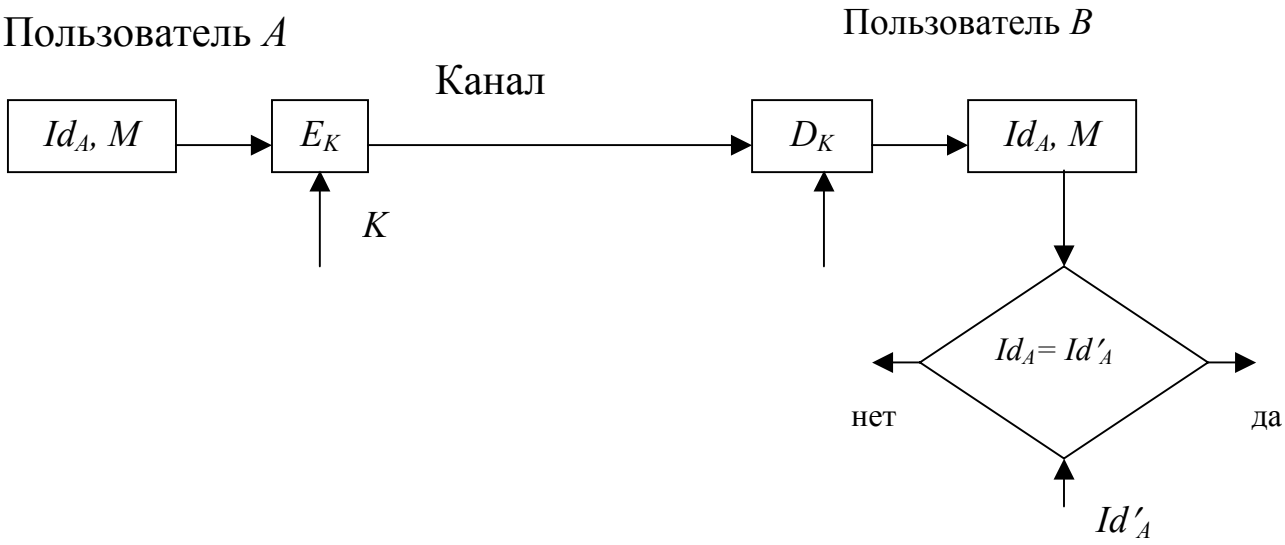

Рис. 4.2. Схема непрерывной проверки подлинности отправителя

4.5. Типовые схемы идентификации и аутентификации пользователя

4.5.1. *Протоколы идентификации с нулевой передачей знаний*

Интеллектуальные карты (смарт-карты) широко распространены для различных коммерческих, гражданских и военных применений, что привело к необходимости обеспечения безопасной идентификации этих карт и их владельцев. Здесь главной проблемой является оперативное обнаружение обмана и отказа обманщику в допуске, ответе или обслуживании.

В настоящее время разработаны протоколы идентификации с нулевой передачей знаний [13] для безопасного использования смарт-карт. Неотъемлемым признаком владельца такой карты

становится секретный ключ, и доказательство знания секретного ключа с нулевой передачей этого знания является доказательством подлинности личности владельца смарт-карты.

4.5.2. *Упрощённая схема идентификации с нулевой передачей знаний*

Наиболее известным доказательством идентичности с нулевой передачей конфиденциальной информации является схема, которую в 1986 г. предложили У. Фейге, А. Фиат и А. Шамир [1,25].

Для более чёткого выявления основной концепции схемы рассмотрим ее упрощённый вариант. Сначала выбирают случайное значение модуля n, который является произведением двух больших чисел, и его модуль длиной не менее 512,…,1024 бит. Значение n предоставляется группе пользователей, доказывающей свою подлинность. В процессе участвуют:

- сторона A, доказывающая свою подлинность;
- сторона B, проверяющая предоставляемое стороной A доказательство.

Доверенный арбитр для генерации открытого и секретного ключей выбирает некоторое число V, являющееся квадратичным вычетом по модулю n, такое, что сравнение

$$x^2 \equiv V \,(mod\ n)$$

имеет решение и существует целое число $V^{-1}\ mod\ n$. Здесь V является открытым ключом для пользователя A. Затем вычисляют наименьшее значение S, для которого

$$k \equiv sqrt\,(V^{-1})\ (mod\ n),$$

где k –является секретным ключом для A.

В дальнейшем приступают к выполнению протокола идентификации. С этой целью:

1. Пользователь A выбирает случайное число r, $r < n$, вычисляет

$$x = r^2 \ mod \ n$$

и отправляет x стороне B.

2. Пользователь B посылает A случайную последовательность битов b.

3. Пользователь A отправляет r пользователю B, если $b = 0$. Если $b = 1$, то пользователь A отправляет B

$$y = r \ S \ mod \ n.$$

4. Если $b = 0$, пользователь B проверяет

$$x = r^2 \ mod \ n$$

для того, чтобы убедиться, что пользователь A знает $sqrt(x)$. Если $b = 1$, пользователь B проверяет

$$x = y^2 \ V \ mod \ n,$$

для того, чтобы знать, что пользователь A знает $sqrt\ (V^{-1})$.

Подобные шаги образуют один цикл протокола, называемый аккредитацией. Пользователи A и B повторяют этот цикл t раз при различных значениях r и b до тех пор, пока пользователь B не убедится, что пользователь A знает значение S.

Вероятность того, что пользователь A обманет пользователя B в одном цикле, равна 1/2. Вероятность обмануть B в t циклах составляет $(1/2)^t$.

Для работы этого протокола необходимо, чтобы пользователь A никогда повторно не использовал значение r.

4.5.3. Параллельная схема идентификации с нулевой передачей знаний

Параллельная схема уменьшает длительность процесса идентификации за счет увеличения аккредитаций, выполняемых за один цикл.

Для начала генерируется число n как произведение двух больших чисел. Затем выбирается K различных чисел $V_1, V_2 \ldots, V_K$, где каждое V_i является квадратичным вычетом по модулю n, для того, чтобы сгенерировать открытый и секретный ключи. V выбирают таким, что сравнение

$$x^2 \equiv V_i \bmod n$$

имеет решение и существует $V_i^{-1} \bmod n$. Открытым ключом является полученная строка $V_1, V_2, \ldots, V_K$.

После этого вычисляют наименьшие значения S такие, что

$$S_i = sqrt\,(V_i^{-1}) \bmod n.$$

Строка $S_1, S_2, \ldots, S_K$ является секретным ключом.

Далее приступают к выполнению протокола идентификации:

1. Пользователь A выбирает случайное число r, $r < n$, вычисляет

$$x = r^2 \bmod n$$

и посылает x пользователю B.

2. Пользователь B отправляет пользователю A случайную двоичную строку из K бит: $b_1, b_2, \ldots, b_K$.

3. Пользователь A вычисляет

$$y = r \cdot (S_1^{b_1} \cdot S_2^{b_2} \cdot \ldots \cdot S_k^{b_K}) \bmod n.$$

Перемножаются только те значения S_i, для которых $b_i = 1$, если $b_i = 0$, то сомножитель S_i не входит в произведение. Вычисленное значение y отправляется пользователю B.

4. Пользователь B проверяет, что

$$x = y^2 \cdot (V_1^{b_1} \cdot V_2^{b_2} \cdot \ldots \cdot V_K^{b_K}) \bmod n.$$

Пользователь B перемножает только те значения V_i, для которых $b_i = 1$. Оба пользователя повторяют этот протокол t раз, пока B не убедится, что A знает $S_1, S_2, \ldots, S_K$. Вероятность того, что пользователь A может обмануть B, равна $(1/2)^{K_t}$. Рекомендуется в качестве контрольного значения брать вероятность обмана пользователя B равной $(1/2)^{20}$ при $K = 5$ и $t = 4$.

Рассмотрим пример из [17]. Выберем небольшие числовые значения. Если n = 35 (n – произведение 5·7), то возможные квадратичные вычеты будут следующими:

1: $x^2 \equiv 1 \ (mod\ 35)$ имеет решения: x = 1, 6, 29, 34;

4: $x^2 \equiv 4 \ (mod\ 35)$ имеет решения: x = 2, 12, 23, 33;

9: $x^2 \equiv 9 \ (mod\ 35)$ имеет решения: x = 3, 17, 18, 32;

11: $x^2 \equiv 11 \ (mod\ 35)$ имеет решения: x = 9, 16, 19, 26;

14: $x^2 \equiv 14 \ (mod\ 35)$ имеет решения: x = 7, 28;

15: $x^2 \equiv 15 \ (mod\ 35)$ имеет решения: x = 15, 20;

16: $x^2 \equiv 16 \ (mod\ 35)$ имеет решения: x = 4, 11, 24, 31;

21: $x^2 \equiv 21 \ (mod\ 35)$ имеет решения: x = 14, 21;

25: $x^2 \equiv 25 \ (mod\ 35)$ имеет решения: x = 5, 30;

29: $x^2 \equiv 29 \ (mod\ 35)$ имеет решения: x = 8, 13, 22, 27;

30: $x^2 \equiv 30 \ (mod\ 35)$ имеет решения: x = 10, 25.

Взаимно простыми с 35 не являются 14, 15, 21,25 и 30, поэтому они не имеют обратных значений по модулю 35. Число квадратичных вычетов по модулю 35, взаимно простых с $n = p \cdot q$ = 5·7 = 35 (для которых НОД (x, 35) = 1), равно

$$(p - 1)\ (q - 1)/4 = (5 - 1)\ (7 - 1)/4 = 6.$$

В табл. 4.3 представлены квадратичные вычеты по модулю 35, а также обратные к ним значения.

Пользователь A получает открытый ключ K, состоящий из четырех значений V: {4, 11, 16, 29}, а также секретный ключ K_c, состоящий из следующих значений S: {3, 4, 9, 8}.

Таблица 4.3

V	V^{-1}	S =*sqrt* (V^{-1})
1	1	1
4	9	3
9	4	2
11	16	4
16	11	9
29	29	8

Рассмотрим один цикл протокола.

1. Пользователь A выбирает некоторое случайное число $r = 16$ и вычисляет

$$x = 16^2 \ mod\ 35 = 11,$$

затем посылает это значение x пользователю B.

2. Пользователь B отправляет пользователю A случайную двоичную строку из четырех бит $b_i\{1, 1, 0, 1\}$.

3. Пользователь A вычисляет значение

$y = r \cdot (S_1^{b_1} \cdot S_2^{b_2} \cdot \ldots \cdot S_K^{b_K})\ mod\ n = 16 \cdot (3^1 \cdot 4^1 \cdot 9^0 \cdot 8^1)\ mod\ 35 = 31$ и отправляет полученное значение пользователю B.

4. Пользователь B проверяет значение x

$x = y^2 \cdot (V_1^{b_1} \cdot V_2^{b_2} \cdot \ldots \cdot V_K^{b_k})\ mod\ n = 31^2 \cdot (4^1 \cdot 11^1 \cdot 16^0 \cdot 29^1) mod\ 35 = 11.$

Пользователи A и B повторяют этот протокол t раз, меняя случайное число r, пока пользователь B не будет удовлетворён.

При малых значениях величин не достигается настоящей безопасности. Однако если n представляет собой число длиной более 512 бит, то пользователь B не сможет узнать секретного ключа пользователя A.

Рассмотренный протокол позволяет включить идентификационную информацию, формируемую в центре выдачи интеллектуальных карт, по заявке пользователя A. Эта информация l есть двоичная строка, представляющая идентификационную информацию о владельце карты (имя, адрес, персональный идентификационный номер, физическое описание) и о карте (дата окончания действия и т.п.).

Используя одностороннюю функцию $f(\cdot)$ вычисляют значения

$$V_j = f(l, j),$$

где j – некоторое двоичное число, сцепляемое со строкой l.

При этом для небольших значений j, отбирают k разных значений j, для которых V_j являются квадратичными вычетами по модулю n. Для отобранных квадратичных вычетов V_j вычисляют

наименьшие квадратные корни из V_j^{-1} (*mod n*). Совокупность из k значений V_j образует открытый ключ, а совокупность из k значений S_j – секретный ключ пользователя A.

ГЛАВА 5. ХЭШ-ФУНКЦИЯ

5.1. Основные понятия

Хэш-функцией [22,25,27] называется функция, отображающая аргумент произвольной конечной длины в образ фиксированной длины. Вычислимой в одну сторону хэш-функцией называется функция, для которой по данному аргументу вычислить её значение легко, а по данному значению функции аргумент найти сложно. Если хэш-функция зависит от секретного ключа, то она называется ключевой; в противном случае хэш-функция называется бесключевой.

Хэш-функции, используемые в протоколах аутентификации должны обладать стойкостью против попыток найти коллизии. Коллизией называется пара аргументов M и M', которым соответствует одно и то же значение хэш-функции $h(M) = h(M')$.

В настоящее время существует три способа построения хэш-функций:

- на основе трудно решаемой математической задачи;
- на основе алгоритмов блочного шифрования;
- разработанные с нуля.

Каждый из перечисленных выше методов обладает своими достоинствами и недостатками, однако наиболее распространенными на сегодняшний день оказались два последних метода. Это связано с тем, что при построении хэш-функций с нуля появляется возможность учета эффективности программной реализации. В свою очередь, широкое использование хэш-функций, построенных на основе алгоритмов блочного шифрования, является результатом тщательной проработки вопроса стойкости многих из существующих блочных алгоритмов.

5.2. Однонаправленные хэш-функции

Хэш-функция служит для сжатия подписываемого документа M до нескольких десятков или сотен бит. Хэш-функция принимает в качестве аргумента документ M произвольной длины и возвращает хэш-значение $m=h(M)$ фиксированной длины. Вообще, хешированная информация является сжатым двоичным представлением основного сообщения произвольной длины. Значение хэш-функции $h(M)$ сложным образом зависит от документа и не позволяет восстановить сам документ M.

На основе однонаправленной функции $f(\cdot)$ строится большинство хэш-функций. Функция $f(\cdot)$ образует выходное значение с фиксированной длиной n при задании двух входных значений длиной n. Этими входами являются блок исходного текста M_i и хэш-значение H_{i-1} предыдущего блока текста (рис. 5.1)

$$H_i = f(M_i, H_{i-1}).$$

Хэш-значение, вычисленное при вводе последнего блока текста, становится хэш-значением *m* сообщения *M*.

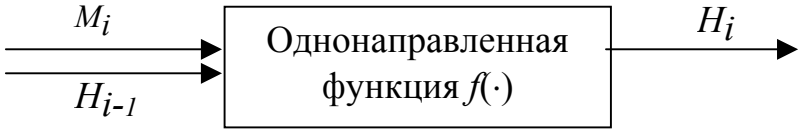

Рис. 5.1. Схема формирования однонаправленной хэш-функции

Независимо от длины входного текста однонаправленная хэш-функция всегда формирует выход фиксированной длины n.

Основным требованием к таким хэш-функциям является равномерность распределения их значений при случайном выборе значений аргументов.

В криптографии хэш-функции применяются для решения следующих задач:

- построение систем контроля целостности данных при их передаче или хранении,
- аутентификация источника данных.

При решении первой задачи для каждого набора данных вычисляется значение хэш-функции (называемое кодом аутентификации сообщений или имитовставкой), которое передаётся или хранится вместе с самими данными. При получении данных пользователь вычисляет значение свертки и сравнивает его с имеющимся контрольным значением. Несовпадение говорит о том, что данные были изменены.

Кроме того, хэш-функция может использоваться для обнаружения модификации сообщения, то есть служить в качестве криптографической контрольной суммы (также называемой кодом обнаружения изменений или кодом аутентификации сообщения). В этом качестве функция хэширования применяется при формировании и проверке электронной цифровой подписи.

В ряде технологий информационной безопасности используется своеобразный прием шифрования - с помощью односторонней хэш-функции. Своеобразие этого шифрования заключается в том, что оно, по существу, является односторонним, то есть не сопровождается обратной процедурой - расшифрованием на приемной стороне. Это имеет место только в том случае, когда отправитель и получатель используют одну и ту же процедуру одностороннего шифрования на основе хэш-функции.

Для того чтобы функция хэширования могла быть надлежащим образом использована в процессе аутентификации, она должна обладать:

- возможностью применения к аргументу любого размера;
- строго фиксированным размером;

- простотой и удобством вычисления для любых x;

- чувствительностью к всевозможным изменениям в тексте M таким, как вставки, выбросы, перестановки и т.п;

- свойством необратимости, иначе говоря, задача подбора документа M', который обладал бы требуемым значением хэш-функции, должна быть вычислительно неразрешима;

- вероятностью того, что значения хэш-функции двух различных документов (вне зависимости от их длин) совпадут, должна быть ничтожно мала, то есть гарантируется, что не может быть найдено другое сообщение, дающее ту же свертку. Это предотвращает подделку и также позволяет использовать значение хэш-функции в качестве криптографической контрольной суммы для аутентификации пользователей и проверки целостности сообщения.

Учитывая изложенные выше требования, рассмотрим примеры построения однонаправленных хэш-функций.

5.3. Хэш-функции на основе симметричных блочных алгоритмов

Наиболее очевидный подход состоит в том, чтобы шифровать сообщение M посредством блочного алгоритма в режиме *CBC* или *CFB* с помощью фиксированного ключа и некоторого вектора инициализации. Последний блок шифротекста можно рассматривать в качестве хэш-значения сообщения M.

Безопасный вариант хэш-функции можно получить, используя блок сообщения в качестве ключа, т.е. предыдущее хэш-значение - в качестве входа, а текущее - в качестве выхода. При этом сообщение M разбивается на блоки M_i, определяемые длиной ключа и обрабатываются поочередно. Поскольку блочные алгоритмы в большинстве своем являются 64-битовыми, некоторые схемы

хэширования проектируют так, чтобы хэш-значение имело длину, равную двойной длине блока.

Безопасность схемы хэширования базируется на безопасности лежащего в ее основе блочного алгоритма.

На рис. 5.2 в качестве примера приведены четыре хэш-функции безопасного хэширования и схемы их реализации.

1. $H_i = E_{H_{i-1}}(M_i) \oplus M_i$ 2. $H_i = E_{H_{i-1}}(M_i \oplus H_{i-1}) \oplus M_i \oplus H_{i-1}$

3. $H_i = E_{H_{i-1}}(M_i) \oplus H_{i-1} \oplus M_i$ 4. $H_i = E_{H_{i-1}}(M_i \oplus H_{i-1}) \oplus M_i$

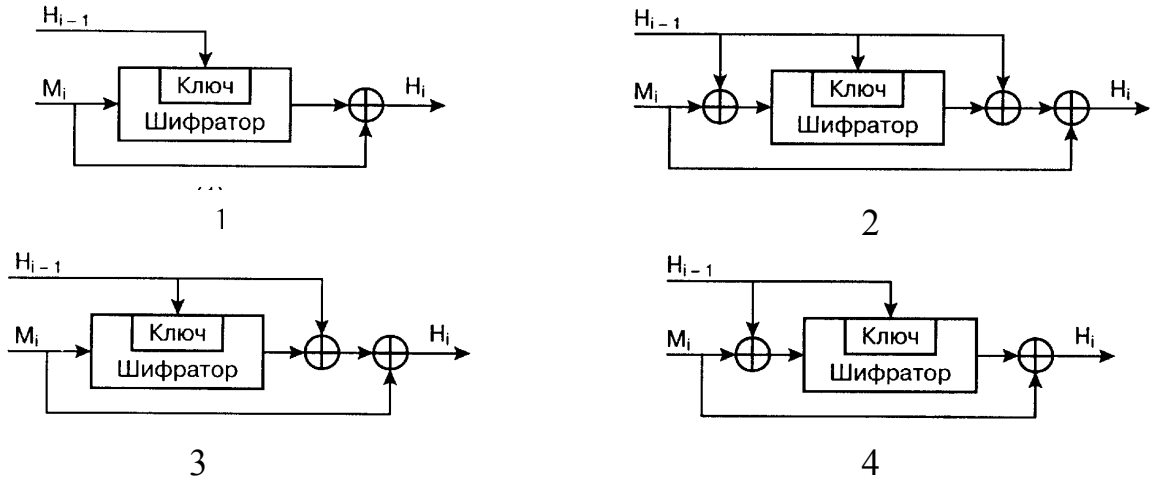

Рис. 5.2. Схемы безопасного хэширования

В настоящее время в системах информационной безопасности наиболее популярны хэш-функции *MD*4 (Message Digest 4), *MD*5 (Message Digest 5) и *SHA* (Secure Hash Algorithm). Хэш-функции *MD*4 и *MD*5 генерируют дайджесты фиксированной длины 128 бит, а длина свертки в *SHA* составляет 160 бит.

5.3.1. *Хэш-функция MD4*

При разработке хэш-функции *MD*4 Р. Райвест [17, 22] преследовал следующие цели:

- безопасность; малая вероятность коллизии; нахождение коллизии (то есть подбор двух различных сообщений с одинаковым хэш-

значением) должно быть вычислительно невозможным; безопасность не должна базироваться на каком-либо допущении, например на трудности факторизации больших чисел;

- быстродействие; алгоритм *MD*4 должен подходить для высокоскоростных программных реализаций и состоять из набора простых битовых операций с 32-битовыми операндами;
- простота и компактность; алгоритм *MD*4 должен быть предельно прост и не должен содержать сложных структур данных и программных модулей;
- унифицированность архитектуры; алгоритм *MD*4 должен легко адаптироваться для различной микропроцессорной архитектуры.

5.3.2. *Хэш-функция MD5*

Хэш-функция *MD*5 [25] является модернизированной хэш-функцией *MD*4. Результатом хэш-функции *MD*5 также является 128-битовое хэш-значение.

Алгоритм *MD*5 обрабатывает входной текст 512-битовыми блоками, разбитыми на шестнадцать 32-битовых подблоков. Выходом алгоритма является набор из четырех 32-битовых блоков, которые объединяются в единое 128-битовое хэш-значение.

Прежде чем перейти к основным этапам формирования хэш-функций *MD*5, необходимо:

- дополнить сообщение так, чтобы его длина была на 64 бит короче числа, кратного 512. Этим дополнением является 1, за которой вплоть до конца сообщения следует необходимое количество нулей. Затем к результату добавляется 64-битовое представление длины исходного сообщения (до дополнения);
- инициализировать четыре переменные:

$$A = 0x01234567;$$

$$B = 0x89abcdef;$$

$$C = 0xfedcba98;$$
$$D = 0x76543210.$$

- копировать во вспомогательные переменные a, b, c, и d

$$A \to a,\ B \to b,\ C \to c \text{ и } D \to d.$$

Основной цикл хэш-функции *MD*5 состоит из четырех практически одинаковых этапов (рис. 5.3). На каждом из них 16 раз используются различные операции. Каждая операция представляет собой нелинейную функцию над тремя из четырех переменных a, b, c и d. Затем результат суммируется с четвертой переменной, подблоком текста и константой. Далее этот результат циклически сдвигается вправо на переменное число битов и суммируется с одной из переменных a, b, c и d. Затем окончательный результат заменяет одну из переменных a, b, c и d (см. рис.5.3 и 5.4).

На каждом из четырех этапов используется своя нелинейная функция:

$$F(X, Y, Z) = (X \wedge Y) \vee (\overline{X} \wedge Z),$$
$$G(X, Y, Z) = (X \wedge Y) \vee (\overline{Y} \vee \overline{Z}),$$
$$H(X, Y, Z) = X \oplus Y \oplus Z,$$
$$I(X, Y, Z) = Y \oplus (X \vee \overline{Z}),$$

где $\oplus$ *XOR*; $\wedge$ *AND*; $\vee$ *OR*; $\overline{\ }$ *NOT*.

Эти функции спроектированы так, что если соответствующие биты X,Y и Z независимы, то каждый бит результата также будет независим. Функция F – это побитовое условие: если X то Y, иначе Z. Функция H -побитовая операция проверки чётности.

$FF(a,b,c,d,M_j,s,t_j)$ означает $a=b+((a+F(b,c,d)+M_j+t_j<<<s$,

$GG(a,b,c,d,M_j,s,t_j)$ означает $a=b+((a+G(\mathrm{b},c,d)+M_j+t_j<<<s$,

$HH(a,b,\mathrm{c},d,M_j,s,t_j)$ означает $a=b+((a+H(b,c,d)+M_j+t_j<<<s$,

II $HH(a,b,c,d,M_j,s,t_j)$ означает $\mathrm{a}=b+((a+I(b,c,d)+M_j+t_j<<<s$,

где M_j – j-й подблок сообщения, $<<<s$ – циклический сдвиг влево на s бит.

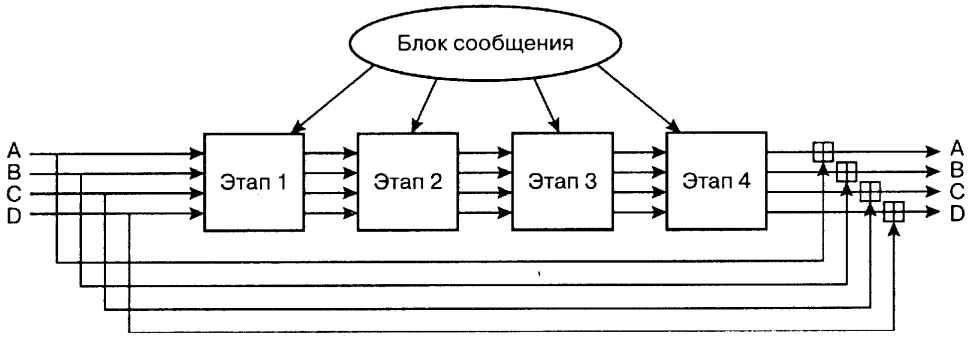

Рис. 5.3. Основной цикл

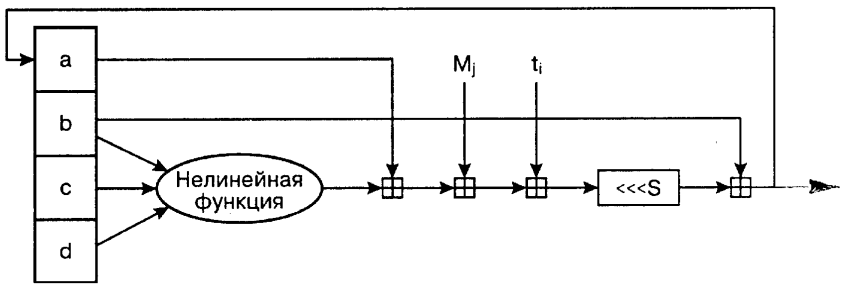

Рис. 5.4. Одна операция *MD*5

По сравнению с *MD*4 алгоритм *MD*5 улучшен следующим образом:

- добавлен четвертый этап;
- в каждом действии теперь используется уникальная аддитивная константа;
- для увеличения асимметрии функция *G* на втором этапе была изменена на $(X \wedge Y) \vee (\overline{Y} \vee \overline{Z})$;
- для получения более быстрого лавинного эффекта результат каждой операции теперь суммируется с результатом предыдущего этапа;
- для увеличения несходства шаблонов изменен порядок, в котором использовались подблоки сообщения на этапах 2 и 3;
- для увеличения лавинного эффекта были приближенно оптимизированы значения циклического сдвига влево на каждом

этапе. Четыре сдвига, используемые на каждом этапе, отличаются от значений, используемых на других этапах.

5.3.3. *Хэш-функция SHA*

Алгоритм хэширования *SHA* описан в стандарте США безопасного хэширования *SHS* (Secure Hash Standard) [25].

Для любого входного сообщения длиной меньше 2^{64} бит алгоритм *SHA* выдает 160-битовый результат - хэш-значение (дайджест) сообщения. Алгоритм безопасного хэширования *SHA* используется в качестве входа алгоритма *DSA*, который вычисляет цифровую подпись сообщения *M*. Формирование цифровой подписи на основе дайджеста сообщения, а не самого сообщения повышает эффективность процесса подписания, поскольку дайджест обычно намного короче самого сообщения. Такой же дайджест сообщения должен вычисляться пользователем, проверяющим полученную подпись; при этом в качестве входа в алгоритм *SHA* используется полученное сообщение *M*.

Алгоритм хэширования *SHA* назван безопасным, потому что спроектирован таким образом, чтобы было вычислительно невозможно восстановить сообщение, соответствующее данному дайджесту, а также найти два различных сообщения, которые дадут одинаковый дайджест. Любое изменение сообщения при передаче с очень большой вероятностью приведет к формированию отличающегося дайджеста, и принятая цифровая подпись не пройдет проверку.

Принципы, лежащие в основе построения алгоритма *SHA*, аналогичны принципам, использованным при разработке алгоритма *MD*4. Алгоритм хэширования *SHA* выдает более длинное хэш-значение (160 бит), чем у *MD*5 (128 бит).

Рассмотрим более подробно алгоритм хэширования *SHA*.

Прежде всего, дополняется исходное сообщение, так чтобы его длина стала кратной 512 бит. В сущности, используется та же процедура дополнения сообщения, как и в *MD*5: сначала добавляется единица, затем столько нулей, сколько необходимо для получения сообщения, длина которого на 64 бит меньше числа, кратного 512 бит, и затем добавляется 64-битовое представление длины исходного сообщения.

Далее инициализируются пять 32-битовых переменных (в *MD*5 используется четыре переменных):

$$A = 0x67452301;$$
$$B = 0xefcdab89;$$
$$C = 0x98badcfe;$$
$$D = 0x10325476;$$
$$E = 0xc3d2elf0.$$

Затем начинается основной цикл алгоритма. Он обрабатывает блоки сообщения по 512 бит и осуществляет этот процесс поочередно для всех 512-битовых блоков сообщения. Сначала пять переменных A, B, C, D, E копируются во вспомогательные переменные a, b, c, d, e:

$$A \to a, B \to b, C \to c, D \to d \text{ и } E \to e.$$

Основной цикл состоит из четырех этапов по 20 операций, каждая операция из которых реализует нелинейную функцию над тремя из пяти переменными a, b, c, d, e, а затем производит сдвиг и сложение.

В алгоритме *SHA* используется следующий набор нелинейных функций:

$$f_i(X, Y, Z) = (X \wedge Y) \vee (\overline{X} \wedge Z) \quad \text{для } i = 0 \div 19,$$
$$f_i(X, Y, Z) = X \oplus Y \oplus Z \quad \text{для } i = 20 \div 39,$$
$$f_t(X, Y, Z) = (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z) \quad \text{для } i = 40 \div 59,$$
$$f_i(X, Y, Z) = X \oplus Y \oplus Z \quad \text{для } i = 60 \div 79,$$

где i- номер операции ($i = 0 \div 79$).

В алгоритме используются следующие четыре шестнадцатеричные константы:

$K_i = 2^{1/2}/4 = 0x5a827999$ для $i = 0 \div 19$,

$K_i = 3^{1/2}/4 = 0x6ed9eba1$ для $i = 20 \div 39$,

$K_i = 5^{1/2}/4 = 0x8flbbcdc$ для $i = 40 \div 59$,

$K_i = 10^{1/2}/4 = 0xca62c1d6$ для $i = 60 \div 79$.

Блок сообщения преобразуется из шестнадцати 32-битовых слов ($M_0 \div M_{15}$) в восемьдесят 32-битовых слов ($W_0 \div W_{79}$) с помощью следующего алгоритма:

$$W_t = M_t \quad \text{для } t = 0 \div 19,$$

$$W_t = (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) <<< 1 \quad \text{для t} = 16 \div 79,$$

где t - номер операции (для $t = 1 \div 80$), W - t-й подблок расширенного сообщения, $<<< S$ - циклический сдвиг влево на S бит.

С учетом введенных обозначений главный цикл из восьмидесяти операций можно описать так:

FOR $t = 0$ *to* 79

$TEMP = (a <<< 5) + f_t(b, c, d) + e + W_t + K_t$

$e = d$

$d = c$

$c = (b <<< 30)$

$b = a$

$a = TEMP$

Схема выполнения одной операции *SHA* показана на рис. 5.5. Сдвиг переменных осуществляет ту же функцию, которую в *MD*5 выполняет использование в различных местах разных переменных. После этого значения a, b, c, d и e суммируются с A, B, C, D и E соответственно, и алгоритм переходит к обработке следующего блока данных. Окончательный выход формируется в виде конкатенации значений A, B, C, D и E.

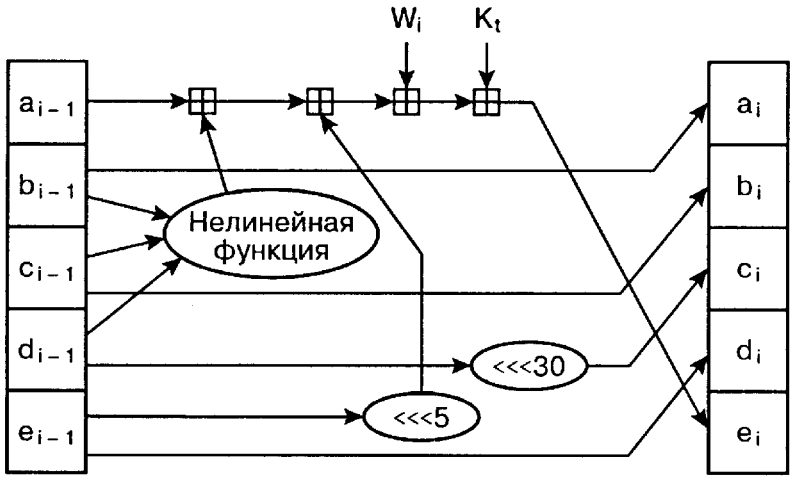

Рис. 5.5. Схема выполнения одной операции алгоритма *SHA*

Алгоритм *SHA* - модернизированный *MD*4, но выдает 160-битовое хэш-значение. Главными изменениями являются введение расширяющего преобразования и суммирование выходного значения предыдущего шага с последующим для получения более быстрого лавинного эффекта.

Сопоставим структурные и информационно-логические изменения, внесенные *SHA* относительно *MD*4 и *MD*5:

1. В *SHA* также добавился четвертый этап. Однако в *SHA* на четвертом этапе используется та же функция *f*, что и на втором этапе.

2. *SHA* придерживается схемы *MD*4, повторно используя константы.

3. В *SHA* используется версия функции из *MD*4:

$$(X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z).$$

4. Изменение, увеличивающее лавинный эффект, внесено и в *SHA*. Отличие состоит в том, что в *SHA* к переменным *a*, *b*, *c* и *d*, которые уже используются в f_t, добавлена пятая переменная.

5. В *MD*5 изменился порядок, в котором использовались подблоки сообщения на этапах 2 и 3, чтобы шаблоны стали менее похожими. Алгоритм *SHA* на данной стадии совершенно отличается,

так как использует циклический код исправления ошибок.

6. В *SHA* на каждом этапе использует постоянное значение сдвига. Это значение — число взаимно простое с размером слова, как и в *MD*4.

Данное сопоставление приводит к следующему заключению: алгоритм *SHA* – это модернизированный *MD*4 с добавлением расширяющего преобразования, дополнительного этапа и улучшенным лавинным эффектом. Алгоритм *MD*5 - это *MD*4 с улучшенным битовым хешированием, дополнительным этапом и улучшенным лавинным эффектом. Поскольку *SHA* выдает 160-битовое хэш-значение, он более устойчив к атакам полного перебора и атакам дня рождения, чем большинство других алгоритмов хэширования, формирующих 128-битовые хэш-значения.

5.3.4.*Алгоритм хэширования ГОСТ Р 34. 11-94*

Этот стандарт разработан для использования совместно со стандартом на цифровую подпись (ГОСТ Р 34.10-94) [28]. Основу данной хэш-функции составляет алгоритм блочного шифрования ГОСТ 28147-89, который оперирует с блоками длиной 64 бита и длиной ключа 256 бит, соответственно длина выходной последовательности хэш-функции составляет 256 бит.

Алгоритм вычисления хэш-функции работает с любой двоичной последовательностью, то есть не накладывает ограничения на длину входной последовательности. Описать работу алгоритма можно следующим образом: $H_i = h(M, H_{i-1})$,

где H_{i-1} - значение хэш-кода предыдущей итерации; M - входная последовательность, не подвергшаяся обработке.

Процедура вычисления функции h состоит из последовательности следующих шагов.

На первом шаге:

- инициализируются переменные L (текущее значение длины обработанной части входной последовательности) и I (значение контрольной суммы);

- если длина входной необработанной последовательности больше 256 ($|M| > 256$), алгоритм переходит к шагу 3, в противном случае переходит к шагу 2. На втором шаге определяются:

- $L = (L + |M|) \ mod \ 256$;

- $I = (I + (0^{256 - |M|} || M) \ mod \ 256$ $0^{256 - |M|}$ - последовательность битовых нулей длиной 256 - $|M|$, || - конкатенация битовых строк. Таким образом, на этом шаге вычисляется текущее значение контрольной суммы;

- $H = \chi ((0^{256 - |M|} || M), H)$ - вычисляется значение функции хэширования χ от аргументов, представляющих собой хэшируемый блок и начальный вектор хэширования (H);

- $H = \chi(L, H)$;

- $H = \chi(I, H)$ - результат данного вычисления и является окончательным результатом хэш-функции;

- конец работы алгоритма.

Из входной последовательности выбирается очередной блок длиной 256 бит ($M = M_0 \ || \ M_1$) и на третьем шаге производятся следующие действия:

- $H = \chi(M_0, H)$;

- $L = (L + 256) \ mod \ 256$;

- $I = (I + M_0) \ mod \ 256$;

- $M = M_0$;

- переход к шагу 2.

Вычисление функции x состоит из следующих этапов:

- генерация четырех секретных ключей длиной 256 бит;

- зашифрование четырех подблоков (составляющие начальный вектор хэширования H) по 64 бита каждый в соответствии с алгоритмом ГОСТ 28147-89 в режиме простой замены;

- применение перемешивающей функции к результату зашифрования.

Генерация секретных ключей

При генерации секретных ключей используются данные H и M (входная последовательность, представленная в двоичном коде) и инициализируются следующие константы: $C_2 = C_4 = 0^{256}$ и $C_3 = \text{I}^8\ 0^8 1^{16}\ 0^{24} 1^{16}\ 0^8\ (0^8 1^8)^2 1^8\ 0^8$ x x $(0^8 1^8)^4\ (1^8 0^8)^4$, где a^k - двоичная последовательность из к бинарных знаков a - {0,1}. После чего выполняется алгоритм:

$$1 = 1\ U = H\ V = M$$

$$W = U \oplus V$$

$$K_\text{i} = P(W)$$

$$FOR\ i = 2\ to\ 4$$

$$\{U = A(U) \oplus Q;$$

$$V = A(A(V))$$

$$W = U \oplus \text{V}$$

$$K_i = P(W)\}.$$

Функция P представляет собой перестановку $s(i + 1 + 4\ (k - 1)) = 8i + k$, $i = 0 \div 3$, $k = 1 \div 8$ над подблоками длиной 8 бит исходной двоичной последовательности длиной 256 бит. Таким образом, последовательность $x_{32} \| \ldots \| x_1$ преобразуется в $x_{s(32)} \| \ldots \| x_{s(1)}$. Функцией $A(X)$ обозначают следующее преобразование последовательности, разделенной на 4 подблока по 64 бита каждый: $A(X) = (x_t \oplus x_2) \| x_4 \| x_3 \| x_2$.

Шифрующее преобразование

Начальный вектор H разбивается на 4 подблока h_i длиной 64 бита, и каждый подблок зашифровывается на своем ключе K_i, то есть $s_j = Ek_i, (h_i)$ (i=1,2,3,4). В результате получается последовательность $S = s_1 \| s_2 \| s_3 \| s_4$.

Перемешивающая функция

Сначала исходная последовательность разбивается на шестнадцать 16-битных подблоков n_i; $i = 1 \div 16$ и при помощи регистра сдвига с обратной связью преобразуется в последовательность следующего вида: $n_1 \oplus n_2 \oplus n_3 \oplus n_4 \oplus n_{13} \oplus n_{16} \,|\, n_{15} \,|\, \ldots$

Если обозначить данную функцию через φ, то исходная функция хэширования - $\chi(M, H) = \varphi^{61}((H \oplus \varphi(M \oplus \varphi^{12}(S))))$, где степень функции φ обозначает, сколько раз она применяется к битовой последовательности.

Криптографическая стойкость данной хэш-функции основана на стойкости применяемого в ней алгоритма блочного шифрования, используемого в режиме простой замены.

5.4. Требования к хэш-функциям

При практическом использовании хэш-функции должны удовлетворять следующим требованиям:

- алгоритм должен обладать высокой скоростью обработки информации;
- хэш-функция должна быть стойкой против атаки методом «грубой силы»;
- программная реализация хэш-функции должна быть максимально адаптирована под использование на современной аппаратно-программной базе.

Этим требованиям должен удовлетворять как сам алгоритм выработки хэш-значения, так и хэширующая функция.

На практике повышение скорости выработки хэш-значения может быть достигнуто в основном за счет применения простого преобразования, которое переводит одно сообщение в другое

посредством элементарной операции, например, для увеличения скорости обработки сообщения необходимо, чтобы алгоритм выработки хэш-значения включал в себя также алгоритм вычисления хэш-значения одного сообщения из хэш-значения другого сообщения, которое получается из начального с помощью элементарного преобразования.

5.5. Стойкость хэш-функций

С точки зрения криптографической стойкости важным свойством хэш-функций является отсутствие коллизий, то есть невозможность найти такие значения $x \neq y$, чтобы $h(x) = h(y)$. В криптографических приложениях важным понятием является криптографически стойкая хэш-функция, для которой не существует эффективного алгоритма нахождения значений $x \neq y$, где выполнялось бы условие $h(x) = h(y)$ или не существует эффективного алгоритма нахождения коллизии при заданном x такого $y \neq x$, что $h(x) = h(y)$. Однако следует заметить, что данное требование носит формальный характер. Практически значимым является отсутствие у хэш-функции корреляции. Свободной от корреляции называется хэш-функция, у которой невозможно найти пары таких значений $x \neq y$, что вес Хэмминга двоичного вектора
$h(x)$ *xor* $h(y)$ будет меньше веса Хэмминга применительно к двоичному вектору $h(M)$. Свобода от корреляции с точки зрения криптографической стойкости является гораздо более сильным свойством хэш-функции, чем свобода от коллизий. Данный факт подтверждается тем, что из любой хэш-функции, являющейся свободной от коллизий и одновременно свободной от корреляций, можно построить другую хэш-функцию, которая тоже будет свободной от коллизий, но при этом может не сохранить свойство свободы от корреляции.

ГЛАВА 6. ЦИФРОВАЯ ПОДПИСЬ

Наиболее важной областью применения криптографии с открытым ключом являются цифровые подписи (ЦП) [1,16,29]. Они обеспечивают целый комплекс защиты, который было бы довольно сложно осуществить для обычного бумажного документооборота.

Однако при этом возникают проблемы аутентификации автора документа и самого документа (т.е. установления подлинности подписи и отсутствия изменений в полученном документе).

Проблема аутентификации является актуальной в вычислительных системах управления и вообще там, где необходимо удостовериться в подлинности полученного по каналам связи или на машинных носителях сообщения (документа).

Задачи аутентификации можно разделить на следующие типы: аутентификация абонента, аутентификация принадлежности абонента к данной группе, аутентификация хранящихся на машинных носителях документов.

Рассмотрим аутентификацию документов (или файлов) как более важную. Если рассматривать случай обмена секретными материалами (военная или дипломатическая связь), то с большой степенью уверенности можно предположить, что обмен осуществляют доверяющие и достойные доверия стороны.

Однако возможно, что обмен находится под наблюдением криптоаналитика, который способен выполнять сложные вычисления и затем либо создавать свои собственные документы, либо перехватывать и изменять документы законного отправителя. Это тот случай, когда защищаться надо только от противника, поскольку «свои» подвести не могут. В коммерческом мире верно почти обратное, т.е. отправитель и получатель хотя и «свои», но могут обманывать друг друга даже в большей степени, чем посторонние.

В ситуациях, когда нет полного доверия между отправителем и получателем, требуется нечто большее, чем простая аутентификация.

Наиболее привлекательным решением этой проблемы оказывается ЦП. Она является аналогом подписи, сделанной от руки и должна обеспечивать следующие возможности:

- возможность установить автора, а также дату и время подписи;
- возможность установить достоверность содержимого сообщения на время подписи:
- возможность проверки подписи третьей стороной на случай возникновения спора.

На основе этих возможностей могут быть сформулированы требования, выдвигаемые к цифровой подписи:

- подпись должна быть двоичным кодом, зависящим от подписываемого сообщения;
- подпись должна использовать некоторую информацию, уникальную для отправителя, чтобы предотвратить возможность, как фальсификации, так и отрицания авторства;
- цифровую подпись можно относительно просто произвести;
- цифровую подпись можно относительно просто распознать и проверить;
- с точки зрения вычислений нереально фальсифицировать цифровую подпись ни с помощью создания нового сообщения для имеющейся цифровой подписи, ни с помощью создания фальшивой цифровой подписи для имеющегося сообщения.

Непосредственная цифровая подпись подразумевает участие только обменивающихся данными сторон (отправитель, получатель). Предполагается, что получатель знает открытый ключ отправителя. Цифровая подпись может быть сформирована с помощью зашифрования всего сообщения личным ключом отправителя или с

помощью шифрования хэш-кода сообщения личным ключом отправителя.

При формировании ЦП отправитель первым делом вычисляет хэш-функцию $h(M)$ подписываемого текста M. Вычисленное значение хэш-функции $h(M)$ представляет собой короткий блок информации m, характеризующий весь текст M в целом.

Затем число m шифруется секретным ключом отправителя. Получаемая при этом пара чисел представляет собой электронную цифровую подпись для данного сообщения M.

При проверке ЦП получатель сообщения снова вычисляет хэш-функцию $m = h(M)$ принятого по каналу текста M, затем при помощи открытого ключа отправителя проверяет, соответствует ли полученная подпись вычисленному значению m хэш-функции.

Без знания секретного ключа подписывания отправителя нельзя подделать ЦП. В качестве подписываемого документа можно использовать любой файл.

Подписанный файл создается из неподписанного путем добавления в него одной или более электронных подписей, которые должны содержать следующую информацию:

- дату подписи;
- срок окончания действия ключа данной подписи;
- информацию о лице, подписавшим файл (Ф.И.О., должность, краткое наименование фирмы);
- идентификатор подписавшего (имя открытого ключа);
- собственно цифровую подпись.

6.1. Концепция формирования цифровой подписи

Абоненты посылают друг другу подписанные электронные документы. Для каждого абонента генерируется пара ключей: секретный K_c и открытый K_o. Причем секретный ключ хранится абонентом в тайне и используется им только для формирования

ЦП. Открытый ключ известен всем другим пользователям и предназначен для проверки цифровой подписи получателем подписанного электронного документа. Открытый ключ не позволяет вычислить секретный, а является необходимым инструментом, позволяющим проверить подлинность электронного документа и автора подписи.

Для генерации пары ключей (открытого и секретного) в алгоритмах цифровой подписи используются различные математические схемы, основанные на применении однонаправленных функций. По известным сложным вычислительным задачам эти схемы разделяются на две группы:

- задача факторизации (разложения на простые множители) больших чисел;
- задача дискретного логарифмирования.

В качестве примера рассмотрим процесс формирования ЦП с использованием алгоритмов *RSA*, ЭльГамаля, *DSA* , ГОСТ 34.10-94, на базе эллиптических кривых и цифровых подписей с функциональными дополнительными свойствами.

6.2. Алгоритм цифровой подписи *RSA*

Процедура постановки цифровой подписи состоит из следующих этапов [9,10]:

Действия отправителя:

1. Вычисление секретного и открытого ключей, для чего отправитель электронных документов выбирает два больших простых числа P и Q, находит их произведение $N = P \cdot Q$.

Затем вычисляет значение функции Эйлера $\varphi(N)$:

$$\varphi(N) = (P - 1)(Q - 1)$$

2. Вычисление открытого ключа K_o из условий:

$$K_o \le \varphi(N), \ \text{НОД}\ (K_o, \varphi(N)) = 1,$$

и секретного ключа K_c из условий:

$$K_c < N,\ K_o \cdot K_c \equiv 1(mod\ \varphi(N)).$$

3. Передача пользователем сети пары чисел (K_o, N), которая является открытым ключом, партнерам по переписке для проверки его цифровой подписи. Число K_c сохраняется отправителем как секретный ключ для подписывания.

Схема формирования и проверки цифровой подписи *RSA* показана на рис. 6.1.

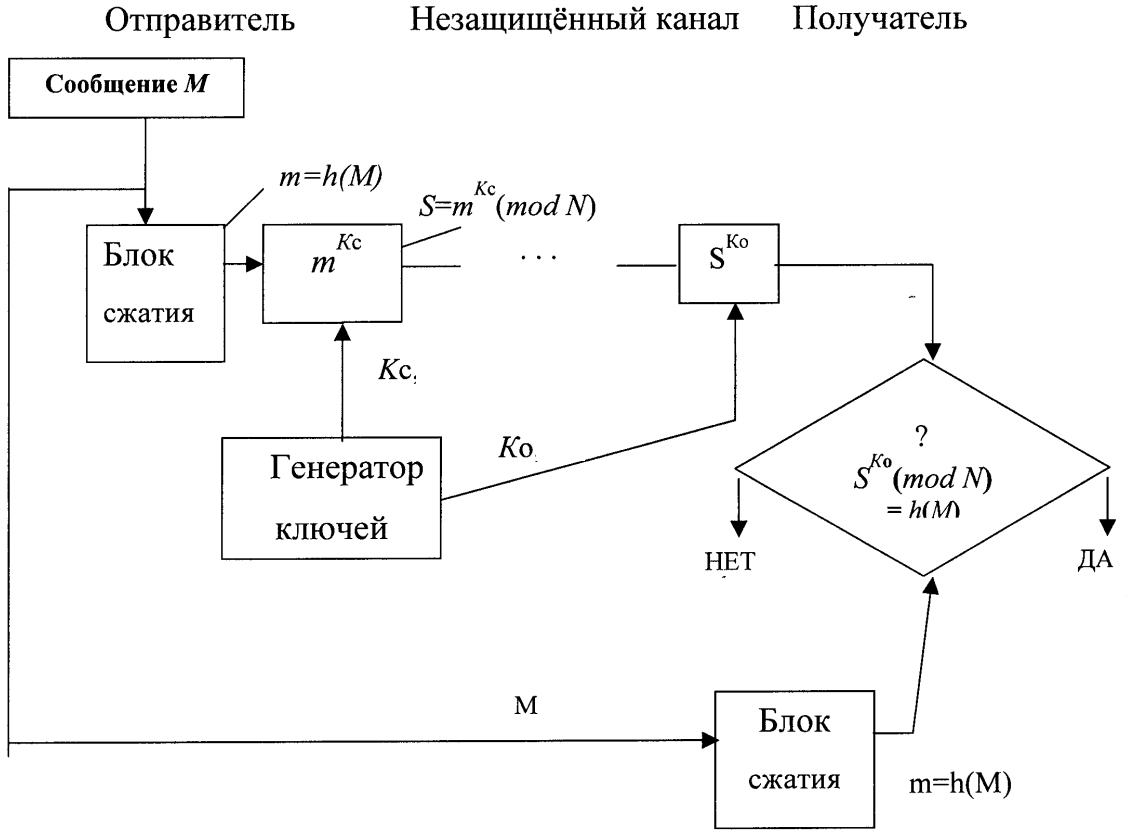

Рис. 6.1. Обобщенная схема цифровой подписи *RSA*

Для формирования ЦП по схеме *RSA* необходимо:

1. Сообщение M (блок информации, файл и т.д.) сжать с помощью хэш-функции $h(M)$ в целое число m.

2. Вычислить цифровую подпись S под электронным документом M, используя хэш-значение m и секретный ключ K_c:

$$S = m^{K_c}\ (mod\ N).$$

3. Передать получателю электронный документ M, подписанный цифровой подписью S.

Действия получателя:

1. После приёма пары (M, S) получатель вычисляет хэш-значение сообщения M как $m = h(M)$ и восстанавливает хэш-значение m' цифровой подписи S по формуле

$$m' = S^{K_o} \ (mod\ N).$$

2. Если $S^{K_o}(mod\ N) = h(M)$, то получатель признаёт пару (M, S) подлинной.

Итак, процедура проверки электронной цифровой подписи состоит из двух этапов: вычисления хэш-функции документа и собственно математических вычислений, предусмотренных в данном алгоритме подписи. Последние заключаются в проверке того или иного соотношения, связывающего хэш-функцию документа, подпись под ним и секретный ключ K_c подписавшего абонента. Если рассматриваемое соотношение оказывается выполненным, то подпись признается правильной, а сам документ – подлинным, в противном случае документ считается измененным, а подпись под ним – поддельной.

Доказано [9], что только обладатель секретного ключа K_c может сформировать электронную цифровую подпись S по документу M, а определить секретное число K_c по открытому ключу K_o не легче, чем разложить число N на простые множители.

Результат проверки цифровой подписи S будет положительным тогда и только тогда, когда при вычислении S был применен секретный ключ K_c, соответствующий открытому ключу K_o. В связи с этим открытый ключ K_o иногда называется «идентификатором» подписавшего.

Рассмотрим числовой пример использования хэш-функции имеющей следующий вид:

$$H_i = [(H_{i-1} \oplus M_i)^2] \ (mod\ N), \qquad (6.1)$$

где H_o –вектор инициализации; $M_i = M_1, M_2, \ldots, M_n$.

Пример 3 [14,17]. Хешируемое сообщение «531». Выбираем числа $P = 7$, $Q = 3$, $H_o = 6$. Определяем $N = P \cdot Q = 7 \cdot 3 = 21$. Вычисляем хэш-код сообщения 531 поблочно по формуле (6.1).

1. $M_1 + H_o = 5 + 6 = 11$;

$[M_1 + H_o]^2 \ (mod\ N) = 11^2 \ (mod\ 21) = 16 = H_1$;

2. $M_2 + H_1 = 3 + 16 = 19$;

$[M_2 + H_1]^2 \ (mod\ N) = 19^2 (mod\ 21) = 4 = H_2$;

3. $M_3 + H_2 = 1 + 4 = 5$;

$[M_3 + H_2]^2 \ (mod\ N) = 5^2 \ (mod\ 21) = 4 = H_3$.

В итоге получаем хэш-значение сообщения «531», равное 4.

Пример 4 [14]. Получить хэш-код для сообщения «HESHING» при помощи хэш-функции, вычисляемой по формуле (6.1). Выбираем числа $P = 17$, $Q = 19$.

Порядок вычисления хэш-кода:

1) вычисляем значение модуля $N = P \cdot Q = 323$;

2) представляем сообщение «HASHING» в виде символов *ASC* II:

3)

H	A	S	H	I	N	G
72	65	83	72	73	78	71

4) представляем коды ASC II битовой строкой:

72	65	83	72	77	78	71
01001000	01000001	01010011	01001000	01001001	01001110	01000111

5) разбиваем байт пополам, затем добавляем в начало полубайта единицы и получаем хэшируемые блоки M_i :

M_1	M_2	M_3	M_4	M_5	M_6	M_7
11110100	11111000	11110100	11110001	11110101	11110011	11110100
M_8	M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}
11111000	11110100	11111001	11110100	11111110	11110100	11110111

Вычисляем хэш-код сообщения поблочно по формуле

$$H_i = [(H_{i-1} \oplus M_i)^2] \ (mod\ N),$$

1. Вычисляем H_1

M_1 $= 11110100$

$\oplus$

H_o $= \underline{00000000}$

$H_o \oplus M_1$ $= 11110100_2 = 244_{10}$

$[(H_o \oplus M_1)]^2 \ (mod\ 323)$ $= 244^2 \ (mod\ 323) = 104$

H_1 $= 104_{10} = 01101000_2$

2. Вычисляем H_2

M_2 $= 11111000$

$\oplus$

H_1 $= \underline{01101000}$

$H_1 \oplus M_2$ $= 10010000_2 = 144_{10}$

$[(H_1 \oplus M_2)]^2 \ (mod\ 323)]$ $= 144^2 \ (mod\ 323) = 64$

H_2 $= 64_{10} = 01000000_2$

. . . . . .

Вычисляем H_i. . . . . и так далее.

Замечание. Здесь и далее символ 64_{10} означает число 64 в 10-тичной системе исчисления, а 01000000_2 – число 64 в 2-ной системе счисления.

Для алгоритма цифровой подписи *RSA* присущи следующие недостатки:

- при вычислении модуля N и ключей K_o, K_c для системы цифровой подписи *RSA* необходимо проверить большое количество дополнительных условий, что сделать достаточно трудно. Невыполнение любого из этих условий делает возможным фальсификацию цифровой подписи со стороны того, кто обнаружит такое невыполнение. При подписании важных документов нельзя допустить такую возможность даже теоретически;

- для обеспечения криптостойкости цифровой подписи *RSA* по отношению к попыткам фальсификации необходимо использовать при вычислениях N, K_o, K_c целые числа не менее 2^{512} каждое, что требует больших вычислительных затрат, превышающих на 20÷30% вычислительные затраты других алгоритмов электронной цифровой подписи при сохранении того же уровня криптостойкости;
- цифровая подпись *RSA* чувствительна к атакам, то есть алгоритм позволяет злоумышленнику без знания K_c сформировать подписи под теми документами, у которых результат хэширования можно вычислить как произведение результатов хэширования уже подписанных документов.

Предположим, что есть два сообщения M_1 и M_2. Первое сообщение достоверно, а для второго криптоаналитик пытается получить то же самое значение хэш-кода, выдав сообщение M_1 за сообщение M_2 (т.е. пытается получить коллизию). Для этого криптоалалитик подготавливает примерно $\sqrt{N}$ различных, незначительно отличающихся версий M_1 и M_2 и для каждой из них вычисляет хэш-код. С высокой вероятностью удаётся обнаружить пару версий M'_1 и M'_2, имеющих один и тот же хэш-код. В дальнейшем при использовании данного хэш-кода в схеме ЦП можно выдать сообщение M'_2 вместо сообщения M'_1, содержание которого близко к содержанию сообщения M_1.

К основным методам предотвращения данной атаки можно отнести увеличение длины получаемых хэш-кодов.

6.3. Алгоритм цифровой подписи Эль Гамаля (*EGSA*)

Идея *EGSA* [1,13] основана на том, что для обоснования практической невозможности фальсификации ЦП может быть применена более сложная вычислительная задача, чем разложение на множители большого целого числа, а именно задача дискретного логарифмирования. Кроме того, данный алгоритм позволяет избежать явной слабости алгоритма цифровой подписи *RSA*, связанной с возможностью подделки цифровой подписи под некоторыми документами без определения секретного ключа.

Рассмотрим этапы процесса формирования электронной цифровой подписи по схеме Эль Гамаля.

Действия отправителя:

1. Выбирает некоторое большое простое число P и большое число G, причем $G < P$, для того, чтобы генерировать открытый ключ Y и секретный ключ X. Отправитель и получатель подписанного сообщения используют при вычислениях большие одинаковые целые числа P ($\sim 2^{1024}$) и G ($\sim 2^{512}$), которые не являются секретными.

2. Выбирает случайное целое число X, $1 < X \le (P - 1)$, и вычисляет $Y = G^X \bmod P$.

3. Передаёт число Y, являющееся открытым ключом, используемым для проверки подписи отправителя, всем потенциальным пользователям сети (получателям документов), при этом число X, которое является секретным ключом отправителя для подписания документов должно храниться в секрете.

4. Хэширует сообщение M с помощью хэш-функции $h(\cdot)$ в целое число m для того, чтобы подписать сообщение M

$$m = h(M), \quad 1 < m < (P - 1)$$

и генерирует случайное целое число K_c, $1 < K_c < (P - 1)$, где K_c- второй секретный ключ отправителя. Числа K_c и $(P - 1)$ являются взаимно простыми.

5. Отправитель вычисляет целое число a:

$$a = G^{K_c} \bmod P$$

и, воспользовавшись расширенным алгоритмом Евклида, вычисляет с помощью секретного ключа X целое число b из уравнения

$$m = X \cdot a + K_c \cdot b \ (mod(P - 1)).$$

6. Под документом M проставляется пара чисел (a, b), образующих цифровую подпись $S = (a, b)$.

7. Получателю передаётся тройка чисел (M, a, b), в то время как пара чисел (X, K_c) держится в секрете.

Действия получателя:

1. После приёма подписанного документа (M, a, b) получатель должен проверить, соответствует ли подпись $S = (a, b)$ документу M. С этой целью получатель хэширует принятое сообщение M:

$$m' = h(M).$$

2. Вычисляет значение

$$A = Y^a a^b \ (mod\ P)$$

и признает сообщение M подлинным, только тогда, когда

$$A = G^{m'} \ (mod\ P).$$

В конечном итоге получатель проверяет справедливость соотношения

$$Y^a a^b \ (mod\ P) = G^{m'} \ (mod\ P).$$

Последнее равенство будет выполняться тогда и только тогда, когда подпись $S = (a, b)$ под документом M получена с помощью именно того секретного ключа X, из которого был получен открытый ключ Y. Так можно надёжно удостовериться в том, что отправителем сообщения M был обладатель именно данного

секретного ключа X, не раскрывая при этом сам ключ, и что отправитель подписал именно этот конкретный документ M.

По методу Эль Гамаля выполнение каждой подписи требует нового значения K_c, выбираемого случайным образом. Если нарушитель раскроет когда-либо значение K_c, повторно используемое отправителем, то он сможет раскрыть секретный ключ X отправителя.

Пример 5 [14,17]. Сформировать цифровую подпись с использованием алгоритма Эль Гамаля. Исходные данные: $P = 11$, $G = 2$ и секретный ключ $X = 8$.

Определим значение открытого ключа:

$$Y = G^X \bmod P = Y = 2^8 \bmod 11 = 3.$$

Предположим, исходное сообщение M характеризуется хэш–значением $m = 5$.

Выберем случайное целое число $K_c = 9$ для того, чтобы вычислить цифровую подпись для сообщения M, имеющего хэш-значение $m = 5$. Убедимся, что числа K_c и $(P - 1)$ являются взаимно простыми.

$$\text{НОД}\ (9, 10) = 1.$$

Вычислим элементы a и b подписи:

$$a = G^{K_c} \bmod P = 2^9 \bmod (11) = 6.$$

Элемент b определяем, используя расширенный алгоритм Евклида:

$$m = X \cdot a + K_c \cdot b \ (mod\ (P - 1)).$$

При $m = 5$, $a = 6$, $X = 8$, $K_c = 9$, $P = 11$ получаем

$$5 = (6 \cdot 8 + 9 \cdot b)\ (mod\ (10)) \text{ или } 9 \cdot b = -43\ (mod\ (10)).$$

Решение $b = 3$. Цифровая подпись представляет собой пару: $a = 6$, $b = 3$.

Для проверки подписи необходимо убедиться, что

$$Y^a a^b\ (mod\ P) = G^{m'}\ (mod\ P);\ 3^6 \bmod (11) = 2^5 \bmod (11).$$

После этого отправитель передает подписанное сообщение и открытый ключ $Y = 3$. Приняв (M и Y), получатель вычисляет хэш-значение для сообщения M: $m' = h(M)$, $m' = 5$, а затем вычисляет два числа:

1) $Y \cdot {}^{a}a^{b} \ (mod\ P) = 3^{6} \cdot \ 6^{3} \ (mod\ (11)) = 10 \ (mod\ (11))$;

2) $G^{m} \ (mod\ P) = 2^{5} \ (mod\ (11)) = 10 \ (mod\ (11))$.

Поскольку эти два значения чисел равны, принятое получателем сообщение признаётся подлинным.

Следует отметить, что по сравнению со схемой цифровой подписи *RSA* схема цифровой подписи Эль Гамаля имеет ряд преимуществ:

- в связи с тем, что при заданном уровне стойкости алгоритма цифровой подписи целые числа, участвующие в вычислениях, имеют запись на 1/4 короче, что уменьшает сложность вычислений почти в два раза и позволяет заметно сократить объем используемой памяти;
- схема цифровой подписи Эль Гамаля имеет всего два достаточно просто проверяемых условия, так как при выборе модуля P достаточно проверить, что у числа (P–1) имеется большой простой делитель;
- без знания секретного ключа K_c процедура формирования подписи по схеме Эль Гамаля не позволяет вычислить цифровые подписи под новыми сообщениями.

По сравнению со схемой подписи *RSA* алгоритм подписи Эль Гамаля имеет длину цифровой подписи в 1,5 раза больше, что, в свою очередь, увеличивает время её вычисления, что является недостатком этого алгоритма.

6.4. Алгоритм цифровой подписи *DSA*

Национальный институт стандартов и технологии (НИСТ) США опубликовал федеральный стандарт обработки информации *FIPS PUB* 186, известный также как *DSS* (Digital Signature Standard – стандарт цифровой подписи) [25]. Стандарт *DSS* основан на алгоритме хэширования *SHA* (Secure Hash Algorithm – защищенный алгоритм хэширования) и представляет новую технологию использования цифровой подписи – алгоритм *DSA* (Digital Signature Algorithm – алгоритм цифровой подписи).

DSA представляет собой вариант алгоритма Эль Гамаля в модификации Шнора. Алгоритм использует следующие параметры: p – простое число длиной l битов, где l принимает значение, кратное 64, в диапазоне от 512 до 1024; q - 160-битовое простое число – делитель $p - 1$.

$$g = h^{(p-1)/q} \bmod p,$$

где h – любое число, меньшее $p - 1$, такое, что $h^{(p-1)/q} \bmod p > 1$.

$$y = g^{x} \bmod p,$$

где $x < q$. В алгоритме также используется однонаправленная хэш-функция $H(m)$ стандарта *SHS*.

Первые три параметра p, q и g открыты и могут быть общими для абонентов криптосети. Число y является открытым ключом и передается всем получателям документов. Секретным ключом является число x.

Рассмотрим алгоритм формирования ЦП по схеме *DSA*:

1. Отправитель генерирует случайное число $k < q$ и вычисляет

$$r = (g^{k} \bmod p) \bmod q,$$

$$s = (k^{-1} (H(m) + xr) \bmod q.$$

Подписью отправителя служат числа r и s.

1. Получатель проверяет подпись, вычисляя

$$w = s^{-1} \bmod q,$$

$$u_1 = (H(m) \cdot w) \bmod q,$$

$$u_2 = (rw) \bmod q,$$

$$v = (g^{u_1} \cdot g^{u_2} \bmod p) \bmod q.$$

Если $v = r$, то подпись под документом признается получателем правильной.

По сравнению с алгоритмом цифровой подписи Эль Гамаля алгоритм *DSA* обладает рядом преимуществ:

- при любой паре чисел g и p (от 512 до 1024) числа q, x, r, s имеют длину по 160 бит, сокращая длину подписи до 320 бит, при любом допустимом уровне стойкости;
- большинство операций с числами k, r, s, x при вычислении подписи производится по модулю числа q длиной 160 бит, что сокращает время вычисления подписи;
- при проверке подписи большинства операций с числами u_1, u_2, v, w также производится по модулю числа q длиной 160 бит, что сокращает объем памяти и время вычисления.

Алгоритм *DSA* не позволяет получать максимальное быстродействие, так как при подписывании и при проверке подписи приходится выполнять сложные операции деления по модулю q:

$$S = (k^{-1}(H(m) + rx)) \ (mod\ q), \quad w = (1/s) \ (mod\ q),$$

что является недостатком алгоритма *DSA*.

6.5. Алгоритм цифровой подписи ГОСТ Р34.10-94

Это отечественный стандарт цифровой подписи, называемый ГОСТ Р 34.10-94 [29]. Алгоритм похож на алгоритм *DSA* и использует следующие параметры:

p – большое простое число, длина которого лежит в диапазоне либо 509 – 512 битов, либо 1020 –1024 битов;

q – простое число – множитель p - 1, длиной от 254 до 256 битов;

a – любое число, меньшее $p - 1$, для которого $a^q \bmod p = 1$;

K_c – число, меньше q;

$K_o = a^{K_c} \bmod p$.

Этот алгоритм использует также однонаправленную хэш-функцию $H(x)$. Стандарт ГОСТ З 34.11-94 определяет хэш-функцию, основанную на симметричном алгоритме ГОСТ 28147-89.

Первые три параметра p, q, a открыты и могут совместно использоваться всеми пользователями сети. Число K_c является секретным ключом, число K_o – открытым ключом.

Чтобы подписать сообщение m, а потом проверить подпись, необходимо предпринять следующие шаги:

- отправитель генерирует случайное число k', причём $k' < q$ и вычисляет значения

$$r = (a^{k'} \bmod p) \bmod q,$$

$$s = (K_c * r + k' (H(m))) \bmod q.$$

Если $H(m) \bmod q = 0$, то его принимают равным 1. Если $r = 0$, то выбирают другое значение k' и начинают снова. Подписью служат два числа $r \bmod 2^{256}$ и $s \bmod 2^{256}$.

Затем получатель проверяет полученную подпись, вычисляя

$$v = H(m)^{q-2} \bmod q,$$

$$Z_1 = (s \cdot v) \bmod q,$$

$$Z_2 = ((q - r) \cdot v) \bmod q,$$

$$U = ((a^{Z_1} \cdot K_o{}^{Z_2}) \bmod p) \bmod q.$$

Если $u = r$, то подпись считается подлинной.

Этот алгоритм отличается от *DSA* уравнением для подсчёта s, в *DSA* значение $s = (K_c \cdot r + k'^{-1} (H(m))) \bmod q$, что дает другое уравнение проверки.

В отечественном стандарте электронной цифровой подписи параметр q имеет длину 256 бит, тогда как большинство западных криптографов довольствуются параметром $q \approx 160$ бит.

6.6. Алгоритм цифровой подписи на базе эллиптических кривых *ECDSA*

Для реализации данного алгоритма на первом этапе реализации цифровой подписи должны быть определены следующие входные параметры:

- конечное поле $GF(p)$;
- эллиптическая кривая $E(GF(p))$;
- большой простой делитель количества точек кривой n;
- точка G, координаты которой имеют тот же порядок, что и число n.

Каждый пользователь системы генерирует пару ключей следующим образом:

- выбирает случайное целое число d, $1 < d < n$ -1;
- вычисляет точку $Q = dG$.

При этом секретным ключом пользователя является число d, открытым ключом - точка Q.

Алгоритм генерации подписи

Для реализации данного алгоритма необходимо:

1) выбрать случайное целое число k, взаимно простое с n (поскольку n является простым числом по определению, данное условие выполняется автоматически), $1 \le k \le n$- 1;

2) Вычислить $(x_1,y_1) = [k]G$.

3) Вычислить $r = x_1 \ mod \ n$.

4) Если $r = 0$, то вернуться к шагу 1.

5) Вычислить $z = k^{-1} \ mod\ n$.

6) Вычислить хэш значение сообщения $e = h(m)$.

7) Вычислить $s = z(e + dr)\ mod\ n$.

8) Если $s = 0$, то вернуться к шагу 1.

9) Вывести пару (r,s) как подпись к сообщению m.

Алгоритм проверки цифровой подписи

1) Если условия $1 \le r, s \le n - 1$ нарушаются, то вывести «подпись фальшивая» и завершить работу алгоритма.

2) Вычислить $e = h(m)$.

3) Вычислить $w = s^{-1} \ mod\ n$.

4) Вычислить $u_1 = ew\ mod\ n$.

5) Вычислить $u_2 = rw\ mod\ n$.

6) Вычислить $X = [u_1]G + [u_2]Q = (x_1,y_1)$.

7) Если $r = x_1\ mod\ n$, то вывести «подпись действительная», иначе «подпись фальшивая», и завершить работу алгоритма.

Проверка корректности алгоритма генерации подписи

Докажем, что любая подпись, сгенерированная по алгоритму, рассмотренному выше, будет «действительной» согласно алгоритму проверки подписи.

Прежде всего, заметим, что параметры r и s, вычисляемые в данном алгоритме, не превосходят $n - 1$, как остатки при делении на модуль n. С другой стороны, выполняется проверка того, что $r, s \neq 0$ на шагах 4 и 8 алгоритма. Далее, согласно шагам 5 и 7 алгоритма генерации подписи, имеем $ks \equiv e + dr (mod\ n)$. Поскольку $w = s^{-1} mod\ n$, то $k \equiv we + wrd\ (mod\ n)$, а так как $Q = dG$ и точка G имеет порядок n, то

$$[k]\,G = [we + wrd]\,G = [we]\,G + [wr]\,[d]\,G =$$
$$= [we]\,G + [wr]\,Q = [u_1]\,G + [u_2]\,Q = X.$$

Таким образом, точка X, получаемая на шаге 6 алгоритма проверки подписи, совпадет с точкой $[k]G$, сгенерированной при получении подписи по алгоритму генерации. Первая координата X будет равна x_1, и ее остаток *mod n* будет равен r (согласно шагу 3 алгоритма генерации подписи). Корректность доказана.

6.7. Цифровые подписи с дополнительными функциональными свойствами

Такие подписи объединяют базовую схему цифровой подписи со специальным протоколом, который обеспечивает достижение дополнительных свойств, которыми базовая схема цифровой подписи не обладает.

К схемам цифровой подписи с функциональными дополнительными свойствами относятся:

- схемы слепой (blind) подписи;
- схемы неоспоримой (undeniable) подписи.

6.7.1. *Схемы слепой подписи*

Целью схемы слепой подписи [17] является воспрепятствование подписывающему лицу не только ознакомиться с сообщением отправителя, которое он подписывает, но и с подписью под этим сообщением. Это не дает возможности в дальнейшем связать подписанное сообщение с отправителем. Схемы подписи в слепую являются двусторонними протоколами между отправителем и стороной, подписывающей документ.

Данная схема используется в тех случаях, когда отправитель, например клиент банка, не хочет, чтобы подписывающая сторона то есть банк смог в дальнейшем связать сообщение m и подпись s_b (m) с определенным шагом, выполненного ранее протокола

(транзакции).

Для реализации протокола слепой подписи необходимо выбрать:

1 - механизм генерации цифровой подписи для подписывающей стороны *B* $s_b(m) = m^{K_c} \bmod N$**;**

2 - функции *f*(*m*) и *g*(*m*), (известны только отправителю) должны подчиняться следующим требованиям:

$g\, s_b(f(m)) = s_b(m)$,

где *f*(*m*) – замаскированное сообщение *m*, определяемое из соотношения $f(m) = m \cdot k'^{K_o} \bmod N$,

***g*(*m*) – демаскирующая функция, определяемая по формуле**

$g(m) = k'^{-1} \bmod N$.

Согласно требованию 2 получаем

$$g s_b(f(m)) = k'^{-1} m^{K_c} k'^{K_o K_c} \bmod N = k'^{-1} k' m^{K_c} \bmod N =$$

$$= m^{K_c} \bmod N = s_b(m).$$

Если в качестве примера генерации слепой подписи $s_b(m)$ выбрать асимметричную криптосистему (*RSA*) с открытым ключом (N,K_o), секретным ключом K_c и случайным секретным числом k', удовлетворяющим условиям: $0 \le k' \le N$-1 и НОД (N, k'), то общий

протокол слепой подписи включает последовательность следующих действий:

- **отправитель вычисляет замаскированное сообщение**

$m^* = mk'^{K_o} \bmod N$ **и посылает его стороне *B*;**

- **подписывающая сторона *B* генерирует цифровую подпись**

$s^* = (m^*)^{K_c} \bmod N$ **и посылает эту подпись отправителю;**

- **отправитель вычисляет подпись** $s_b(m) = k'^{-1} s^* \bmod N$**, которая является подписью *B* на сообщение *m*.**

Проверка корректности данного алгоритма показывает, что любая подпись, сгенерированная по данной схеме, будет действительна, если выполняются два следующих соотношения:

$(m^*)^{K_c} \equiv (mk'^{K_o})^{K_c} \equiv m^{K_c} k' \pmod N$,

$k'^{-1} s^* \equiv m^{K_c} k' k^{-1} \equiv m^{K_c} \pmod N$.

В качестве примера рассмотрим общую схему платежной системы с использованием слепой подписи.

6.7.2. *Схема алгоритма платежной системы*

Совершенно уникальные возможности для реализации возможностей электронных платежных систем [26] предоставляют так называемые электронные деньги. Фактически они выступают электронным эквивалентом бумажных денег, хотя при этом у них есть и своя специфика:

- электронные деньги могут пересылаться с использованием Interneta или обычных телефонных каналов;
- для их хранения могут использоваться как физические устройства (например, смарт-карты), так и рабочая станция пользователя;
- электронные деньги (как, впрочем, и любую другую информацию, представленную в электронном виде) можно легко размножить или скопировать, поэтому при построении систем, имеющих дело с подобной формой платежей, необходимо обеспечить уникальность каждой электронной банкноты;
- передача по общедоступным каналам связи приводит к тому, что появляется потенциальная возможность отследить, где, когда и какие средства были потрачены тем или иным субъектом. Понятно, что это обстоятельство приводит к необходимости разрабатывать особые меры по обеспечению анонимности плательщика. Вывод очевиден - создание электронных денег невозможно без использования как широко известных криптографических механизмов, так и совершенно новых идей, например затемняющей подписи и др.

Рассмотрим схему построения платежных систем с использованием электронных денег

Представляемая здесь система построена с участием третьей

стороны - банка или другой финансовой организации, в функции которой входит как создание электронных монет, так и проверка их подлинности и уникальности при проведении платежа между покупателем и продавцом.

В соответствии с предлагаемой схемой пользователь на первом этапе должен пройти процедуру регистрации, которая заключается в следующем:

1. Пользователь создает пару ключей для подписи: открытый - KP и секретный - $KP_{\text{с}}$.

2. Банк создает сертификат для пользователя, содержащий имя банка (B), имя пользователя (P), срок действия сертификата (Δt), сумму денежного лимита пользователя (L_P), открытый ключ пользователя (KP), и подписывает его своим секретным ключом $KB_{\text{с}}$. В итоге сертификат пользователя имеет следующий вид: $CP = \{B, P, \Delta t, L_P, KP\}_{KB_{\text{с}}}$.

3. Банк создает общий симметричный ключ банк-пользователь -(K) и разделяет его знание с пользователем.

4. Банк создает электронные ДЕ, представляющие 64-битные последовательности, зашифрованные на общем симметричном ключе K и содержащие серийный номер ДЕ банка, выпустившего ее. Для простоты изложения предположим, что все ДЕ имеют одну и ту же стоимость и объединяются в пачки - CS по k ДЕ в каждой.

5. Каждая ДЕ имеет следующий вид: $C_i=\{i||j||B\}K$, где i - номер ДЕ в пачке, j - номер пачки и || - операция конкатенации.

6. Банк подписывает пачки ДЕ с указанием времени подписи t

$$CS = C_1, C_2, ..., C_k, \{B, t, C_1, ..., C_k\}_{KB_{\text{с}}}.$$

7. При осуществлении оплаты покупатель выбирает хэш-функцию h и вычисляет:

$$h_i = h(C_i), \text{ где } i = 1 ,\ldots, k.$$

8. Покупатель посылает продавцу сертификат пользователя – CP и следующее сообщение: $S_P=\{ M, t, h_i , P, B \}_{KP_{\text{с}}}$, где M - имя

продавца, t - время подписи.

9. Продавец проверяет подпись покупателя, используя сертификат покупателя *CP*, выданный банком. Если подпись достоверна, то пользователь проходит регистрацию.

6.7.3. *Схема организации процесса платежей*

1. Покупатель, желая потратить ДЕ - C_1, посылает продавцу значения C_1 и h_1.

2. Продавец проверяет равенство $h_1 = h(C_1)$ и в случае его выполнения ДЕ C_1 принимается.

3. Если покупателю необходимо потратить другие ДЕ, то и в этом случае производятся аналогичные операции.

В ходе проверки легитимности ДЕ покупателя продавец передает в банк сообщение S_p, полученные ДЕ C_1, ..., C_m и соответствующие их хэш-значения h_1, ...,h_m. Банк проверяет каждую ДЕ в отдельности, используя при этом формулу из шага 2, убеждается, что эти ДЕ не были потрачены раньше, и в случае подтверждения производит перевод денег на счет продавца.

Если продавец обслуживается в другом банке, нежели покупатель, то используются традиционные схемы взаимодействия между банком, выдавшим ДЕ, и банком продавца. Продавец представляет в свой банк следующие данные: M, P, B, t, S_P, C_1,...,C_m, h_1...h_m, которые в дальнейшем передаются через системы электронных платежей, например через клиринговую инфраструктуру.

Приведенная выше схема организации платежей на практике может быть реализована с использованием смарт-карт или других физических устройств, обеспечивающих хранение ключей пользователей, сертификатов, а также проведение вычислений (шаги 4 и 5). При этом физический вычислитель может использоваться в качестве еще одного уровня защиты от повторно-

го применения ДЕ. Недостатком данной системы является отсутствие анонимности действий покупателя, то есть банк может отслеживать проходящие платежи.

Платежные системы обычно имеют многоуровневую защиту, обеспечивающую устойчивость системы к различным атакам и несанкционированным действиям легальных пользователей. Основная задача обеспечения информационной безопасности решается при организации самих платежных систем (например, банкноты должны быть надежно защищены от подделки). Организовать на качественно новом уровне защиту от активных несанкционированных действий, направленных на нарушение выбранной политики безопасности, можно только с использованием смарт-карт.

Иной подход к организации электронных платежных систем, построенных на основе электронных ДЕ, был предложен Шаумом, которая основана на использовании цифровой подписи *RSA*.

6.7.4. *Схема алгоритма платежей по методу Шаума*

Процедура платежей по данной схеме [15,25,26] начинается с того, что банком выбираются и публикуются величины N и $f(n)$, где $N = pq$, $f(n)$- однонаправленная функция, где n фактически является номером ДЕ. Электронной ДЕ в целом в этих условиях считается пара $(n, f(n)^{1/Z} \ mod \ N)$, где Z характеризует номинал ДЕ. Для значений Z устанавливается соответствие с номиналом ДЕ - 1 ($Z = 3$), 2 ($Z = 5$), 3 ($Z = 3 \cdot 5$), 4 ($Z = 7$), 5 ($Z = 3 \cdot 7$), … 10 ($Z = 5 \cdot 11$) и т.д.

Например, для получения от банка ДЕ достоинством 10 условных единиц (у. е.) ($Z = 5 \cdot 11$), пользователь выбирает случайное значение n_1 и вычисляет $f(n_1)$. Далее пользователю необходимо пройти в банке процедуру подписи данной электронной ДЕ, то есть вычислить корень степени Z. Но пользователь не может послать $f(n_1)$, поскольку, произведя оплату конкретной ДЕ, банк в дальнейшем узнает ее и анонимность

действий пользователя будет нарушена. Поэтому пользователь выбирает число r_1 (называемое затемняющим множителем), возводит его в степень Z и отправляет банку значение $f(n_1)r_1^Z \bmod N$. Получив эту величину, банк извлекает из нее корень степени Z по модулю N и отправляет пользователю полученное значение, далее пользователь снимает затемняющий множитель и получает ДЕ $f(n_1)^{1/Z} \bmod N$.

6.7.5. ***Схема неоспоримой подписи***

Неоспоримая подпись [22] (подпись, не допускающая подлога) зависит от подписанного документа и секретного ключа так же, как и обычная цифровая подпись, однако она не может быть верифицирована без участия лица, поставившего эту подпись.

Д. Чом разработал следующий алгоритм неоспоримой цифровой подписи. Рассмотрим алгоритм генерации ключей, с помощью которого отправитель A, подписывающий сообщение, выбирает секретный и открытый ключи.

Отправитель должен сделать следующее:

- **выбрать случайное простое число $p = 2q + 1$, где q – простое число;**
- **выбрать генераторное число α для подгруппы порядка q в циклической группе Z_p;**
- **выбрать случайный элемент $\beta \in Z^*_p$ и вычислить**

$\alpha = \beta^{(p-1)/q} \bmod p$;

- если $\alpha = 1$, тогда необходимо вернуться к предыдущему шагу;

- выбрать случайное целое $K_c \in \{1, 2, \ldots, q-1\}$ и вычислить

$$y = \alpha^x \bmod p;$$

- для отправителя открытый ключ равен (p, α, y), секретный ключ равен K_c.

Согласно алгоритму отправитель подписывает сообщение m, принадлежащее подгруппе порядка q в Z^*_p. Получатель может проверить эту подпись при участии отправителя.

В работе алгоритма неоспоримой подписи можно выделить следующие этапы:

- генерация подписи, когда отправитель вычисляет $s = m^{K_c} \bmod p$, где s – подпись отправителя на сообщении m, которое отсылается получателю;

- верификация подписи, выполняемая получателем с участием отправителя.

Она включает в себя следующие этапы:

- получение получателем подлинного открытого ключа отправителя;

- выбор получателем двух случайных секретных целых чисел

$$a, b \in \{1, 2, \ldots, q\text{-}1\};$$

- вычисление получателем $Z = s^a y^b \bmod p$ и отправка значения Z отправителю;

- вычисление отправителем $w = (Z)^{1/K_c} \bmod p$, где $K_c K_c^{-1} \equiv 1 (\bmod p)$, и отправка значения w получателю;

- вычисление получателем значения $w' = m^a \alpha^b \bmod p$ и признание подписи s подлинной, если $w = w'$.

Итак,

$$w \equiv (Z)^{1/K_c} \equiv (s^a y^b)^{1/K_c} \equiv (m^{K_c a} \alpha^{K_c b})^{1/K_c} \equiv m^a \alpha^b \equiv w' \bmod p,$$

что и требовалось доказать, т. е. неоспоримая подпись может быть верифицирована только путем непосредственного взаимодействия с отправителем сообщения.

ГЛАВА 7. ПРОТОКОЛЫ УПРАВЛЕНИЯ КРИПТОГРАФИЧЕСКИМИ КЛЮЧАМИ

Эффективность криптографической защиты информации в компьютерных системах и сетях определяется стойкостью используемых алгоритмов криптографических преобразований и надёжностью протоколов управления ключами.

В понятие управления ключами входит совокупность методов решения таких задач, как:

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

Правильное решение перечисленных задач имеет огромное значение, так как в большинстве случаев криптоаналитику проще провести атаку на ключевую подсистему, а не на сам алгоритм криптографической защиты. Использование стойкого алгоритма шифрования является необходимым, но далеко не достаточным условием построения надёжной системы криптографической защиты информации. Используемые в процессе информационного обмена ключи нуждаются в не менее надёжной защите на всех стадиях своего жизненного цикла.

Под алгоритмом хранения ключей понимают организацию их безопасного хранения, учета и удаления. Секретные ключи никогда не должны записываться в явном виде на носителе, который может быть считан или скопирован.

В данной главе основное внимание уделяется протоколам распределения ключей на основе симметричных и асимметричных криптосистем. Протоколы распределения ключей должны обеспечивать взаимную аутентификацию сторон, целостность сообщений и защиту от повторных запросов. Вопросы генерации, хранения и уничтожения ключей подробно рассмотрены в [1,5,21].

Длина ключа определяет верхнюю границу стойкости криптосистемы. Криптоаналитик всегда может воспользоваться силовой атакой, которая состоит в тотальном переборе по всему пространству возможных ключей. Однако размер этого пространства растёт по экспоненциальному закону при увеличении длины ключа. Например, если длина ключа $l = 256$, то число возможных ключей составляет более 10^{76}.

В настоящее время перебор 10^{76} вариантов ключей представляется недостижимым не только для современных вычислительных технологий, но и в обозримом будущем.

7.1. Генерация ключей

Безопасность любого криптографического алгоритма определяется используемым криптографическим ключом, который должен иметь достаточную длину и случайные значения битов.

Для генерации случайных значений ключей используются различные аппаратные и программные средства. Поскольку степень случайности генерации чисел должна быть достаточно высокой, то используются устройства на основе «натуральных» случайных процессов, например на основе белого радиошума.

В качестве примера рассмотрим метод генерации сеансового ключа для симметричных криптосистем, описанный в стандарте *ANSI X* 9.17 [8,24], который предполагает использование криптографического алгоритма *DES*.

Схема генерации случайного сеансового ключа R_i показана на рис. 7.1, где введены следующие обозначения:

E_K – результат шифрования алгоритмом *DES* значения X;

K - ключ, зарезервированный для генерации секретных ключей;

t – временная отметка.

Случайный ключ R_i генерируют, вычисляя значение

$$R_i = E_K(E_K(t_i) \oplus V_i).$$

$V_{i=0}$ – секретное 64-битовое начальное число.

Следующее значение V_{i+1} вычисляют по формуле

$$V_{i+1} = E_K(E_K(t_i) \oplus R_i).$$

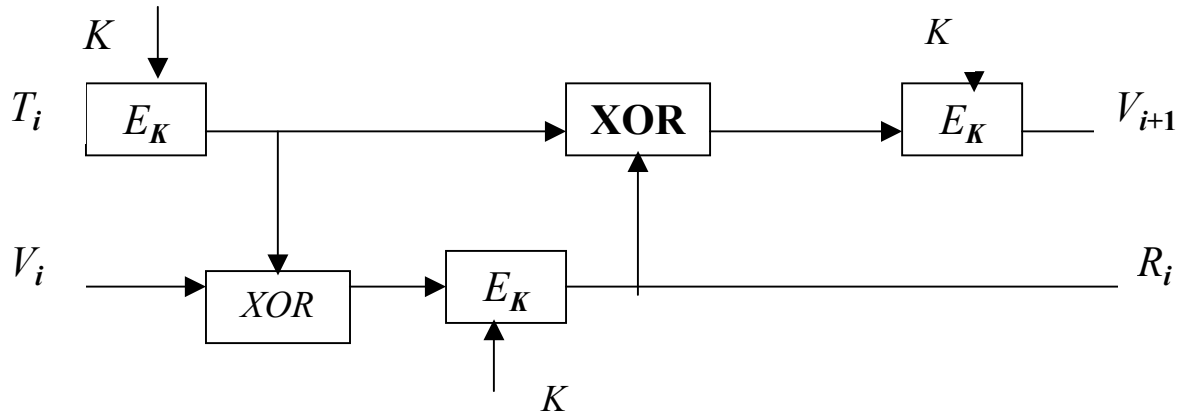

Рис. 7.1. Схема генерации случайного ключа R_i

В случае необходимости 128-битового ключа, генерируют пару ключей R_i, R_{i+1} и объединяют их.

Ключ необходимо регулярно менять, для того, чтобы избежать его раскрытия и утечки информации. Для симметричной криптосистемы замену ключа можно осуществить, используя процедуру модификации ключа, то есть это генерирование нового ключа из его предыдущего значения с помощью однонаправленной функции.

Участники информационного обмена, использующие один и тот же секретный сеансовый ключ, одновременно вводят его значение в качестве аргумента в однонаправленную функцию, получая один и тот же результат. Чтобы создать новое значение ключа, берут определенные биты из этих результатов.

Процедура модификации ключа работоспособна, однако новый ключ безопасен в той же мере, в какой был безопасен прежний.

Для асимметричных криптосистем с открытыми ключами генерация ключей намного сложнее, так как они должны обладать определенными математическими свойствами.

7.2. Носители ключевой информации

Основными носителями ключевой информации могут быть пластиковые карты, устройства хранения типа touch-memory, магнитные дискеты и т.д.

Электронные пластиковые карты в настоящее время наиболее распространены и являются универсальными носителями конфиденциальной информации, позволяющими идентифицировать и аутентифицировать пользователей, хранить коды, пароли и криптографические ключи.

Смарт-карты эффективно используются не только для хранения ключевой информации, но и широко используются в электронных платёжных системах, транспорте и т.д. Эти миниатюрные интеллектуальные карты со встроенным микрочипом представляют собой микрокомпьютеры, содержащие кроме центрального процессора постоянное и оперативное запоминающие устройства, а также репрограммируемое постоянное запоминающее устройство, объемом памяти до 64 Кб.

Устройство хранения ключей типа touch-memory (ТМ) представляет собой энергонезависимую память, размещенную в металлическом корпусе.

В структуру ТМ входят следующие блоки:

1. Постоянное запоминающее устройство, хранящее 64-разрядный код, состоит из байтового кода типа прибора, 48-битового уникального серийного номера и 8-битовой контрольной суммы. Содержимое ПЗУ уникально и не может быть изменено в течение всего срока службы прибора.

2. Оперативное запоминающее устройство ёмкостью от 128 до 8192 байт содержит практически все модификации ТМ с защитой оперативной памяти от несанкционированного доступа.

Магнитные диски (МД) также являются носителями ключевой информации. Для обеспечения надёжного хранения ключевой информации на МД применяют как минимум двукратное резервирование хранимой информации, что позволяет защитить ключевую информацию от ошибок при считывании с МД. Для предотвращения возможности перехвата ключевой информации в процессе её считывания с ГМД применяют хранение ключевой информации в зашифрованном виде.

7.3. Иерархия ключей

Любая информация об используемых ключах должна храниться в зашифрованном виде.

Необходимость хранения и передачи ключей, зашифрованных с помощью других ключей, приводит к концепции иерархии ключей. В настоящее время принята следующая иерархия ключей: главный ключ (ГК), ключ шифрования ключей (КК), ключ шифрования данных (КД).

Иерархия ключей может быть:

- двухуровневой (КК /КД);
- трёхуровневой (ГК /КК /КД).

На самом нижнем уровне находятся рабочие, или сеансовые КД, использующиеся для шифрования данных, персональных идентификационных номеров *PIN* и аутентификации сообщений. Если эти ключи необходимо зашифровать с целью защиты их при передаче или хранении, то применяют ключи следующего уровня, а именно ключи шифрования ключей, которые никогда не используются как рабочие и наоборот. Такое разделение функций обеспечивает максимальную безопасность, т.е. стандарт устанавливает, что различные типы рабочих ключей должны всегда шифроваться с помощью различных версий ключей шифрования ключей.

На верхнем уровне иерархии ключей располагается главный ключ, мастер ключ, который применяют для шифрования КК, при необходимости хранения их на диске. Мастер ключ распространяется между участниками обмена при личном контакте для исключения его перехвата и компрометации, так как раскрытие противником значения главного ключа полностью уничтожает защиту компьютера.

На практике мастер ключ компьютера создается случайным выбором изо всех возможных значений ключей. Его помещают в защищённый не только от считывания и записи, но и от механических воздействий блок криптографической системы так, чтобы было невозможно раскрыть значение этого ключа.

Важное условие безопасности информации заключается в периодическом обновлении ключевой информации в АСОИ. При этом должны переназначаться как рабочие, так и мастер ключи. Вопрос обновления ключевой информации тесно связан с третьим элементом управления ключами – распределением ключей.

К процессу распределения ключей предъявляются следующие требования:

- оперативность и точность распределения;
- конфиденциальность и целостность распределения ключей.

Для распределения ключей между пользователями компьютерной сети используются следующие основные способы [8]:

- использование одного или нескольких центров распределения ключей;
- прямой обмен сеансовыми ключами между пользователями сети.

Недостаток первого подхода заключается в том, что центру распределения ключей известно, кому и какие ключи распределены, и это позволяет читать все сообщения, передаваемые по сети. Возможные злоупотребления существенно влияют на защиту. При втором подходе проблема состоит в том, чтобы надёжно удостовериться в подлинности субъектов сети.

В обоих случаях должна быть обеспечена подлинность сеанса связи. Этого можно достичь, используя механизм запроса-ответа или отметки времени.

Механизм запроса-ответа заключается в следующем. Пользователь *A* включает в посылаемое сообщение для пользователя *B* непредсказуемый элемент (например, случайное

число). При ответе пользователь *B* должен выполнить некоторую операцию с этим элементом, что невозможно осуществить заранее, поскольку неизвестно, какое случайное число придёт в запросе. После получения результата действия пользователя *B* пользователь *A* может быть уверен, что сеанс является подлинным.

Механизм отметки времени предполагает фиксацию времени для каждого сообщения. Это позволяет каждому субъекту сети определить временной интервал пришедшего сообщения, и отвергнуть его, если появится сомнение в его подлинности. При использовании отметок времени необходимо установить допустимый временной интервал задержки.

В обоих случаях для защиты элемента контроля используют шифрование, чтобы быть уверенным, что ответ отправлен не криптоаналитиком и не изменен штемпель отметки времени.

Таким образом, задача распределения ключей сводится к построению протокола распределения ключей, обеспечивающего:

- взаимное подтверждение подлинности участников сеанса;
- подтверждение достоверности сеанса механизмом запроса-ответа или отметки времени;
- использование минимального числа сообщений при обмене ключами;
- возможность исключения злоупотреблений со стороны центра распределения ключей (вплоть до отказа от него).

В связи с этим основу решения задачи распределения ключей целесообразно положить принцип отделения процедуры идентификации партнера от процедуры распределения ключей. Цель такого подхода состоит в создании метода, при котором после установления подлинности участники сами формируют сеансовый ключ без участия центра распределения ключей с тем, чтобы распределитель ключей не имел возможности выявить содержание сообщений.

7.4. Распределение ключей с участием центра распределения ключей

При распределении ключей между участниками предстоящего информационного обмена должна быть гарантирована подлинность сеанса связи, для чего применяется модель рукопожатия, при которой ни один из участников не получит никакой секретной информации во время процедуры установления подлинности [8,25].

Взаимное установление подлинности гарантирует вызов нужного субъекта с высокой степенью уверенности, что связь установлена с нужным адресатом и никаких попыток подмены не было. Реальная процедура организации соединения между участниками информационного обмена включает как этап распределения, так и этап подтверждения подлинности партнёров. На следующем этапе осуществляется подтверждение подлинности участников через обмен удостоверяющими сообщениями, обеспечивающими возможность выявления любой подмены или повтора одного из предыдущих вызовов.

Рассмотрим более подробно протоколы аутентификации и распределения ключей для асимметричных криптосистем и комбинированного метода шифрования.

7.5. Информационная безопасность коммуникационных связей

Как известно, криптография решает проблемы секретности и проверки подлинности передаваемого сообщения. О криптографических алгоритмах можно знать все, но если не уметь применять их для решения конкретных проблем, то наши знания будут чисто академическими. При этом необходимо помнить, что невозможно сделать или узнать больше, чем определено протоколом. Именно поэтому необходимо обсудить последовательность целенаправленных действий при решении конкретных задач.

Протоколом называют последовательность действий, исполняемых двумя и более сторонами, предназначенную для решения какой-либо задачи. Протокол исполняется последовательно, от начала и до конца. Каждое действие выполняется поочередно, причем каждое последующее действие начинает исполняться только после окончания предыдущего.

Кроме того, каждый участник протокола должен знать не только всю последовательность его действий, но и строго следовать протоколу, т.е. каждое действие должно быть определено и в нем указаны точные действия в любой возможной ситуации. Информационная безопасность коммутационных связей обеспечивается протоколом, в котором используются криптографические методы защиты информации. Пользователи сети могут безоговорочно доверять либо не доверять друг другу, даже при получении сообщений о текущем времени. Однако следует иметь в виду, что компьютерам нужны только формальные протоколы, устойчивые к разрушениям.

Помимо формализации действий, протоколы позволяют отделить процесс решения задачи от механизма решения. Поэтому мы можем исследовать протокол, не забираясь в дебри подробностей его реализации.

7.5.1. *Процесс организации коммуникационной связи с помощью симметричной криптосистемы*

Каким образом две стороны могут обмениваться секретной информацией? Конечно же, шифруя сведения. Рассмотрим процесс обмена шифрованной информацией между пользователями сети *A* и *B*.

1) *A* и *B* выбирают симметричную криптосистему.

2) *A* и *B* выбирают ключ.

3) *A* шифрует текст своего сообщения, используя выбранный

ключ. Таким образом, он создает шифртекст сообщения.

4) *A* посылает шифртекст сообщения *B*.

5) *B* расшифровывает шифртекст сообщения, используя выбранный ключ, для получения открытого текста сообщения.

Как известно, безопасность криптосистемы всецело зависит от знания ключа и ни в коей мере - от знания алгоритма. Именно поэтому в криптографии так важно управление распределением ключей. Используя симметричный алгоритм, *A* и *B* могут, не скрываясь, выполнить этап 1, но этап 2 они должны выполнить тайно. Ключ должен храниться в тайне до, после и во время работы протокола - словом, до тех пор, пока должно сохраняться в тайне передаваемое сообщение. В противном случае сообщение будет немедленно раскрыто. Этот протокол действует пока *A* и *B* доверяют друг другу. Рассмотрим недостатки симметричных криптосистем.

Поскольку знание ключа позволяет раскрыть все сообщения, распространение ключей должно выполняться в тайне. Ключи столь же ценны, что и все сообщения, зашифрованные ими, поэтому для известных в настоящее время криптосистем задача распределения ключей является весьма серьезной.

Если ключ будет скомпрометирован, то криптоаналитик может расшифровать все сообщения, зашифрованные этим ключом. Более того, он сможет выступать в качестве одной из сторон и создавать ложные сообщения.

Если допустить, что каждая пара пользователей сети будет использовать отдельный ключ, общее число ключей быстро возрастет с ростом числа пользователей. Действительно, в сети из n пользователей необходимы $n(n\text{-}1)/2$ ключей. Например, для общения между собой 100 пользователям необходимо 4950 различных ключей. Для решения данной проблемы можно уменьшить число пользователей сети, однако это не всегда

возможно.

7.5.2. *Процесс организации коммуникационной связи с помощью*

криптосистемы с открытым ключом

Как известно, в криптосистемах с открытым ключом используются два разных ключа - один открытый, а другой закрытый. Вычислительными методами очень трудно определить закрытый ключ по открытому ключу.

С точки зрения математики в основе технологии шифрования лежат рассмотренные в главе 5 однонаправленные хэш-функции с потайным входом. Шифрование выполняется легко. Инструкции по шифрованию общедоступны - зашифровать сообщение может любой человек. Расшифрование, наоборот, очень сложно. Секретом, или потайным входом, служит закрытый ключ. Если знать этот секрет, расшифрование столь же просто, как и зашифрование.

Рассмотрим процесс организации связи между пользователями сети с помощью криптосистемы с открытым ключом:

1) *A* и *B* договариваются использовать криптосистему с открытым ключом.

2) *B* посылает *A* свой открытый ключ.

3) *A* шифрует свое сообщение, используя открытый ключ *B*, и отсылает его *B*.

4) *B* расшифровывает сообщение *A* своим закрытым ключом.

Как нетрудно заметить, криптография с открытым ключом устраняет болезненную для симметричных криптосистем проблему распространения ключей. При использовании криптосистемы с симметричным ключом *A* и *B* должны были решить проблему тайного распределения секретного ключа. Криптография с

открытым ключом упрощает эту задачу: *A* может отправить *B* секретное сообщение без какой-либо предварительной подготовки.

Однако и в этом случае используемая криптосистема должна заранее согласовываться с пользователями сети, то есть у каждого пользователя есть открытый и закрытый ключи.

7.5.3. *Процесс организации коммуникационной связи с помощью смешанных (гибридных) криптосистем*

Как известно, алгоритмы с открытым ключом не заменяют симметричные алгоритмы. Они используются для шифрования не самих сообщений, а ключей. Этому есть две причины:

1. Алгоритмы с открытым ключом исполняются медленно. Симметричные алгоритмы, по крайней мере, в 1000 раз быстрее алгоритмов с открытым ключом. Однако компьютеры становятся более мощными, и лет через 5 компьютеры будут способны обеспечить скорость криптографии с открытым ключом, сопоставимую с сегодняшней скоростью симметричной криптографии. Но требования к пропускной способности тоже возрастают, поэтому всегда будет необходимость шифровать данные быстрее, чем это может обеспечить криптография с открытым ключом.

2. Криптосистемы с открытым ключом уязвимы к атакам на основе подобранного открытого текста. Если $C = E(M)$, где M - открытый текст из множества n возможных открытых текстов, криптоаналитику достаточно зашифровать все n возможных открытых текстов и сравнить результаты с C (как известно, ключ зашифрования является открытым). Он не сможет таким путем восстановить ключ расшифрования, но сумеет определить M.

Криптоанализ на основе подобранного открытого текста особенно эффективен, если число возможных криптограмм этого сообщения относительно невелико. Симметричные криптосистемы неуязвимы к

вскрытиям этого типа, поскольку криптоаналитик, не зная ключ, не сможет выполнять пробные шифрования. С учетом изложенного выше, в большинстве практических реализаций криптография с открытым ключом используется только для засекречивания и распространения сеансовых ключей, а сеансовые ключи используются симметричными алгоритмами для защиты трафика сообщений. Такая организация коммуникационной связи называется смешанной криптосистемой. Рассмотрим процесс реализации протокола обмена сообщениями между пользователями сети с использованием смешанной криптосистемы.

1) *B* посылает *A* свой открытый ключ.

2) *A* генерирует случайный сеансовый ключ, шифрует его с помощью открытого ключа *B* и посылает его *B*.

3) Используя свой закрытый ключ, *B* расшифровывает сообщение *A*, восстанавливая сеансовый ключ.

4) Обе стороны шифруют свои сообщения с помощью одинакового сеансового ключа.

Использование криптографии с открытым ключом для распространения ключей решает эту очень важную проблему. При использовании данного протокола для зашифрования сообщения создается сеансовый ключ, который по завершении сеанса связи уничтожается. Это резко снижает опасность компрометации сеансового ключа. Конечно, закрытый ключ тоже уязвим к компрометации, но риск значительно меньше, так как во время сеанса этот ключ используется однократно - для шифрования сеансового ключа.

7.6. Протокол распределения ключей с помощью асимметричных криптосистем с использованием сертификата открытых ключей

Сертификатом открытого ключа называется сообщение центра распределения ключей (ЦРК) [3,21], удостоверяющее целостность некоторого открытого ключа объекта. Например, сертификат открытого ключа для пользователя *A*, обозначаемый C_A, содержит отметку времени *t*, идентификатор Id_A, открытый ключ K_A, зашифрованные секретным ключом ЦРК $k_{ЦРК}$, то есть

$$C_A = E_{k_{ЦРК}}(t, Id_A, K_A).$$

Отметка времени t используется для подтверждения актуальности сертификата и тем самым предотвращает повторения прежних сертификатов, содержащих открытые ключи и соответствующие секретные ключи, являющиеся несостоятельными.

Секретный ключ $k_{\text{ЦРК}}$ известен только менеджеру ЦРК. Открытый ключ $K_{\text{ЦРК}}$ известен участникам A и B.

Вызывающий объект A инициирует стадию установления ключа, запрашивая у ЦРК сертификат своего открытого ключа и открытого ключа участника B:

$A \rightarrow$ ЦРК: Id_A, Id_B «Вышлите сертификаты открытых ключей пользователей A и B».

Здесь Id_A и Id_B – уникальные идентификаторы соответственно участников A и B.

Менеджер ЦРК отвечает сообщением

ЦРК $\rightarrow A$: $E_{k_{\text{ЦРК}}}(t, Id_A, K_A), E_{k_{\text{ЦРК}}}(t, Id_B, K_B)$.

Участник A, используя открытый ключ ЦРК $K_{\text{ЦРК}}$, расшифровывает ответ ЦРК и проверяет оба сертификата. Идентификатор Id_B убеждает A, что личность вызываемого участника правильно зафиксирована в ЦРК и K_B – действительно открытый ключ участника B, поскольку оба зашифрованы ключом $k_{\text{ЦРК}}$.

Следующий шаг протокола включает установление связи A с B:

$A \rightarrow B$: C_A, $E_{k_A}(t), E_{K_B}(r_1)$.

Здесь C_A – сертификат открытого ключа пользователя A; $E_{k_A}(t)$ – отметка времени, зашифрованная секретным ключом участника A и являющаяся подписью участника A, так как никто другой не может создать такую подпись; r_1 – случайное число, генерируемое A и используемое для обмена с B в ходе процедуры проверки подлинности.

Если сертификат C_A и подпись A верны, то участник B уверен, что сообщение пришло от A. Часть сообщения $E_{K_B}(r_1)$ может расшифровать только B, так как никто другой не знает секретного ключа k_B, соответствующего открытому ключу K_B. Пользователь B расшифровывает значение числа r_1 и, чтобы подтвердить свою подлинность, посылает участнику A сообщение $B \rightarrow A$: $E_{K_A}(r_1)$.

Участник *A* восстанавливает значение r_1, расшифровывая это сообщение с использованием своего секретного ключа k_A. Если это ожидаемое значение r_1, то *A* получает подтверждение, что вызываемый участник действительно *B*.

Кроме того, способность систем с открытыми ключами генерировать цифровые подписи, обеспечивающие различные функции защиты, компенсируют избыточность требуемых вычислений.

7.7. Протокол прямого обмена ключами

Два пользователя, желающие обменяться криптографически защищённой информацией, при использовании для информационного обмена криптосистемы с симметричным секретным ключом, должны обладать общим секретным ключом и обменяться им по каналу связи безопасным образом. Если ключ меняется достаточно часто, то доставка его превращается в серьёзную проблему.

Для решения этой проблемы применяют два способа:

- использование асимметричной криптосистемы с открытым ключом для шифрования и передачи секретного ключа симметричной криптосистемы;

- использование системы открытого распределения ключей Диффи-Хеллмана.

Реализация первого способа, иногда называемого способом электронного цифрового «конверта», осуществляется в рамках комбинированной криптосистемы с симметричными и асимметричными ключами. При таком подходе симметричная криптосистема применяется для шифрования и передачи открытого текста, а асимметричная криптосистема с открытым ключом – для шифрования, передачи и последующего расшифрования только секретного ключа симметричной криптосистемы.

В качестве примера реализации способа электронного цифрового «конверта» рассмотрим порядок работы комбинированной криптосистемы с симметричными и асимметричными ключами с применением электронной цифровой подписи и сертификатов открытых ключей [3,21]:

1. Безопасно создаются и распространяются асимметричные открытые и закрытые ключи. Закрытый асимметричный ключ передаётся его владельцу. Открытый асимметричный ключ хранится в базе данных открытых ключей и администрируется центром выдачи сертификатов ЦРК. Пользователи должны доверять такой системе, где производится безопасное создание, распределение и администрирование ключей.

2. Создаётся электронная цифровая подпись текста с помощью вычисления его хэш-функции. Полученное значение шифруется с использованием асимметричного закрытого ключа отправителя, а затем полученная строка символов добавляется к передаваемому тексту (только отправитель может создать электронную подпись).

3. Создаётся секретный симметричный ключ, который будет использоваться для шифрования только данного сообщения или сеанса взаимодействия (сеансовый ключ); затем при помощи этого ключа и симметричного алгоритма шифрования шифруется исходный текст вместе с добавленной к нему электронной подписью, таким образом получается зашифрованный текст.

4. Далее необходимо решить проблему с передачей сеансового ключа получателю сообщения.

5. Отправитель должен иметь асимметричный открытый ключ центра выдачи сертификатов $K_{ЦРК}$. Перехват незашифрованных запросов на получение этого открытого ключа - это самая распространённая форма атаки. Может существовать целая система сертификатов, подтверждающих подлинность открытого ключа $K_{ЦРК}$.

6. Отправитель запрашивает у ЦРК асимметричный открытый ключ получателя сообщения. Этот процесс уязвим к атаке, в ходе которой атакующий вмешивается во взаимодействие между отправителем и получателем и может модифицировать трафик, передаваемый между ними. Поэтому открытый асимметричный ключ получателя «подписывается» ЦРК. Это означает, что $K_{ЦРК}$ использовал свой асимметричный секретный ключ для шифрования асимметричного открытого ключа получателя. Только ЦРК знает асимметричный секретный ключ $K_{ЦРК}$, поэтому есть гарантия того, что открытый асимметричный ключ получателя получен именно от ЦРК.

7. После получения асимметричный открытый ключ получателя расшифровывается с помощью асимметричного открытого ключа $K_{ЦРК}$ и алгоритма асимметричного шифрования/расшифрования.

8. Затем отправитель шифрует сеансовый ключ с использованием асимметричного алгоритма шифрования/расшифрования и асимметричного ключа получателя (полученного от ЦРК и расшифрованного).

9. Зашифрованный сеансовый ключ присоединяется к зашифрованному тексту, который включает в себя добавленную ранее электронную подпись.

10. Сформированный электронный цифровой «конверт» отправляется получателю.

11. Получатель выделяет зашифрованный сеансовый ключ из присланного цифрового «конверта» и решает проблему с расшифровкой сеансового ключа.

12. Используя свой секретный асимметричный ключ и тот же самый асимметричный алгоритм шифрования, получатель расшифровывает сеансовый ключ.

13. Получатель применяет к полученному зашифрованному тексту расшифрованный симметричный (сеансовый) ключ и тот же самый симметричный алгоритм шифрования/расшифрования и получает исходный текст вместе с электронной подписью.

14. Получатель отделяет электронную подпись от исходного текста.

15. Получатель запрашивает у ЦРК асимметричный открытый ключ отправителя.

16. После получения ключа получатель расшифровывает его с помощью открытого ключа $K_{ЦРК}$ и соответствующего асимметричного алгоритма шифрования/расшифрования.

17. Затем расшифровывается хэш-функция текста с использованием открытого ключа отправителя и асимметричного алгоритма шифрования/расшифрования.

18. Повторно вычисляется хэш-функция полученного исходного текста. Две эти хэш-функции сравниваются для проверки того, что исходный текст не был изменён.

Второй способ безопасного распространения секретных ключей основан на применении алгоритма открытого распределения ключей Диффи-Хеллмана. Этот алгоритм позволяет пользователям обмениваться ключами по незащищённым каналам связи.

7.8. Протокол алгоритма открытого распределения ключей Диффи-Хеллмана

Данный алгоритм [22] базируется на тех же принципах, что и асимметричная система шифрования с открытым ключом. Рассмотрим алгоритм выработки общего секретного ключа для организации незащищенного коммуникационного канала с использованием алгоритма открытого распределения ключей. С этой целью пользователям A и B необходимо:

1. Выбрать модуль N (число N должно быть простым) и примитивный элемент $g \notin Z_N$ ($1 \le g \le N-1$), который образует все ненулевые элементы множества Z_N, то есть ($g, g^2, \ldots, g^{N-1}$). Два целых числа Z и g могут не храниться в секрете. Как правило, эти значения являются общими для всех пользователей системы.

2. Выбрать собственные секретные ключи k_A и k_B (k_A и k_B – случайные большие целые числа, которые хранятся пользователями в секрете).

3. Вычислить собственные открытые ключи по следующим соотношениям:

$$K_A = g^{k_A} (mod\ N),$$

$$K_B = g^{k_B} (mod\ N).$$

4. Обменяться вычисленными значениями открытых ключей K_A и K_B по незащищённому каналу.

5. Вычислить общий секретный ключ применяя следующие соотношения:

- **пользователь A:** $K = (K_B)^{k_A} \equiv (g^{k_B})^{k_A} (mod\ N)$;
- **пользователь B:** $K' = (K_A)^{k_B} \equiv (g^{k_A})^{k_B} (mod\ N)$.

При этом $K = K'$, так как $(g^{k_B})^{k_A} (mod\ N) = (g^{k_A})^{k_B} (mod\ N)$.

Схема реализации алгоритма Диффи-Хеллмана приведена на рис. 7.2. В данном алгоритме ключ K может быть использован не только в качестве общего разделяемого секретного ключа (или ключа

шифрования ключей) в симметричной криптосистеме, но и как ключ для зашифрования, используя следующие преобразования: $C=E_k(M)=M^k (mod\ N)$.

При этом для расшифрования криптограммы получателю необходимо:

1) вычислить ключ расшифрования K^* с помощью сравнения

$K \cdot K^* \equiv 1 (mod\ N\text{-}1)$,

2) восстановить исходное сообщение M с помощью следующего преобразования:

$$M = D_k(C) = C^{k^*} (mod\ N).$$

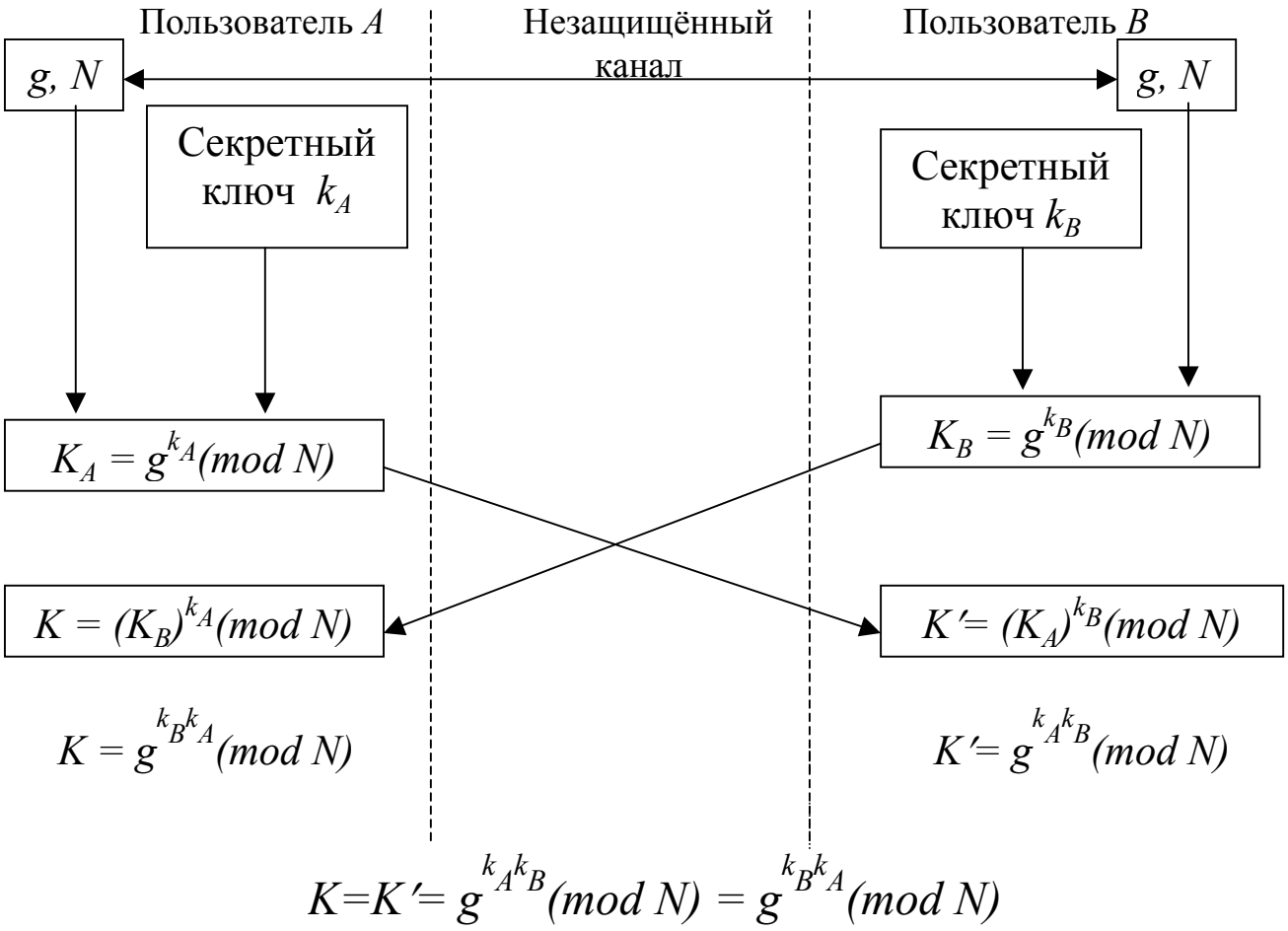

Рис. 7.2. Схема реализации алгоритма Диффи-Хеллмана

Пример 6 **[22]. Допустим, модуль N = 47, g = 23. Предположим, что пользователи A и B выбрали свои секретные ключи: $k_A = 12$, $k_B = 33$.**

Для того чтобы определить общий секретный ключ K, пользователи на первом этапе вычисляют значения частных открытых ключей:

$$K_A = g^{k_A} \pmod N = 23^{12} \pmod{47} = 27 \pmod{47},\ K_A = 27,$$

$$K_B = g^{k_B} \pmod N = 23^{33} \pmod{47} = 33 \pmod{47},\ K_B = 33.$$

На втором этапе пользователи A и B обмениваются своими открытыми ключами K_A и K_B и вычисляют общий секретный ключ:

$$K = (K_B)^{k_A} \pmod N = (K_A)^{k_B} \pmod N,$$

$$K = 33^{12} \pmod{47} = 27^{33} \pmod{47} = 25 \pmod{47}, \quad K = 25.$$

Вычисленный ключ K может использоваться в качестве общего секретного ключа для зашифрования сообщения M.

Для расшифрования криптограммы пользователи A и B вычисляют секретный ключ, используя сравнение

$$K \cdot K^* \equiv 1 \pmod{N\text{-}1};$$

$$25 \cdot K^* \equiv 1 \pmod{46} = 25 \cdot 35 \equiv 1 \pmod{46},\ \text{откуда}\ K^* = 35.$$

Таким образом, используя вычисленные ключи можно не только зашифровать сообщение M (например, M=16) в криптограмму C

$$C = M^k = 16^{25} = 21 \pmod{47},\ C = 21,$$

но и расшифровать его, используя преобразование

$$M = C^{k^*} = 21^{35} = 16 \pmod{47}.$$

Таким образом, расшифрованное сообщение соответствует исходному (M = 16).

Преимущество метода Диффи-Хеллмана по сравнению с методом *RSA* заключается в том, что формирование общего секретного ключа происходит в сотни раз быстрее.

Алгоритм Диффи-Хеллмана даёт возможность шифровать данные при каждом сеансе связи на новых ключах, что позволяет не хранить секреты на дискетах или других носителях, так как любое хранение секретов повышает вероятность попадания их в руки конкурентов или противника.

7.9. Алгоритм комплексной защиты конфиденциальности передаваемых данных на основе алгоритма Диффи-Хеллмана

Алгоритм Диффи-Хеллмана [25] позволяет реализовать комплексную защиту конфиденциальности и аутентичности передаваемых данных. Он предоставляет пользователю возможность сформировать и использовать одни и те же ключи для постановки цифровой подписи и шифрования передаваемых данных с помощью общего секретного ключа.

Этот алгоритм позволяет пользователям *A* и *B* сначала генерировать свои секретные ключи k_A и k_B и вычислить свои открытые ключи K_A и K_B. Затем абоненты *A* и *B* вычисляют общий разделяемый секретный ключ *K*, который в дальнейшем может использоваться для симметричного шифрования данных. Эти результаты работы алгоритма Диффи-Хеллмана могут быть использованы в качестве промежуточных данных для реализации метода комплексной защиты целостности и конфиденциальности передаваемой информации, суть которого заключается в следующем:

1. Абонент *A* подписывает сообщение *M* с помощью своего секретного ключа k_A, например по алгоритму цифровой подписи ГОСТ Р 34.10-94.

2. Абонент *A* вычисляет совместно разделяемый секретный ключ *K* по алгоритму Диффи-Хеллмана из своего секретного ключа k_A и открытого ключа K_B абонента *B*.

3. Абонент *A* зашифровывает сообщение *M* на полученном совместном разделяемом ключе *K*, например по алгоритму шифрования ГОСТ 28147-89.

4. Абонент *B* при получении зашифрованного сообщения *M* вычисляет по алгоритму Диффи-Хелмана совместно разделяемый секретный ключ *K* из своего секретного ключа k_B и открытого ключа K_A абонента *A*.

5. Абонент *B* расшифровывает полученное сообщение *M* на ключе *K*.

6. Абонент *B* проверяет подпись расшифрованного сообщения *M* с помощью открытого ключа абонента K_A.

На основе алгоритма Диффи-Хеллмана функционируют протоколы управления криптоключами *SKIP* и *IKE*, применяемые при построении защищённых виртуальных сетей *VPN* на сетевом уровне.

Важным достоинством алгоритма Диффи-Хеллмана является то, что он позволяет обойтись без защищённого канала передачи ключей. Проблема гарантии того, что пользователь *A* получил открытый ключ именно от пользователя *B*, и наоборот, решается с помощью сертификатов открытых ключей, создаваемых и распространяемых центрами сертификации ЦРК в рамках инфраструктуры управления открытыми ключами *PRI*.

7.10. Протокол вычисления общего секретного ключа на базе эллиптической кривой *ECKEP*

Протокол *ECKEP* (Elliptic Curve Key Establishment Protocol) предназначен для организации защищённого коммуникационного канала [15,22,23]. Предположим, что для конфиденциального информационного обмена пользователи A и B должны обменяться общим секретным ключом по незащищенным каналам связи. При этом пользователи должны иметь свои секретный и открытый ключи. Пользователь A имеет секретный ключ K_{cA} и открытый ключ $K_{oA} = K_{cA}\ P = (x_A, y_A)$. Аналогично пользователь B имеет секретный ключ K_{cB} и открытый ключ $K_{oB} = K_{cB}\ P = (x_B, y_B)$, где P выбранная точка эллиптической кривой.

Вычисление ключа парной связи проводится в четыре этапа:

1. Действия пользователя *A*:

- выбирает случайное целое число k_A, $1 \le k_A \le n - 1$;
- вычисляет $R_A = k_A P$;
- вычисляет $(x_1, y_1) = k_A\ K_{oB}$;
- вычисляет $s_A = k_A + K_{cA}\ x_A\ x_1 (mod\ n)$;
- R_A отправляется пользователю B.

2. Действия пользователя B:

- выбирает случайное целое число k_B, $1 \le k_B \le n - 1$;

- вычисляет $R_B = k_B P$;

- вычисляет $(x_2, y_2) = k_B K_{oA}$;

- вычисляет $s_B = k_B + K_{cB} x_B x_2 (\ mod\ n)$;

- R_B отправляет пользователю A.

3. Действия пользователя *A*:

- вычисляет $(x_2, y_2) = K_{cA} R_B$;

- вычисляет ключ парной связи $K = s_A (R_B + x_B x_2 K_{oB})$.

4. Действия пользователя *B*:

- вычисляет $(x_1, y_1) = K_{cB} R_A$;

- вычисляет ключ парной связи $K = s_B (R_A + x_A x_1 K_{oA})$, что эквивалентно значению $s_A (R_B + x_B x_2 K_{oB})$.

ПРИЛОЖЕНИЯ

ЭЛЕМЕНТЫ ТЕОРИИ ЧИСЕЛ

Как известно [4, 9], целые числа по модулю n с использованием операций сложения и умножения образуют коммутативное кольцо при соблюдении требований ассоциативности, коммутативности и дистрибутивности.

Поскольку приведение по модулю n является гомоморфным отображением из кольца целых в кольцо целых по модулю n, то можно либо сначала приводить по модулю n, а затем выполнять операции, либо сначала выполнять операции, а затем приводить по модулю n:

$$(a \pm b)\ mod\ n = [a(mod\ n) \pm b(mod\ n)]\ mod\ n,$$

$$(a \cdot b)\ mod\ n = [a(mod\ n) \cdot b(mod\ n)]\ mod\ n,$$

$$[a \cdot (b + c)]\ mod\ n = \{[a \cdot b(mod\ n)] + [a \cdot c(mod\ n)]\}\ mod\ n.$$

В криптографии [1, 22, 23] используется множество вычислений по модулю n, например вычисления дискретных логарифмов и квадратных корней. Это связано с тем, что с вычислениями по модулю удобнее работать, потому что они ограничивают диапазон всех вычислений промежуточных величин и результата.

Например, промежуточные результаты операции сложения, вычитания или умножения по модулю n длиной k бит будут не длиннее $2k$ бит, а вычисление степени числа a по модулю n

$$a^x\ mod\ n$$

можно выполнить как ряд умножений и делений. В силу дистрибутивности этих операций удобнее и быстрее произвести возведение в степень как ряд последовательных умножений, выполняя каждый раз приведение по модулю.

Так, например [17], если нужно вычислить $a^{16}\ mod\ n$, не следует применять примитивный подход с выполнением 15 перемножений и одного приведения по модулю большого числа:

$(a_1 \cdot a_2 \cdot \ldots \cdot a_{16}) \bmod n$.

Вместо этого следует выполнить только четыре малых умножения и четыре приведения по модулю:

$a^{16} \bmod n = (((a^2 \bmod n)^2 \bmod n)^2 \bmod n)^2 \bmod n$.

В свою очередь, вычисление $a^x \bmod n$, **где** x **не является степенью 2, производят аналогичным образом. Для этого число** x **представляют как сумму степеней 2:**

$x = 25_{(10)} \rightarrow 1\ 1\ 0\ 0\ 1_{(2)}$, **поэтому** $25 = 2^4 + 2^3 + 2^0$.

Затем вычисляют рассмотренным выше способом:

$a^{25} \bmod n = (a \cdot a^{24}) \bmod n = (a \cdot a^8 \cdot a^{16}) \bmod n =$

$= a \cdot ((a^2)^2)^2 \cdot (((a^2)^2)^2)^2 \bmod n = ((((a^2 \cdot a)^2)^2)^2 \cdot a) \bmod n$.

При сохранении промежуточных результатов вычислений в данном случае потребуется только шесть умножений:

$(((((((a^2 \bmod n) \cdot a) \bmod n)^2 \bmod n)^2 \bmod n)^2 \bmod n) \cdot a) \bmod n$.

Этот метод уменьшает трудоёмкость вычислений до $1{,}5k$ **операций в среднем, где** k **– длина числа в битах, что особенно заметно при работе с числами длинной более 100 бит.**

Так как многие алгоритмы шифрования основаны на возведении в степень по модулю n, **то целесообразно использовать рассмотренный выше метод быстрого возведения в степень.**

П.1. Алгоритм вычисления наибольшего общего делителя (алгоритм Евклида)

Целое число a **делит без остатка целое число** b, **если** $b = k \cdot a$ **для некоторого целого числа** k. **В этом случае число** a **называют делителем числа** b.

Пусть a **– целое число, большее 1. Тогда** a **является простым числом, если его положительными делителями будут 1 и само** a, **в противном случае** a **называется составным.**

Любое целое $n > 1$ **может быть представлено единственным образом с точностью до порядка сомножителей как произведение простых [1, 15].**

Существенным с точки зрения криптографии является тот факт, что до настоящего времени не известно как эффективных алгоритмов

разложения чисел на множители, так и методов восстановления двух простых чисел p и q из их произведения n:

$n = p \cdot q$.

Наибольший общий делитель чисел a и b, обозначаемый как НОД (a, b) или просто (a, b), – это наибольшее число, делящее одновременно числа a и b. В эквивалентной форме (a, b) – это натуральное число, которое делит a и b, и делится на любое число, делящее и a, и b . Если НОД (a, b) = 1, то целые a и b - взаимно простые.

Наибольший общий делитель может быть вычислен с помощью алгоритма Евклида.

Если принять следующие обозначения: q_i - частное; r_i - остаток, то данный алгоритм можно представить в виде цепочки равенств:

$a = b \cdot q_1 + r_1, \quad 0 < r_1 < b,$

$b = r_1 \cdot q_2 + r_2, \quad 0 < r_2 < r_1,$

$r_1 = r_2 \cdot q_3 + r_3, \quad 0 < r_3 < r_2,$

$\vdots \qquad\qquad \vdots$

$r_{k-2} = r_{k-1} \cdot q_k + r_k, \quad 0 < r_k < r_{k-1},$

$r_{k-1} = r_k \cdot q_{k+1}.$

Как видно из алгоритма остатки r_i от делений образуют строго убывающую последовательность натуральных чисел. Откуда следует, что r_k есть общий делитель чисел a и b и, более того, любой общий делитель чисел a и b делит и r_k.

Пример I **[14,17]. Найти НОД (175, 77) согласно алгоритма Евклида**

$(a, b) = (b, r_1) = (r_1, r_2) = \cdots = (r_{n-1}, r_n) = r_n$.

$175 = 77 \cdot 2 + 21, \quad 77 = 21 \cdot 3 + 14, \quad 21 = 14 \cdot 1 + 7, \quad 14 = 7 \cdot 2.$

Таким образом, последний положительный остаток – $r_3 = 7$, т.е. НОД (175, 77) = 7.

П.2. Алгоритмы вычисления обратных величин

В арифметике действительных чисел нетрудно вычислить мультипликативную обратную величину a^{-1} для $a \neq 0$:

$a^{-1} = 1/a$ **или** $a \cdot a^{-1} = 1.$

Например, мультипликативная обратная величина от числа 7 равна 1 / 7, поскольку $7 \cdot 1/7 = 1$.

В модулярной арифметике [22,23] вычисление обратной величины является более сложной задачей. Например, решение сравнения

$$4 \cdot x \equiv 1 \ (mod\ 7)$$

эквивалентно нахождению таких значений x и k, что

$$4 \cdot x \equiv 7 \cdot k + 1,$$

где x и k – целые числа.

Таким образом, основной задачей данного алгоритма является нахождение такого целого числа x, что

$$a \cdot x \ (mod\ n) = 1 \quad \text{или} \quad a^{-1} \equiv x \ (mod\ n).$$

Решение этой задачи не всегда существует. Например, обратная величина для числа 5 по модулю 14 равна 3, так как

$$5 \cdot 3 = 15 \equiv 1 \ (mod\ 14).$$

В то же время, число 4 не имеет обратной величины по *mod* 14, так как числа 4 и 14 не являются взаимно простыми.

Таким образом, сравнение $a^{-1} \equiv x \ (mod\ n)$ имеет единственное решение, если a и n - взаимно простые числа.

Если числа a и n не являются взаимно простым, то сравнение $a^{-1} \equiv x \ (mod\ n)$ не имеет решения.

Рассмотрим основные методы нахождения обратных величин. Пусть целое число $a \in \{0, 1, 2,\ldots, n\text{-}1\}$. Если НОД $(a, n) = 1$, то $ax_i \ (mod\ n)$ при $x_i = 0, 1, 2,\ldots, n\text{-}1$ является перестановкой множества $(0, 1, 2,\ldots,n\text{-}1)$.

Например, если $a = 3$ и $n = 7$, (НОД $(3, 7) = 1$), то

$$3 \cdot x_i \ (mod\ 7) \quad \text{при} \quad x_i = 0, 1, 2, \ldots,6$$

является последовательностью 0, 3, 6, 2, 5, 1, 4, то есть перестановкой множества {0, 1, 2, …, 6}.

Это становится неверным, когда НОД $(a, b) \neq 1$. Например, если $a = 2$ и $n = 6$, то

$$2 \cdot x_i (mod\ 6) \equiv 0, 2, 4, 0, 2, 4 \quad \text{при} \quad x_i = 0, 1, 2,\ldots, 5.$$

Если НОД $(a, b) = 1$, то только тогда существует обратное число a^{-1}, $0 < a^{-1} < n$, такое что $a \cdot a^{-1} \equiv 1 \ (mod\ n)$.

Действительно, если $a \cdot x_i \ (mod\ n)$ является перестановкой чисел 0, 1, …, n - 1, то существует такое x_i, что $a \cdot x_i \equiv 1 \ (mod\ n)$.

Как известно, набор целых чисел от 0 до n - 1 называют полным набором вычетов по модулю n, поэтому для любого целого числа $a>0$ его вычет $r = a\ (mod\ n)$ - это некоторое целое число в интервале от 0 до n - 1.

Выделим из полного набора вычетов подмножество вычетов, взаимно простых с n . Такое подмножество называют приведенным набором вычетов.

Пример II **[17]. Пусть модуль n = 10, тогда полный набор вычетов по модулю (n - 1) равен {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}.**

Из них только 1, 3, 7, 9 не имеют общего сомножителя с числом 10. Поэтому приведенный набор вычетов по модулю 10 равен (1, 3, 7, 9). При формировании этого приведенного набора были удалены 6 элементов (0 и элементы кратные 2 и 5).

Для произведения простых чисел $p \cdot q$ приведенный набор вычетов имеет $(p - 1)(q - 1)$ элементов.

Произведение $(p - 1)(q - 1) = \varphi(n)$ называется функцией Эйлера.

При $n = p \cdot q = 2 \cdot 5 = 10$ число элементов в приведенном наборе

$\varphi(n) = (p - 1)(q - 1) = (2 - 1)(5 - 1) = 4$.

Пусть n =27, тогда приведенный набор вычетов по модулю $27 = 3^3$ имеет 18 элементов:

{1, 2, 4, 5, 7, 8, 10, 11, 13, 14, 16, 17, 19, 20, 22, 23, 25, 26}.

Из полного набора вычетов исключены элементы, кратные 3. Для модуля в виде простой степени n^r приведенный набор вычетов определяется из выражения

$\varphi(n) = n^{r-1}(n - 1)$.

Например, для $n = 3$ и $r = 3$ получаем $3^{3-1}(3-1) = 3^2 \cdot 2 = 18$.

Подводя итог изложенному можно сделать вывод, что функция Эйлера $\varphi(n)$ определяет число элементов в приведенном наборе вычетов (табл. П.1)

Иначе говоря, функция $\varphi(n)$ – это количество положительных целых, меньших n , которые взаимно просты с n. Согласно малой теореме Ферма: если n - простое и НОД $(a, n) = 1$, то

$a^{n-1} \equiv 1\ (mod\ n)$ или $a^{\varphi(n)} \equiv 1\ (mod\ n)$.

Рассмотрим три метода нахождения обратной величины вида

$a^{-1} \equiv 1\ (mod\ n)$.

1. Определение обратной величины методом поочерёдной проверки значений 1, 2,… , n - 1, пока не будет найден

$x = a^{-1} (mod\ n)$ **такой, что** $a \cdot x \equiv 1\ (mod\ n)$.

Пример III **[11]: дано** $n = 7$, $a = 5$.

Необходимо найти $x = a^{-1} (mod\ n)$.

$a \cdot x \equiv 1 (mod\ n)$ **или** $5 \cdot x \equiv 1 (mod\ 7)$.

Получаем $x = 5^{-1}\ (mod\ 7) = 3$.

2. Определение $a^{-1}\ (mod\ n) \equiv a^{\varphi(n)-1}\ (mod\ n)$ **(если известна функция Эйлера** $\varphi(n)$**), используя алгоритм быстрого возведения в степень.**

Таблица П.1

Модуль n	**Функция** $\varphi(n)$
n **- простое**	n **- 1**
n^2	$n(n-1)$
. . .	. . .
n^r	$n^{r-1}(n-1)$
$p \cdot q$ (p, q **– простые**)	$(p-1)(q-1)$
. . .	. . .
p_i^{ei} **(- простые)**	$p_i^{\ e_i-1}(p_i - 1)$

Пример IV **[11]: дано** $n = 7$, $a = 5$. **Найти** $x = a^{-1} (mod\ n) = 5^{-1} (mod\ 7)$. **Модуль** $n = 7$ **– простое число. Поэтому функция Эйлера** $\varphi(n) = \varphi(7) = n - 1 = 6$. **Обратная величина от 5 по** *mod* 7

$a^{-1} (mod\ n) = a^{\varphi(n)-1} (mod\ n) =$

$= 5^{6-1} mod\ 7 = 5^5 mod\ 7 = (5^2\ mod\ 7)(5^3 mod\ 7)\ mod\ 7 =$

$= (25\ mod\ 7)(125\ mod\ 7)\ mod\ 7 = (4 \cdot 6)\ mod\ 7 = 24\ mod\ 7 = 3.$

Итак, $x = 5^{-1} (mod\ 7) = 3$.

3. Определение обратной величины $a^{-1} (mod\ n)$ **с помощью расширенного алгоритма Евклида (РАЕ). РАЕ при заданных неотрицательных целых числах** a **и** b **позволяет вычислить целые числа** u_1, u_2 **и** u_3 **такие, что** $a \cdot u_1 + b \cdot u_2 = u_3 =$ **НОД** (a, b).

В процессе вычисления используются вспомогательные векторы (v_1, v_2, v_3), (t_1, t_2, t_3). **Действия с векторами производятся таким образом, что в течение всего процесса вычисления выполняются соотношения**

$a \cdot t_1 + b \cdot t_2 = t_3$, $a \cdot u_1 + b \cdot u_2 = u_3$, $a \cdot v_1 + b \cdot v_2 = v_3$.

Если использовать частный режим работы РАЕ, при котором $b = n$, НОД $(a, n) = 1$ и $u_3 = 1$, то обратная величина $a^{-1}(mod\ n)$ определяется из следующего сравнения:

$a \cdot u_1 + n \cdot u_2 =$ НОД $(a, n) = 1$,

$(a \cdot u_1 + n \cdot u_2)\ mod\ n \equiv a \cdot u_1(mod\ n) \equiv 1$,

$a^{-1}(mod\ n) \equiv u_1(mod\ n)$.

Для решения более сложных сравнений $a \cdot x \equiv b\ (mod\ n)$, $b \neq 1$ используется следующий алгоритм. Сначала решают сравнение $a \cdot y \equiv 1\ (mod\ n)$, т.е. определяют $y = a^{-1}(mod\ n)$, а затем находят

$x = a^{-1} \cdot b\ (mod\ n) = y \cdot b\ (mod\ n)$.

Пример V **[6, 14]. Найти x для сравнения $5 \cdot x \equiv 9(mod\ 23)$.**

Сначала решаем сравнение $5 \cdot x \equiv 1(mod\ 23)$.

Получаем $y = 5^{-1}(mod\ 23) = 14$.

Затем находим

$x = 5^{-1} \cdot 9(mod\ 23) = 14 \cdot 9(mod\ 23) = 126(mod\ 23) = 11(mod\ 23)$, **$x = 11$**

П.3. Китайская теорема об остатках

Китайская теорема об остатках [1,22] формулируется следующим образом.

Если $m_1, m_2, \ldots, m_t$ – модули, которые являются попарно взаимно простыми целыми числами, то есть. НОД $(m_i, m_j) = 1$ при $i \neq j$, то сравнения $x \equiv a_i(mod\ m_i)$, $i = 1, 2, \ldots, t$ имеют в интервале $[0, M-1]$ единственное общее решение

$$x = \sum_{i=1}^{t} a_i \cdot N_i \cdot M_i \left(\mathrm{mod}\, M\right),$$

где $M = m_1 \cdot m_2 \cdot \ldots \cdot m_t$ – произведение всех m; N_i - обратный элемент к $M_i\ (mod\ m_i)$, $i = 1, 2, \ldots, t$, то есть $M_i \cdot N_i \equiv 1(mod\ m_i)$; $a_1, a_2, \ldots, a_t$ –целые числа , $0 \leq a_i \leq m_i$; $M_i = M/m_i$.

Так как НОД $(M_i, m_i) = 1$, то обратный элемент N_i существует и легко находится с помощью алгоритма Евклида из соотношения

$M_i \cdot N_i + m_i \cdot n_i = 1, \quad i = 1, 2, \ldots, t.$

Рассмотрим частный случай. Пусть $M = m_1 \cdot m_2$, **где** m_1, m_2 **– взаимно простые числа. Тогда для произвольных целых** $a_1 < m_1$ **и** $a_2 < m_2$ **существует единственное число** $x < M$ **такое, что**

$N_1 \cdot M_1 \equiv 1 (mod\ m_1)$ **и** $N_2 \cdot M_2 \equiv 1\ (mod\ m_2)$.

Здесь $M_1 = M/m_1 = (m_1 \cdot m_2)/m_1 = m_2$; $M_2 = M/m_2 = m_1$.

Значение x **вычисляют из соотношения**

$x = (a_1 \cdot N_1 \cdot M_1 + a_2 \cdot N_2 \cdot M_2)(mod\ M)$.

Пример VI **[22]: решить систему из двух сравнений**

$x \equiv 1 (mod\ 5), \qquad x \equiv 10 (mod\ 11)$

и найти общее решение x **по модулю 55.**

Здесь $m_1 = 5$; $M = m_1 \cdot m_2 = 5 \cdot 11 = 55$; $a_1 = 1$; $a_2 = 10$;

$M_1 = M/m_1 = m_2 = 11$; $M_2 = M/m_2 = m_1 = 5$.

Найдём значения N_1 **и** N_2, **обратные к** M_1 **и** M_2 **соответственно по** $mod\ m_1$ **и** $mod\ m_2$:

$M_1 \cdot N_1 \equiv 1 (mod\ m_1), \quad 11 \cdot N_1 \equiv 1 (mod\ 5) \Rightarrow N_1 = 1,$

$M_2 \cdot N_2 \equiv 1\ (mod\ m_2), \quad 5 \cdot N_2 \equiv 1 (mod\ 11) \Rightarrow 9.$

Вычислим общее значение

$x = (a_1 M_1 N_1 + a_2 M_2 N_2)(mod\ M) = (1 \cdot 11 \cdot 1 + 10 \cdot 5 \cdot 9)(mod\ 55) = (11 + 450)(mod\ 55) = 461 (mod\ 55) = 21\ (mod\ 55)$.

Итак, $x = 21 (mod\ 55), \quad x = 21.$

П.4. Вычисления в конечных полях Галуа

В поле Галуа [18, 19] определены операции сложения, вычитания, умножения и деления на ненулевые элементы. Существует единичный элемент сложения - 0, и умножения - I. Для каждого ненулевого числа существует единственное обратное число. Выполняются коммутативный, ассоциативный и дистрибутивный законы.

Конечное поле $F(p)$ с конечным числом p элементов играет важную роль в криптографии. В общем случае число элементов определяется как $p = q^n$, где q – некоторое простое число и $n \geq 1$.

Такие конечные поля называют полями Галуа и обозначают $GF(q^n)$ или $GF(q)$ при $n = 1$.

Вычисления в полях Галуа широко используются в криптографии. В нем работает вся теория чисел, в поле содержатся числа только конечного размера, при делении отсутствуют ошибки округления. В поле Галуа определены четыре алгебраические операции. При этом операции сложения и вычитания выполняются по модулю 2. Операция умножения элементов поля выполняется как умножение по модулю неприводимого многочлена $p(x)$ (то есть многочлена, по модулю которого построены элементы поля $GF(2^n)$).

Чтобы выполнить деление элемента b на элемент a в поле $GF(2^n)$ по модулю $p(x)$, сначала находят обратный элемент $a^{-1}(mod\ p(x))$, а затем вычисляют $b\ a^{-1}\ (mod\ p(x))$.

Каждый двоичный вектор длины n, исключая 0, является взаимно простым с неприводимым многочленом $p(x)$ независимо от значения $p(x)$. Поэтому число вычетов взаимно простых с $p(x)$ равно $\varphi(p(x)) = 2^n - 1$. Поэтому $a^{-1} = a^{\varphi(p(x))-1} mod\ p(x) = a^{2n-2}\ mod\ p(x)$.

В настоящее время многие криптосистемы основываются на полях Галуа, основанных на арифметике по модулю неприводимых многочленов степени n, чьи коэффициенты – целые числа по модулю q, где q – простое. Эти поля Галуа обозначают как $GF(q^n)$. Они имеют элементы, которые описываются многочленами степени не выше $(n - 1)$ в форме

$$a(x) = a_{n-1}x^{n-1} + \ldots + a_1x + a_0.$$

Каждый элемент $a(x)$ является вычетом по модулю $p(x)$, где $p(x)$ – неприводимый многочлен степени n (то есть $p(x)$ нельзя разложить на сомножители – многочлены степени меньше n).

Многочлен называют неприводимым, если его нельзя представить как произведение двух других многочленов (конечно, кроме 1 и самого многочлена). Многочлен $x^2 + 1$ неприводим над целыми

числами, а многочлен $x^3 + 2x^2 + x$ приводим, поскольку может быть представлен как $x\,(x + 1)(x + 1)$.

Многочлен, который в данном поле является образующей, называют примитивным, все его коэффициенты взаимно просты.

Арифметические действия над коэффициентами a_i выполняются по модулю q, а наивысшая степень x равна $(n - 1)$, так как выполняется приведение по модулю многочлена $p(x)$, имеющего степень n.

Особый интерес представляют поля $GF(2^n)$, где коэффициентами a_i являются 0 и 1. Поэтому многочлен $a(x)$ степени не выше $(n - 1)$ можно представить как вектор из n двоичных цифр: $a_{n-1}a_{n-2}\ldots a_1a_0$.

Каждый из n – битовых векторов соответствует конкретному элементу поля $GF(2^n)$.

Например, $GF(2^3)$ включает следующие элементы: 0, 1, x, $x + 1$, x^2, $x^2 + 1$, $x^2 + x$, $x^2 + x + 1$, которым соответствует двоичная форма: 000, 001,010, 011, 100, 101, 110, 111.

Вычисления в полях Галуа предполагают знания следующих основных свойств многочленов:

1. Ненулевые элементы поля $GF(2^n)$ являются корнями обобщенного многочлена $x^{2^n-1} + 1$.

2. Каждый многочлен $p(x)$ степени n, неприводимый над полем $GF(2)$, является делителем двучлена $x^{2^n-1} + 1$, и каждый его делитель, неприводимый над полем $GF(2)$, имеет степень, равную n и менее.

3. Все элементы поля $GF(2^n)$ можно получить как совокупность остатков от деления 1000…00 на неприводимый многочлен $p(x)$, входящий в разложение двучлена $x^{2^n-1} + 1$. Эти остатки - корни двучлена $x^{2^n-1} + 1$, обращающие его в нуль.

Число остатков равно $2^n - 1$.

4. В поле $GF(2^n)$ существует примитивный элемент g такой, что каждый ненулевой элемент поля $GF(2^n)$ может быть представлен как некоторая степень g, т.е. мультипликативная группа $GF(2^n)$ является циклической.

Достоинства вычислений в поле $GF(2^n)$:

1. Все элементы поля Галуа имеют конечный размер, деление элементов не имеет каких-либо ошибок округления.

2. Сложение и вычитание элементов поля $GF(2^n)$ не требует деления на модуль.

3. Алгоритмы вычислений в поле $GF(2^n)$ допускают параллельную реализацию.

4. Для поля $GF(2^n)$ обычно применяют в качестве модуля трёхчлен $P(x^n) = x^n + x + 1$.

Кроме того, длинная строка нулей между коэффициентами при x^n и x обеспечивает более простую реализацию быстрого умножения (с приведением по модулю). Трёхчлен $P(x^n)$ должен быть неприводимым и примитивным.

Трёхчлен $P(x^n) = x^n + x + 1$ является примитивным для следующих значений n ($n < 1000$):

1, 3, 4, 6, 9, 15, 22, 28, 30, 46, 60, 63, 127, 153, 172, 303, 471, 532, 865, 900.

Вычисления в $GF(2^n)$ можно быстро реализовать аппаратно с помощью регистров сдвига с линейной обратной связью. По этой причине вычисления над $GF(2^n)$ часто выполняются быстрее, чем вычисления над $GF(p)$.

П.5. Образующие

Если p – простое число и g меньше, чем p, то g называется образующей по модулю p, если для каждого числа a от 1 до p - 1 существует такое число x, $g^x \equiv a \ (mod\ p)$.

Иногда g называют также примитивным корнем по модулю p. Например [22], если $p = 11$, то 2 – это образующая по модулю 11:

$2^{10} = 1024 \equiv 1(mod\ 11)$; $2^1 = 2 \equiv 2(mod\ 11)$; $2^8 = 256 \equiv 3(mod\ 11)$;

$2^2 = 4 \equiv 4(mod\ 11)$; $2^4 = 16 \equiv 5(mod\ 11)$; $2^9 = 512 \equiv 6(mod\ 11)$;

$2^7 = 128 \equiv 7(mod\ 11)$; $2^3 = 8 \equiv 8(mod\ 11)$; $2^6 = 64 \equiv 9(mod\ 11)$;

$2^5 = 32 \equiv 10(mod\ 11)$.

Каждое число от 1 до 10 может быть представлено как $2^x \pmod p$. Для значения $p = 11$ образующими являются числа 2, 6, 7 и 8. Другие числа не являются образующими. Например, образующей не является число 3, поскольку не существует решения $3^x = 2 \pmod{11}$. В общем случае, установить, является ли данное число образующей, нелегко. Однако задача упрощается, если известно разложение p - 1 на множители. Пусть $q_1, q_2, \ldots, q_n$ – это простые множители p - 1.

Чтобы проверить, является ли число g образующей по модулю p, необходимо вычислить: $g^{(p-1)/q} \bmod p$ для всех значений $q = q_1, q_2 \ldots, q_n$.

Если это число равно 1 для некоторого значения q, то g не является образующей. Если для всех значений q рассчитанное значение не равно 1, то g - образующая.

Пусть, например, $p = 11$. Простые множители p - 1 = 10 равны 2 и 5. Чтобы проверить, что число 2 - образующая, вычислим:

$$2^{(11-1)/2} \pmod{11} = 10, \quad 2^{(11-1)/5} \pmod{11} = 4$$

Ни один из ответов не равен 1, поэтому 2 - это образующая. Теперь проверим, является ли образующей число 3:

$$3^{(11-1)/2} \pmod{11} = 1, \quad 3^{(11-1)/5} \pmod{11} = 9.$$

Следовательно, 3 — это не образующая.

П.6. Квадратичные вычеты

Если a - квадратичный вычет, то сравнение $x^2 \equiv a \pmod p$ имеет два решения: $+x$ и $-x$, т.е. a имеет два квадратичных корня по модулю p. Все квадратичные вычеты [5,25] находят возведением в квадрат элементов 1, 2, 3, …, (p - 1)/2.

Не все значения $a < p$ являются квадратичными вычетами. Если a – квадратичный вычет по модулю p, то a имеет два квадратных корня: один корень между 0 и (p - 1)/2, другой корень между (p –1)/2 и (p - 1). Если n – произведение двух простых чисел p и q, т.е. существуют точно (p - 1)(q – 1)/4 квадратичных вычетов по модулю p,

взаимно простых с n, тогда если p - простое число и a больше 0, но меньше p, то a представляет собой квадратичный вычет по модулю p, если $x^2 = a \pmod p$.

Этому требованию соответствуют не все значения. Чтобы, a было квадратичным вычетом по модулю n, оно должно быть квадратичным вычетом по модулю всех простых делителей n. Например [22], если $p = 7$, квадратичные вычеты равны 1,2 и 4:

$1^2 = 1 \equiv 1 \pmod 7$, $2^2 = 4 \equiv 4 \pmod 7$, $3^2 = 9 \equiv 2 \pmod 7$,

$4^2 = 16 \equiv 2 \pmod 7$, $5^2 = 25 \equiv 4 \pmod 7$, $6^2 = 36 \equiv 1 \pmod 7$.

Следует обратить внимание на то, что в этом списке каждый квадратичный вычет появляется дважды. Таким образом, если a - квадратичный вычет по модулю p, то у a есть ровно два квадратных корня, причем значение одного из корней находится между 0 и $(p - 1)/2$, а второго - между $(p - 1)/2$ и $(p - 1)$. Один из этих квадратных корней одновременно является квадратичным вычетом по модулю p. Этот корень называют главным квадратным корнем. А вот в любом из следующих уравнений не существует значений x, удовлетворяющих этим уравнениям:

$$x^2 = 3 \pmod 7, \qquad x^2 = 5 \pmod 7, \qquad x^2 = 6 \pmod 7.$$

Числа 3, 5 и 6 являются квадратичными невычетами по модулю 7.

Если n - произведение двух простых чисел p и q, тогда существует ровно $(p - 1)(q - 1)/4$ квадратичных вычетов по модулю n. Квадратичный вычет по модулю n является полным квадратом по модулю n и, поскольку чтобы быть квадратом по модулю n, вычет должен быть квадратом по модулю p и квадратом по модулю q.

Например, по модулю 35 ($p = 5$, $q = 7$, $n = 5 \cdot 7 = 35$) существуют $[(5 - 1)(7 - 1)]/4 = (4 \cdot 6)/4 = 6$ квадратичных вычетов: 1, 4, 9, 11, 16, 29, взаимно простых с 35.

ЗАКЛЮЧЕНИЕ

Как известно [1, 22, 23], криптографические методы и средства защиты информации используются в вычислительных системах любой степени сложности и назначения. Владение основами криптографии становится важным для ученых и инженеров, специализирующихся в области разработки современных средств защиты, а также в областях эксплуатации и проектирования информационных и телекоммуникационных систем.

В настоящее время криптография располагает необходимыми алгоритмами и средствами, которые позволяют организовать систему защиты таким образом, что у противника с ограниченными финансовыми и техническими возможностями для расшифрования конфиденциальной информации остаются только две возможности такие, как человеческий фактор и использование ненадежных криптоалгоритмов в процессе зашифрования. При этом типичными ошибками пользователей, нарушающих безопасность всей криптосистемы, являются:

- предоставление своего секретного пароля коллегам по работе;
- повторное использование секретных паролей и ключей;
- генерация ключей и паролей самими пользователями и т.д.

К основным причинам ненадёжности криптосистем, связанных с особенностями их реализации, относятся:

- использование нестойких криптоалгоритмов;
- ошибки в реализации и неправильное их применение.

На основании изложенного выше можно заключить, что надёжная система защиты должна уметь не только обеспечивать высокую криптостойкость, но и оперативно обнаруживать несанкционированные действия противника для минимизации возможного ущерба.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Алферов А.П., Зубов А.Ю. и др. Основы криптографии: Учеб. пособие, 2-е и зд., испр. и доп. - М.: Гелиос АРВ, 2002.- 480 с., ил.

2. Вентцель Е.С., Овчаров Л.А. Теория случайных процессов и ее инженерные приложения. - Учеб. пособие для втузов 2-е изд., стер. - М.: Высш. шк., 2000.- 383 с.: ил.

3. Галицкий А.В., Рябко С.Д., Шаньгин В.Ф. Защита информации в сети – анализ технологий и синтез решений / Галицкий и др. – М.: ДМК Пресс, 2004.- 616 с.: ил.

4. Грэхем Р., Кнут Д., Паташник О. Конкретная математика. Основание информатики: Пер. с англ.- М.: Мир. 1998.- 703 с., ил.

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

6. Домашев А.В., Попов В.О., Правиков Д.И., Прокофьев И.В., Щербаков А.Ю. Программирование алгоритмов защиты информации: Учеб. пособие. - М.: «Нолидж». 2000.- 288 с.: ил.

7. Зенкин О.С., Иванов М.А. Стандарт криптографической защиты – AES. Конечные поля/Под ред. М.А. Иванова. – М.: КУДИЦ-ОБРАЗ, 2002. – 176 с.

8. Иванов М.А. Криптографические методы защиты информации в компьютерных системах и сетях. - М.: Кудиц-Образ, 2001.-368 с.

9. Коутинхо С. Введение в теорию чисел. Алгоритм RSA. - М.: Постмаркет, 2001.- 328 с.

10. Лукацкий А.В. Обнаружение атак. - СПб.: БХВ – Петербург, 2001.- 624 с.: ил.

11. Масленников М.Е. Практическая криптография. – СПб.: БХВ-Петербург, 2003.- 464 с.: ил.

12. Молдовян А.А., Молдовян Н.А., Советов Б.Я. Криптография. – Серия «Учебники для вузов. Специальная литература». – СПб.: Издательство «Лань», 2000. – 224 с.: ил.

13. Нечаев В.И. Элементы криптографии (Основы теории защиты информации): Учеб. пособие для ун-тов и пед. вузов/Под ред. В.А. Садовничего. – М.: Высш. шк., 1999.- 109 с.

14. Петраков А.В. Основы практической защиты информации. - М.: Радио и связь,1999. – 158 с.

15. Петров А.А. Компьютерная безопасность. Криптографические методы защиты. - М.: ДМК, 2000.- 448с.: ил.

16. Проскурин В.Г. Программно - аппаратные средства обеспечения информационной безопасности. Защита в операционных системах: Учеб. пособие для вузов / Проскурин В.Г., Крутов С.В., Мацкевич И.В.-М.: Радио и связь, 2000.- 168 с.: ил.

17. Романец Ю.В., Тимофеев П.А., Шаньгин В.Ф. Защита информации в компьютерных системах и сетях / Под ред. В.Ф. Шаньгина, 2-е изд., перераб. и доп. - М.: Радио и связь, 2001.- 376 с.: ил.

18. Ростовцев А.Г., Маховенко Е.Б. Введение в криптографию с открытым ключом. - СПб.: «Мир и Семья», 2001.- 336 с.: ил.

19. Ростовцев А.Г. Алгебраические основы криптографии. – СПб.: НПО «Мир и семья», ООО «Интерлайн», 2000. – 354 с.: ил.

20. Скляров Д.В. Искусство защиты и взлома информации. - СПб.: БХВ - Петербург, 2004.- 288 с.: ил.

21. Соколов А.В., Шаньгин В.Ф. Защита информации в распределенных корпоративных сетях и системах. - М.:ДМК Пресс, 2002.- 656 с.: ил.

22. Столлингс, Вильям. Криптография и защита сетей: принципы и практика, 2-е изд.: Пер. с англ.- М.: Издательский дом «Вильямс», 2001.- 672 с.: ил.

23. Харин Ю.С., Берник В.И., Матвеев Г.В., Агиевич С.В. Математические и компьютерные основы криптологии: Учеб. пособие. – Мн.: Новое знание, 2003.- 382 с.

24. Чмора А.Л. Современная прикладная криптография. – М.: Гелиос АРВ, 2001. –256 с.: ил.

25. Б. Шнайер. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. - М.: Издательство Триумф, 2002.- 816 с.: ил.

26. Ященко В.В. и др. Введение в криптографию/Под ред. В.В. Ященко. - СПб.: Питер, 2001.- 288 с.: ил.

27. ГОСТ 28147-89. Система обработки информации. Защита криптографическая. Алгоритм криптографического преобразования.

28. ГОСТ Р 34.11-94. Информационная технология. Криптографическая защита информации. Функция хэширования.

29. ГОСТ Р 34.10-94. Информационная технология. Криптографическая защита информации. Процедуры выработки и проверки электронной цифровой подписи на базе асимметричного криптографического алгоритма.

Предметный указатель

ОГЛАВЛЕНИЕ

Хамидуллин Рафкат Рахимжанович

Бригаднов Игорь Альбертович

Морозов Алексей Валентинович

Методы и средства защиты компьютерной информации

Учебное пособие

Редактор И.Н. Садчикова