Министерство образования
Республики Беларусь

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра «Информационные технологии в управлении»

В.Ф. Голиков

БЕЗОПАСНОСТЬ ИНФОРМАЦИИ И НАДЕЖНОСТЬ КОМПЬЮТЕРНЫХ СИСТЕМ

Методическое пособие

Ч а с т ь 2

Минск
БНТУ
2012

Министерство образования Республики Беларусь
БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра «Информационные технологии в управлении»

В.Ф. Голиков

БЕЗОПАСНОСТЬ ИНФОРМАЦИИ И НАДЕЖНОСТЬ КОМПЬЮТЕРНЫХ СИСТЕМ

Методическое пособие
для студентов специальностей
1-40 01 01 «Программное обеспечение информационных технологий»
и 1-53 01 02 «Автоматизированные системы обработки информации»
всех форм обучения

В 2 частях

Ч а с т ь 2

КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА ИНФОРМАЦИИ

Минск
БНТУ
2012

УДК 621.391.25 (076)
ББК 32.811 я 7
Г 60

Р е ц е н з е н т ы:
Л.М. Лыньков, А.А. Лобатый

Голиков, **В.Ф**.
Г 60 Безопасность информации и надежность компьютерных систем: методическое пособие для студентов специальностей 1-40 01 01 «Программное обеспечение информационных технологий» и 1-53 01 02 «Автоматизированные системы обработки информации» всех форм обучения: в 2 ч. / В.Ф. Голиков. – Минск: БНТУ, 2012. – Ч. 2: Криптографическая защита информации. – 91 с.

ISBN 978-985-525-723-4 (Ч. 2).

Пособие предназначено для студентов высших учебных заведений, обучающихся по специальностям 1-40 01 01 «Программное обеспечение информационных технологий» и 1-53 01 02 «Автоматизированные системы обработки информации», а также будет полезно для студентов других специальностей, изучающих информационные технологии.

В части 2 рассмотрены криптографические методы и средства защиты информации в компьютерных системах.

Часть 1 настоящего издания «Безопасность информации и надежность компьютерных систем», автор В.Ф. Голиков, вышла в свет в 2010 г. в БНТУ.

УДК 621.391.25 (076)
ББК 32.811 я 7

ISBN 978-985-525-723-4 (Ч. 2)
ISBN 978-985-525-301-4

ОГЛАВЛЕНИЕ

1. ОСНОВЫ ПОСТРОЕНИЯ КРИПТОСИСТЕМ

Криптография – это наука о методах, алгоритмах, программных и аппаратных средствах преобразования информации в целях сокрытия ее содержания, предотвращения видоизменения или несанкционированного использования.

Исторически сложилось так, что криптография длительное время использовалась исключительно как средство обеспечения конфиденциальности сообщений. Применялась эта наука в области защиты государственных тайн: в военной, дипломатической и разведывательной сферах. Именно поэтому криптография находилась в руках спецслужб. Всякие упоминания о науке в открытой печати были запрещены, несмотря на то, что огромное число специалистов трудилось в этой области: математики, инженеры, разведчики. Криптографическая империя СССР противостояла аналогичной империи США. Часть специалистов трудилась над созданием стойких криптоалгоритмов, другая часть – над раскрытием чужих криптосистем.

В конце XX века обстановка в сфере использования криптографии коренным образом изменилась. Здесь можно выделить несколько причин. Главная – бурное развитие вычислительной техники, появление на этой базе информационных технологий. Доступность информационных технологий широкому кругу коммерческих компаний и частным лицам породила потребность, во-первых, обеспечивать конфиденциальность той информации, которая циркулирует в компьютерных сетях; во-вторых, обеспечивать ряд функций, таких как аутентификация субъектов системы, целостность сообщений, истинность документов и т. д. Оказалось, что все это можно обеспечить, используя принципы криптографии.

Криптография, обслуживающая задачи управления, бизнеса, телекоммуникаций, получила название открытой. Открытые криптотехнологии (электронно-цифровая подпись, идентификация и аутентификация, защита от НСД) становятся коммерческими продуктами и распространяются без особых ограничений. Платежные системы: банковские, индивидуальные на основе пластиковых карт, локальные и корпоративные компьютерные сети – это далеко не полный перечень применения криптографических технологий.

Наряду с решением задач обеспечения конфиденциальности, целостности и доступности информации существует задача анализа

стойкости используемых криптопреобразований. Эта задача решается наукой, называемой *криптоанализ.* Криптография и криптоанализ составляют науку – криптологию.

1.1. Общие принципы криптографической защиты информации

Обобщенная схема криптографической системы, обеспечивающей шифрование передаваемой информации, представлена на рис. 1.1.

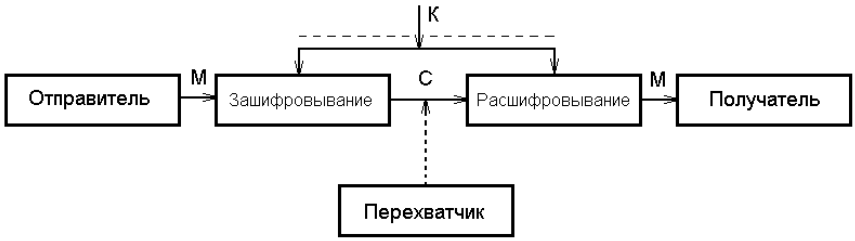

Рис. 1.1. Обобщенная схема симметричной криптографической системы с одним секретным ключом K

Отправитель генерирует открытый текст исходного сообщения M, которое должно передаваться по открытому каналу, шифрует текст с помощью обратимого преобразования E и ключа K: E_K и получает шифротекст $C = E_K(M)$, который отправляет получателю. Получатель, приняв шифротекст C, расшифровывает его с помощью обратного преобразования $D = E_K^{-1}$ и получает исходное сообщение в виде открытого текста $M: M = D(C) = E_K^{-1}(E_K(M))$.

Преобразование E_K выбирается из семейства криптографических преобразований, называемых *криптоалгоритмами*. Параметр, с помощью которого выбирается конкретное преобразование, называется *криптографическим ключом* K. Система, в которой осуществляется зашифровывание и расшифровывание сообщений, называется *криптосистемой.*

Формально криптосистема – это однопараметрическое семейство $(E_K)_{K \in \bar{K}}$ обратимых преобразований $E_K: \bar{M} \to \bar{C}$ из пространства $\bar{M}$ сообщений открытого текста в пространство $\bar{C}$ шифрованных

текстов. Параметр K (ключ) выбирается из конечного множества $\overline{K}$, называемого *пространством ключей*. Криптосистема может иметь разные варианты реализации: набор инструкций, аппаратные или программные средства, аппаратно-программные средства.

Преобразование зашифровывания может быть симметричным или асимметричным относительно преобразования расшифровывания. Поэтому различают два класса криптосистем: симметричные и асимметричные криптосистемы. Иногда их называют одноключевыми (с секретным ключом) и двухключевыми (с открытым ключом).

Обобщенная схема асимметричной криптосистемы с двумя разными ключами K_1 и K_2 показана на рис. 1.2.

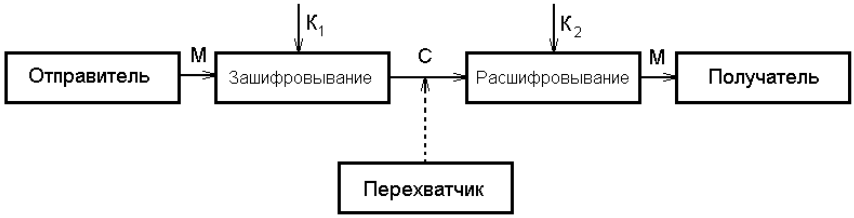

Рис. 1.2. Обобщенная схема асимметричной криптографической системы

В этой криптосистеме один из ключей является открытым K_1, а другой K_2 – секретным. Для ее $C = E_{K_1}(M)$, а $M = D_{K_2}(C) = E_{K_2}^{-1}(E_{K_1}(M))$.

В симметричной криптосистеме секретный ключ надо передавать отправителю и получателю по защищенному каналу распространения ключей, например, спецсвязью. В асимметричной криптосистеме передают по незащищенному каналу только открытый ключ, а секретный ключ сохраняют в месте его генерации.

Злоумышленник при атаке на криптосистему может не только считывать шифротексты, передаваемые по каналу связи, но и пытаться их изменить.

Любая попытка злоумышленника расшифровать шифротекст C для получения открытого текста M или зашифровать свой собственный текст M^1 для получения правдоподобного шифротекста C^1, не имея подлинного ключа, называется *криптоатакой*.

Свойство криптосистемы противостоять криптоатаке называется *криптостойкостью*. Оно измеряется в затратах злоумышленника, которые он несет, вскрывая криптосистему. Например, криптостойкость может выражаться в количестве машинного времени, затраченного на вскрытие криптосистемы.

Фундаментальное правило криптоанализа, впервые сформулированное голландцем А. Керкхоффом еще в XIX веке, заключается в следующем: стойкость шифра должна определяться только секретностью ключа. Иными словами, правило Керкхоффа состоит в том, что весь алгоритм шифрования, кроме значения секретного ключа, известен криптоаналитику противника. Это обусловлено тем, что криптосистема, реализующая семейство криптографических преобразований, обычно рассматривается как открытая система. Такой подход отражает важный принцип технологии защиты информации: защищенность системы не должна зависеть от секретности чего-либо такого, что невозможно быстро изменить в случае утечки секретной информации. Обычно криптосистема представляет собой совокупность аппаратных и программных средств, которую можно изменить только при значительных затратах времени и средств, тогда как ключ является легко изменяемым объектом. Именно поэтому стойкость криптосистемы определяется только секретностью ключа.

Другое почти общепринятое допущение в криптоанализе состоит в том, что криптоаналитик имеет в своем распоряжении шифротексты сообщений.

Существует четыре основных типа криптоаналитических атак. Все они формулируются в предположении, что криптоаналитику известны применяемый алгоритм шифрования и шифротексты сообщений.

1. Криптоаналитическая атака при наличии известного шифртекста. Криптоаналитик имеет только шифротексты C_1, C_2, ..., C_i нескольких сообщений, причем они зашифрованы с использованием одного и того же алгоритма шифрования E_K. Работа криптоаналитика заключается в том, чтобы раскрыть исходные тексты $M_1, M_2, \ldots, M_i$ большинства сообщений или вычислить ключ K, использованный для их зашифровывания, с тем чтобы расшифровать и другие сообщения, зашифрованные этим ключом.

2. Криптоаналитическая атака при наличии известного открытого текста. Криптоаналитик имеет доступ не только к шифротек-

стам C_1, C_2, ... , C_i нескольких сообщений, но также и к открытым текстам $M_1, M_2, \ldots, M_i$ этих сообщений. Его работа заключается в нахождении ключа K, используемого при их шифровании, или алгоритма расшифровывания D_K любых новых сообщении, зашифрованных тем же ключом.

3. Криптоаналитическая атака при возможности выбора открытого текста. Криптоаналитик не только имеет доступ к шифротекстам C_1, C_2, ..., C_i и связанным с ними открытым текстам $M_1, M_2, \ldots, M_i$ нескольких сообщений, но и может выбирать открытые тексты, которые затем получает в зашифрованном виде. Такой тип анализа получается более мощным по сравнению с криптоанализом с известным открытым текстом, потому что криптоаналитик может выбрать для шифрования такие блоки открытого текста, которые дадут больше информации о ключе. Работа криптоаналитика состоит в поиске ключа K, использованного для шифрования сообщений, или алгоритма расшифровывания D_K новых сообщений, зашифрованных тем же ключом.

4. Криптоаналитическая атака с адаптивным выбором открытого текста – это особый вариант атаки с выбором открытого текста. Криптоаналитик может не только выбирать открытый текст, который затем шифруется, но и изменять свой выбор в зависимости от результатов предыдущего шифрования. При криптоанализе с простым выбором открытого текста криптоаналитик обычно может выбирать несколько крупных блоков открытого текста для их шифрования; при криптоанализе с адаптивным выбором открытого текста он имеет возможность выбрать сначала более мелкий пробный блок открытого текста, затем выбрать следующий блок в зависимости от результатов первого выбора и т. д. Эта атака предоставляет криптоаналитику еще больше возможностей, чем предыдущие типы.

Кроме перечисленных криптоаналитических атак существует *силовая атака*, которая заключается в переборе всех возможных значений ключа. С появлением мощных компьютеров и сетей этот вид атаки стал актуальным, так как он может сочетаться с перечисленными ранее типами. В связи с этим ключ криптосистемы должен обладать определенными свойствами: если рассматривать его совокупность двоичных знаков, то это должна быть случайная равномерно

распределенная последовательность длины, которая делала бы перебор всех возможных значений ключа практически невозможным.

1.2. Блочные и поточные шифры

Применение функции шифрования ко всему сообщению реализуется редко. Практически все применяемые криптографические методы связаны с разбиением сообщения на большое число фрагментов (или знаков) фиксированного размера, каждый из которых шифруется отдельно. Такой подход существенно упрощает задачу шифрования, так как сообщения обычно имеют различную длину.

Можно выделить следующие характерные признаки методов шифрования данных:

– выполнение операций с отдельными битами или блоками;

– зависимость или независимость функции шифрования от результатов шифрования предыдущих частей сообщения;

– зависимость или независимость шифрования отдельных знаков от их положения в тексте.

В соответствии с этим различают три основных способа шифрования:

– поточный;

– блочный;

– блочный с обратной связью.

Поточное шифрование. Поточное шифрование состоит в том, что каждый бит открытого текста и соответствующий бит ключа преобразовываются по определенному алгоритму (например, складываются по модулю 2). К достоинствам поточных шифров относятся высокая скорость шифрования, относительная простота реализации и отсутствие размножения ошибок. Недостатком является необходимость использовать для каждого сообщения свой ключ. Это обусловлено тем, что если два различных сообщения шифруются на одном и том же ключе, то эти сообщения легко могут быть расшифрованы ($C_1 = M_1 + K$, $C_2 = M_2 + K$, $C_1 + C_2 = M_1 + M_2$, считая M_2 ключом можно вычислить M_1, т. к. M_2 не обладает свойствами ключа). Поэтому часто используют дополнительный, случайно выбираемый ключ сообщения, который передается в начале сообщения и применяется для модификации ключа шифрования. В результате

разные сообщения будут шифроваться с помощью различных последовательностей. Это требует передачи информации синхронизации перед заголовком сообщения, которая должна быть принята до расшифровывания любого сообщения.

Поточные шифры широко применяются для шифрования преобразованных в цифровую форму речевых сигналов и цифровых данных, требующих оперативной доставки потребителю информации.

Блочное шифрование. При блочном шифровании открытый текст сначала разбивается на равные по длине блоки, затем применяется зависящая от ключа функция шифрования для преобразования блока открытого текста длиной m бит в блок шифротекста такой же длины. При блочном шифровании каждый бит блока шифротекста зависит от значений всех битов соответствующего блока открытого текста, и никакие два блока открытого текста не могут быть представлены одним и тем же блоком шифротекста. При этом незначительные изменения в шифротексте вызывают большие и непредсказуемые изменения в соответствующем открытом тексте, и наоборот. Вместе с тем применение блочного шифра имеет серьезные недостатки. Первый из них заключается в том, что вследствие детерминированного характера шифрования при фиксированной длине блока 64 бита можно осуществить криптоанализ шифротекста «со словарем» в ограниченной форме. Это обусловлено тем, что идентичные блоки открытого текста длиной 64 бита в исходном сообщении представляются идентичными блоками шифротекста, что позволяет криптоаналитику сделать определенные выводы о содержании сообщения. Другой потенциальный недостаток этого шифра связан с размножением ошибок: результатом изменения только одного бита в принятом блоке шифротекста будет неправильное расшифровывание всего блока. Это, в свою очередь, приведет к появлению искаженных битов (от 1 до 64) в восстановленном блоке исходного текста.

Из-за отмеченных недостатков блочные шифры редко применяются в указанном режиме для шифрования длинных сообщений. Однако в финансовых учреждениях, где сообщения часто состоят из одного или двух блоков, блочные шифры широко используют в режиме прямого шифрования. Такое применение обычно связано с возможностью частой смены ключа шифрования, поэтому вероят-

ность шифрования двух идентичных блоков открытого текста одним и тем же ключом очень мала.

Криптосистема с открытым ключом также является системой блочного шифрования и должна оперировать блоками довольно большой длины. Это обусловлено тем, что криптоаналитик знает открытый ключ шифрования и может заранее вычислить и составить таблицу соответствия блоков открытого текста и шифротекста. Если длина блоков мала, например 30 бит, то число возможных блоков не слишком большое (при длине 30 бит это $2^{30} \approx 10^9$), и может быть составлена полная таблица, позволяющая моментально расшифровать любое сообщение с использованием известного открытого ключа.

Блочное шифрование с обратной связью. Как и при блочном шифровании, сообщения разбивают на ряд блоков, состоящих из m бит. Для преобразования этих блоков в блоки шифротекста, которые также состоят из m бит, используются специальные функции шифрования. Однако если в блочном шифре такая функция зависит только от ключа, то в блочных шифрах с обратной связью она зависит как от ключа, так и от одного или более предшествующих блоков шифротекста.

Важным шифром с обратной связью является шифр со сцеплением блоков шифротекста. В этом случае m бит предыдущего шифротекста суммируются по модулю 2 со следующими m битами открытого текста, а затем применяется алгоритм блочного шифрования под управлением ключа для получения следующего блока шифротекста. Достоинством криптосистем блочного шифрования с обратной связью является возможность применения их для обнаружения манипуляций сообщениями, производимых активными перехватчиками. При этом используется факт размножения ошибок в таких шифрах, а также способность этих систем легко генерировать код аутентификации сообщений. Поэтому системы шифрования с обратной связью используют не только для шифрования сообщений, но и для их аутентификации. Криптосистемам блочного шифрования с обратной связью свойственны некоторые недостатки. Основным из них является размножение ошибок, так как один ошибочный бит при передаче может вызвать ряд ошибок в расшифрованном тексте. Другой недостаток связан с тем, что разработка и реализация систем шифрования

с обратной связью часто оказываются более трудными, чем систем поточного шифрования.

На практике для шифрования длинных сообщений применяют поточные шифры или шифры с обратной связью. Выбор конкретного типа шифра зависит от назначения системы и предъявляемых к ней требований.

2. СИММЕТРИЧНЫЕ КРИПТОСИСТЕМЫ

2.1. Основные понятия и определения

Большинство средств защиты информации базируется на использовании криптографических шифров и процедур зашифровывания-расшифровывания. В соответствии с ГОСТ 28147–89 под *шифром* понимают совокупность обратимых преобразований множества открытых данных на множество зашифрованных данных, задаваемых ключом и алгоритмом криптографического преобразования.

Ключ – это конкретное секретное состояние некоторых параметров алгоритма криптографического преобразования данных, обеспечивающее выбор только одного варианта из всех возможных для данного алгоритма. Основной характеристикой шифра является криптостойкость, которая определяет его стойкость к раскрытию методами криптоанализа. Обычно эта характеристика определяется интервалом времени, необходимым для раскрытия шифра. К шифрам, используемым для криптографической защиты информации, предъявляется ряд требований:

- достаточная криптостойкость, т. е. надежность закрытия данных;
- простота процедур шифрования и расшифровывания;
- незначительная избыточность информации за счет шифрования;
- нечувствительность к небольшим ошибкам шифрования и др.

Этим требованиям отвечают шифры:

- перестановок;
- замены;
- гаммирования;
- основанные на аналитических преобразованиях шифруемых данных.

Шифрование перестановкой заключается в том, что символы шифруемого текста переставляются по определенному правилу в пределах некоторого блока этого текста. При достаточной длине блока, в пределах которого осуществляется перестановка, и сложном неповторяющемся порядке перестановки можно достигнуть приемлемой для простых практических приложений стойкости шифра.

Шифрование заменой (подстановкой) заключается в том, что символы шифруемого текста заменяются символами того же или другого алфавита в соответствии с заранее обусловленной схемой замены.

Шифрование гаммированием заключается в том, что символы шифруемого текста складываются с символами некоторой случайной последовательности, именуемой *гаммой шифра*. Стойкость шифрования определяется в основном длиной (периодом) неповторяющейся части гаммы шифра. Поскольку с помощью ЭВМ можно генерировать практически бесконечную гамму шифра, то данный способ является одним из основных для шифрования информации в автоматизированных системах.

Шифрование аналитическим преобразованием заключается в том, что шифруемый текст преобразуется по некоторому аналитическому правилу (формуле). Например, можно использовать правило умножения вектора на матрицу, причем умножаемая матрица является ключом шифрования, поэтому ее размер и содержание должны храниться в секрете, а символами умножаемого вектора последовательно служат символы шифруемого текста.

Процессы зашифровывания и расшифровывания осуществляются в рамках некоторой криптосистемы. Характерной особенностью симметричной криптосистемы является применение одного секретного ключа как при зашифровывании, так и при расшифровывании сообщений.

Как открытый текст, так и шифротекст образуются из букв, входящих в конечное множество символов, называемое *алфавитом*. Примерами алфавитов являются конечное множество всех заглавных букв, заглавных и строчных букв и цифр и т. п. В общем случае некоторый алфавит можно представить как $\sum = \{a_0, a_1, \ldots, a_{m-1}\}$.

Объединяя по определенному правилу буквы из алфавита $\sum$, можно создать новые алфавиты:

– алфавит $\sum^2$ содержит m^2 биграмм $a_0a_0, a_0a_1, \ldots, a_{m-1}a_{m-1}$;

– алфавит $\sum^3$ содержит m^3 биграмм $a_0a_0, a_0a_1, \ldots, a_{m-1}a_{m-1}a_{m-1}$.

В общем случае, объединяя по n букв, получаем алфавит $\sum^n$, содержащий m^n–n-грамм. Например, английский алфавит {*ABCD…XYZ*} объемом $m = 26$ букв позволяет сгенерировать $26^2 = 676$ биграмм *AA*, *AB*,…, *ZZ*, $26^2 = 17\,576$ триграмм *AAA*, *AAB*,…, *ZZY*, ZZZ.

При выполнении криптографических преобразований полезно заменить буквы алфавита целыми числами (0, 1, 2, …). Это позволит упростить выполнение необходимых алгебраических манипуляций. Например, можно установить взаимно однозначное соответствие между русским алфавитом {АБВГ…ЮЯ} и множеством целых чисел $\sum_{32} = \{0, 1, 2, \ldots, 32\}$, между английским алфавитом {*ABCD…YZ*} и множеством целых чисел $\sum_{32} = \{0, 1, 2, \ldots, 26\}$.

В дальнейшем обычно будет использоваться алфавит $\overline{\sum}_m = \{0, 1, 2, \ldots, m-1\}$, содержащий m «букв» в виде чисел. Замена букв традиционного алфавита числами позволит более четко сформулировать основные концепции и приемы криптопреобразований. В то же время в большинстве иллюстраций будет использоваться алфавит естественного языка.

Текст с n буквами из алфавита $\overline{\sum}_m$ можно рассматривать как n-грамму $\overline{X} = \{x_0, x_1, \ldots, x_{n-1}\}$, где $x \in \overline{Z}_m$, для некоторого целого $n = 1, 2, 3\ldots$. Через $Z_{m,n}$ будем обозначать множество n-грамм, образованных из букв множества Z_m.

Криптографическое преобразование E представляет собой совокупность преобразований $E = \{E^{(n)} : 1 \le n < \infty\}$, $E^{(n)} : Z_{m,n} \to Z_{m,n}$. Преобразование $E^{(n)}$ определяет, как каждая n-грамма открытого текста $\overline{x} \in \overline{Z}_{m,n}$ заменяется n-граммой шифротекста $\overline{y}$, т. е. $\overline{y} = E^{(n)}(\overline{x})$, причем $\overline{x}$, $\overline{y} \in \overline{Z}_{m,n}$, при этом обязательным является требование взаимной однозначности преобразования $E^{(n)}$ на множестве $\overline{Z}_{m,n}$.

Криптографическая система может трактоваться как семейство криптографических преобразований $E = \{E_K : K \in \overline{k}\}$, помеченных параметром K, называемым ключом. Множество значений ключа образуют ключевое пространство $\overline{k}$.

2.2. Традиционные симметричные криптосистемы

Традиционные (классические) методы шифрования отличаются симметричной функцией шифрования. К ним относятся шифры перестановки, простой и сложной замены, а также некоторые их модификации и комбинации. Следует отметить, что комбинации шифров перестановок и замены образуют все многообразие применяемых на практике симметричных шифров.

Шифры перестановок. Правило перестановок символов является ключом и задается различными предметами: цилиндром (скитала, древние греки), размером таблицы, условным словом или фразой (шифрующие таблицы в эпоху Возрождения), магическим квадратом в Средние века.

Шифры простой замены. В шифрах простой замены каждый символ открытого текста заменяется символом того же или другого алфавита по определенному правилу. Широко известны и исследованы шифры Цезаря. Они имеют слабость по отношению к атакам на основе подсчета частот появления букв в шифротексте. Более устойчивыми являются биграммные шифры (замена двух букв) и n-граммные шифры, позволяющие маскировать частоты появления букв.

Шифры сложной замены. Шифры сложной замены называют многоалфавитными, т. к. для шифрования каждого символа исходного сообщения применяют свой шифр простой замены. Многоалфавитная подстановка последовательно и циклически меняет используемые алфавиты. Например, в r-алфавитной подстановке символ x_0 исходного текста заменяется символом y_0 из алфавита B_0, x_1 на y_1 из B_1, x_{r-1} на y_{r-1} из B_{r-1}, x_r на y_r из B_r. Многоалфавитная подстановка маскирует естественную статистику исходного языка, т. к. конкретный символ из алфавита A может быть преобразован в несколько символов шифровальных алфавитов B. К шифрам сложной замены относят шифры Гронсфельда, Вижинера, Вернама. В 20-х годах были созданы первые шифровальные машины (электромеханические), реализующие шифры сложной замены. Этот тип машин использовался до 60-х годов.

Шифрование методом гаммирования. Под *гаммированием* понимают процесс наложения по определенному закону гаммы шифра на открытые данные. *Гамма шифра* – это псевдослучайная последовательность, выработанная по заданному алгоритму для зашифровывания открытых данных и расшифровывания зашифрованных данных. Процесс зашифровывания заключается в генерации гаммы шифра и наложении полученной гаммы на исходный открытый текст обратимым образом, например с использованием операции сложения по модулю 2.

Следует отметить, что перед зашифровыванием открытые данные разбивают на блоки $T_о^i$ одинаковой длины, обычно по 64 бита. Гамма шифра вырабатывается в виде последовательности блоков $Г_ш^i$ аналогичной длины.

Уравнение зашифровывания можно записать в виде

$$T_ш^i = Г_ш^i \oplus T_о^i,\ i = 1, 2, \ldots, M,$$

где $T_о$ – *i*-й блок шифротекста;

$Г_ш^i$ – *i*-й блок гаммы шифра;

$T_о^i$ - *i*-й блок открытого текста;

M – количество блоков открытого текста.

Процесс расшифровывания сводится к повторной генерации гаммы шифра и наложению этой гаммы на зашифрованные данные. Уравнение расшифровывания имеет следующий вид:

$$T_о^i = Г_ш^i \oplus T_ш^i.$$

Получаемый этим методом шифротекст достаточно труден для раскрытия, поскольку теперь ключ является переменным. По сути дела гамма шифра должна изменяться случайным образом для каждого шифруемого блока. Если период гаммы превышает длину всего шифруемого текста и злоумышленнику неизвестна никакая часть исходного текста, то такой шифр можно раскрыть только прямым перебором всех вариантов ключа. В этом случае криптостойкость шифра определяется длиной ключа.

2.3. Современные симметричные криптосистемы

По мнению К. Шеннона, в практических шифрах необходимо использовать два общих принципа: рассеивание и перемешивание.

Рассеивание представляет собой распространение влияния одного знака открытого текста на много знаков шифротекста, что позволяет скрыть статистические свойства открытого текста.

Перемешивание предполагает использование таких шифрующих преобразований, которые усложняют восстановление взаимосвязи статистических свойств открытого и шифрованного текстов. Однако шифр должен не только затруднять раскрытие, но и обеспечивать легкость зашифровывания и расшифровывания при известном пользователю секретном ключе.

Распространенным способом достижения эффектов рассеивания и перемешивания является использование составного шифра, т. е. такого шифра, который может быть реализован в виде некоторой последовательности простых шифров, каждый из которых вносит свой вклад в значительное суммарное рассеивание и перемешивание.

В составных шифрах в качестве простых чаще всего используются простые перестановки и подстановки. При перестановке перемешивают символы открытого текста, причем конкретный вид перемешивания определяется секретным ключом. При подстановке каждый символ открытого текста заменяют другим символом из того же алфавита, а конкретный вид подстановки также определяется секретным ключом. В современном блочном шифре блоки открытого текста и шифротекста представляют собой двоичные последовательности длиной 64 бита. Каждый блок может принимать 2^{64} значений, поэтому подстановки выполняются в большом алфавите, содержащем до $2^{54} \approx 10^{19}$ символов.

При многократном чередовании простых перестановок и подстановок, управляемых достаточно длинным секретным ключом, можно получить достаточно стойкий шифр с хорошим рассеиванием и перемешиванием.

3. СТАНДАРТ ШИФРОВАНИЯ ДАННЫХ ГОСТ 28147–89

ГОСТ 28147–89 представляет собой 64-битовый блочный алгоритм с 256-битовым ключом. Он предназначен для аппаратной и программной реализации, удовлетворяет криптографическим требованиям и не накладывает ограничений на степень секретности защищаемой информации.

Алгоритм предусматривает четыре режима работы шифрования данных:

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

Основными режимами шифрования являются режимы с использованием гаммирования, однако они базируются на использовании шифрования данных в режиме простой замены.

3.1. Режим простой замены

Шифрование открытых данных в режиме простой замены. Открытые данные, подлежащие шифрованию, разбивают на 64-разрядные блоки T_o. Процедура шифрования 64-разрядного блока T_o в режиме простой замены включает 32 цикла ($j = 1, 2, \ldots, 32$). В ключевое запоминающее устройство вводят 256 бит ключа K в виде восьми 32-разрядных подключей (чисел) K_i

$$K = K_7 K_6 K_5 K_4 K_3 K_2 K_1 K_0.$$

Последовательность битов блока

$$T_0 = (a_1(0), a_2(0), \ldots, a_{31}(0), a_{32}(0), b_1(0), b_2(0), \ldots, b_{31}(0), b_{32}(0))$$

разбивают на две последовательности по 32 бита: $b(0)$ и $a(0)$, где $b(0)$ - левые или старшие биты, $a(0)$ - правые или младшие биты.

Работа алгоритма в режиме простой замены изображена на рис. 3.1.

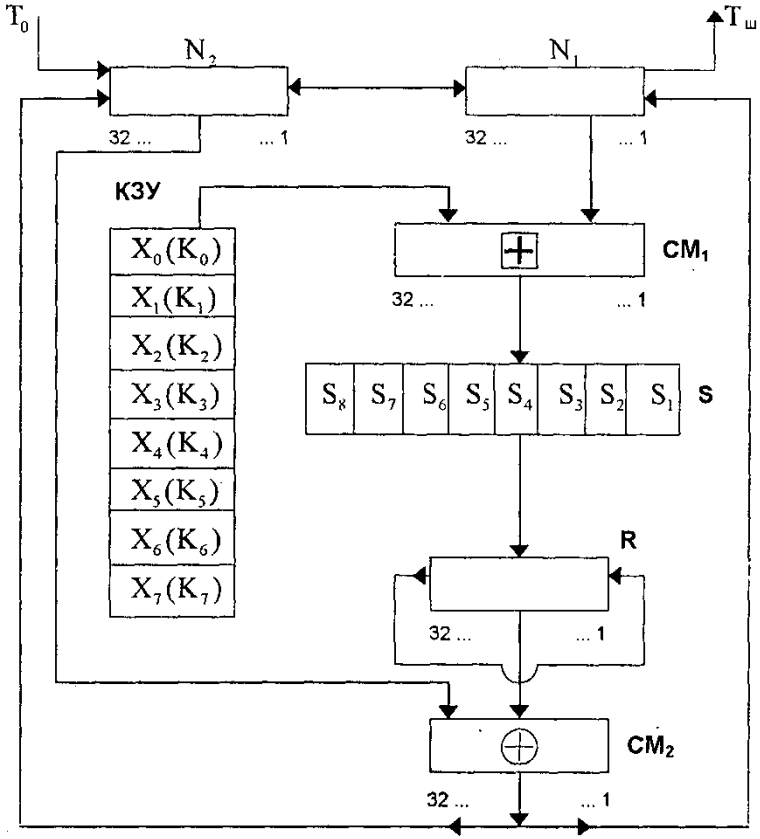

Рис. 3.1. Схема реализации режима простой замены

Обозначения на схеме:

N_1, N_2 – 32-разрядные накопители;

CM_1 – 32-разрядный сумматор по модулю 2^{32} (+);

CM_2 – 32-разрядный сумматор по модулю 2($\oplus$);

R – 32-разрядный регистр циклического сдвига;

КЗУ – ключевое запоминающее устройство на 256 бит, состоящее из восьми 32-разрядных накопителей X_0, X_1, X_2, ... , X_7;

S – блок подстановки, состоящий из восьми узлов замены (S-блоков замены) S_1, S_2, S_3, … , S_8.

Эти последовательности вводят в накопители N_1 и N_2 перед началом первого цикла шифрования. В результате начальное заполнение накопителя N_1

$$a(0) = (a_{32}(0), a_{31}(0), \dots, a_2(0), a_1(0))$$
$$32, \quad 31, \quad \dots, \quad 2, \quad 1 \quad \leftarrow \text{номер разряда } N_1,$$

начальное заполнение накопителя N_2

$$b(0) = (b_{32}(0), b_{31}(0), \dots, b_2(0), b_1(0))$$
$$32, \quad 31, \quad \dots, \quad 2, \quad 1 \quad \leftarrow \text{номер разряда } N_2,$$

Первый цикл ($j = 1$) процедуры шифрования 64-разрядного блока открытых данных можно описать уравнениями:

$$\begin{cases} a(1) = f(a(0) + K_0) \oplus b(0); \\ b(1) = a(0), \end{cases}$$

где $a(1)$ – заполнение N_1 после 1-го цикла шифрования;

$b(1)$ – заполнение N_2 после 1-го цикла шифрования;

f – функция шифрования.

Аргументом функции f является сумма по модулю 2^{32} числа $a(0)$ (начального заполнения накопителя N_1) и числа K_0 подключа, считываемого из накопителя X_0 КЗУ. Каждое из этих чисел равно 32 битам.

Функция f включает две операции над полученной 32-разрядной суммой $(a(0) + K_0)$.

Первая операция называется подстановкой (заменой) и выполняется блоком подстановки S. Блок подстановки S состоит из восьми узлов замены (S-блоков замены) $S_1, S_2, \dots, S_8$ с памятью 64 бит каждый. Поступающий из CM_2 на блок подстановки S 32-разрядный вектор разбивают на восемь последовательно идущих 4-разрядных векторов, каждый из которых преобразуется в 4-разрядный соответствующим узлом замены. Каждый узел замены можно представить в виде таблицы-перестановки шестнадцати 4-разрядных двоичных чисел в диапазоне 0000–1111. Входной вектор указывает адрес строки в таблице, а число в этой строке является выходным вектором. Затем 4-разрядные выходные векторы последовательно объединяют

в 32-разрядный вектор. Узлы замены (таблицы-перестановки) представляют собой ключевые элементы, которые являются общими для сетей ТКС и редко изменяются. Эти узлы замены должны сохраняться в секрете.

Вторая операция – циклический сдвиг влево (на 11 разрядов) 32-разрядного вектора, полученного с выхода блока подстановки S. Циклический сдвиг выполняется регистром сдвига R. Затем результат работы функции шифрования f суммируют поразрядно по модулю 2 в сумматоре CM_2 с 32-разрядным начальным заполнением $b(0)$ накопителя N_2. Затем полученный на выходе CM_2 результат (значение $a(1)$) записывают в накопитель N_1, а старое значение N_1 (значение $a(0)$) переписывают в накопитель N_2 (значение $b(1) = a(0)$). Таким образом, первый цикл завершен. Последующие циклы осуществляются аналогично, при этом во втором цикле из КЗУ считывают заполнение X_1 – подключ K_1, в третьем цикле – подключ K_2 и т. д., в восьмом цикле – подключ K_7. В циклах с 9-го по 16-й, а также с 17-го по 24-й подключи из КЗУ считываются в том же порядке: $K_0, K_1, K_2, \ldots, K_6, K_7$. В циклах с 25-го по 32-й порядок считывания подключей из КЗУ обратный: $K_7, K_6, \ldots K_2, K_1, K_0$. Таким образом, при шифровании в 32 циклах осуществляется следующий порядок выборки из КЗУ подключей:

$$K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7,$$

$$K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0.$$

В 32-м цикле результат из сумматора CM_2 вводится в накопитель N_2, а в накопителе N_1 сохраняется прежнее заполнение. Полученные после 32-го цикла шифрования заполнения накопителей N_1 и N_2 являются блоком шифрованных данных $T_ш$, соответствующим блоку открытых данных $T_о$.

Уравнения шифрования в режиме простой замены имеют следующий вид:

$$\begin{cases} a(j) = f(a(j-1) + K_{(j-1)\bmod 8}) \oplus b(j-1) \\ b(j) = a(j-1) \end{cases} \quad \text{при} \quad j = 1\text{–}24;$$

$$\begin{cases} a(j) = f\left(a(j-1) + K_{(32-j)\bmod 8}\right) \oplus b(j-1) \\ b(j) = a(j-1) \end{cases} \quad \text{при} \quad j = 25\text{–}31;$$

$$\begin{cases} a(j) = a(j-1) \\ b(j) = f(a(j-1) + K_0) \oplus b(j-1) \end{cases} \quad \text{при} \quad j = 32,$$

где $a(j) = (a_{32}(j), a_{31}(j), \ldots, a_1(j))$ – заполнение N_1 после j-го цикла шифрования;

$b(j) = (b_{32}(j), b_{31}(j), \ldots, b_1(j))$ – заполнение N_2 после j-го цикла шифрования;

$j = 1\text{–}32$.

Блок зашифрованных данных $T_ш$ (64 разряда) выводится из накопителей N_1, N_2 в следующем порядке: из разрядов 1–32 накопителя N_1, затем из разрядов 1–32 накопителя N_2, т. е. начиная с младших разрядов

$$T_ш = (a_1(32), a_2(32), \ldots, a_{32}(32), b_1(32), b_2(32), \ldots, b_{32}(32)).$$

Остальные блоки открытых данных зашифровываются в режиме простой замены аналогично.

Расшифрование в режиме простой замены. Криптосхема, реализующая алгоритм расшифрования в режиме простой замены, имеет тот же вид, что и при шифровании (см. рис. 1.1).

В КЗУ вводят 256 бит ключа, на котором осуществлялось шифрование. Зашифрованные данные, подлежащие расшифрованию, разбиты на блоки $T_ш$, по 64 бита в каждом. Ввод любого блока

$$T_ш = (a_1(32), a_2(32), \ldots, a_{32}(32), b_1(32), b_2(32), \ldots, b_{32}(32))$$

в накопители N_1 и N_2 производят так, чтобы начальное значение накопителя N_1 имело вид

$$a(0) = (a_{32}(32), a_{31}(32), \ldots, a_2(32), a_1(32))$$
$$32, \quad 31, \quad \ldots, \quad 2, \quad 1 \quad \leftarrow \text{номер разряда } N_1,$$

а начальное заполнение накопителя N_2

$$b(0) = (b_{32}(32), b_{31}(32), \ldots, b_2(32), b_1(32))$$
$$32, \quad 31, \quad \ldots, \quad 2, \quad 1 \quad \leftarrow \text{номер разряда } N_2.$$

Расшифровывание осуществляется по тому же алгоритму, что и шифрование, с тем изменением, что заполнения накопителей X_0, X_1, X_2,..., X_7 считываются из КЗУ в циклах расшифровывания в следующем порядке:

$$K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0,$$
$$K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0, K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0.$$

Уравнения расшифровывания имеют следующий вид:

$$\begin{cases} a(32-j) = f\left(a(32-j+1) + K_{(j-1)}\right) \oplus b(32-j+1) \\ b(32-j) = a(32-j+1) \end{cases} \quad \text{при} \quad j = 1\text{–}8;$$

$$\begin{cases} a(32-j) = f\left(a(32-j+1) + K_{(32-j)\bmod 8}\right) \oplus b(32-j+1) \\ b(32-j) = a(32-j+1) \end{cases} \text{при } j = 9\text{–}31;$$

$$\begin{cases} a(0) = a(1) \\ b(0) = f(a(1) + K_0) \oplus b(1) \end{cases} \quad \text{при} \quad j = 32.$$

Полученные после 32 циклов работы заполнения накопителей N_1 и N_2 образуют блок открытых данных

$$T_o = (a_1(0), a_2(0), \ldots, a_{32}(0), b_1(0), b_2(0), \ldots, b_{32}(0)),$$

соответствующий блоку зашифрованных данных $T_ø$. При этом состояние накопителя N_1

$$a(0) = (a_{32}(0), a_{31}(0), \ldots, a_2(0), a_1(0))$$
$$32, \quad 31, \quad \ldots, \quad 2, \quad 1 \quad \leftarrow \text{ номер разряда } N_1$$

состояние накопителя N_2

$$b(0) = (b_{32}(0), b_{31}(0), \ldots, b_2(0), b_1(0))$$
$$32, \quad 31, \quad \ldots, \quad 2, \quad 1 \quad \leftarrow \text{ номер разряда } N_2.$$

Аналогично расшифровываются остальные блоки зашифрованных данных.

Если алгоритм зашифровывания в режиме простой замены 64-битового блока $\mathrm{T_o}$ обозначить через A, то

$$A(\mathrm{T_o}) = A\big(a(0), b(0)\big) = (a(32), b(32)) = \mathrm{T_ш}.$$

Следует иметь в виду, что режим простой замены допустимо использовать для шифрования данных только при выработке ключа и шифровании его с обеспечением имитозащиты для передачи по каналам связи или для хранения в памяти ЭВМ.

3.2. Режим гаммирования

Зашифровывание открытых данных в режиме гаммирования. Криптосхема, реализующая алгоритм шифрования в режиме гаммирования, показана на рис. 3.2. Открытые данные разбивают на 64-разрядные блоки

$$\mathrm{T_o^{(1)}}, \mathrm{T_o^{(2)}}, \ldots, \mathrm{T_o^{(i)}}, \ldots, \mathrm{T_o^{(m)}},$$

где $\mathrm{T_o^{(i)}}$ – i-й 64-разрядный блок открытых данных;

$i = 1, \ldots, m$; m определяется объемом шифруемых данных.

Эти блоки поочередно зашифровываются в режиме гаммирования путем поразрядного сложения по модулю 2 в сумматоре CM_5 с гаммой шифра $\Gamma_ш$, которая вырабатывается блоками по 64 бита, т. е.

$$\Gamma_ш = (\Gamma_ш^{(1)}, \Gamma_ш^{(2)}, \ldots, \Gamma_ш^{(i)}, \ldots, \Gamma_ш^{(m)}),$$

где $\Gamma_ш^{(i)}$ – i-й 64-разрядный блок;

$i = 1, \ldots, m$.

Число двоичных разрядов в блоке $\mathrm{T_o^{(m)}}$ может быть меньше 64, при этом не использованная для зашифровывания часть гаммы шифра из блока $\Gamma_ш^{(m)}$ отбрасывается.

Уравнение шифрования данных в режиме гаммирования имеет вид

$$\mathrm{T_ш^{(i)}} = \mathrm{T_o^{(i)}} \oplus \Gamma_ш^{(i)},$$

где $\Gamma_{ш}^{(i)} = A\left(Y_{i-1} + C_2\ , Z_{i-1} + C_1\right)$;

$\Gamma_{ш}^{(i)}$ – i-й блок 64-разрядного блока зашифрованного текста;

$A(*)$ – функция шифрования в режиме простой замены;

C_1, C_2 – 32-разрядные двоичные константы;

Y_i, Z_i – 32-разрядные двоичные последовательности;

$i = 1, \ldots, m$.

Величины Y_i, Z_i определяются итерационно по мере формирования гаммы $\Gamma_{ш}^{(i)}$ следующим образом:

$$(Y_0, Z_0) = A(\tilde{S}),$$

где $\tilde{S}$ – синхропосылка (64-разрядная двоичная последовательность),

$$(Y_i, Z_i) = (Y_{i-1} + C_2, Z_{i-1} + C_1),\ i = 1, \ldots, m.$$

Рассмотрим реализацию процедуры шифрования в режиме гаммирования. В накопители N_6 и N_5 заранее записаны 32-разрядные двоичные константы C_1 и C_2, имеющие следующие значения (в шестнадцатеричной форме):

$$C_1 = 01010104_{(16)},\ C_2 = 01010101_{(16)}.$$

В КЗУ вводится 256 бит ключа; в накопители N_1 и N_2 - 64-разрядная двоичная последовательность (синхропосылка):

$$\tilde{S} = (S_1, S_2, \ldots, S_{64}).$$

Синхропосылка $\tilde{S}$ является исходным заполнением накопителей N_1 и N_2 для последовательной выработки m блоков гаммы шифра.

Исходное заполнение накопителя N_1

$$\begin{array}{cccccc} (S_{32}, & S_{31}, & \ldots, & S_2, & S_1) & \\ 32, & 31, & \ldots, & 2, & 1 & \leftarrow \text{номер разряда } N_1, \end{array}$$

состояние накопителя N_2

$$\begin{array}{cccccc} (S_{64}, & S_{63}, & \ldots, & S_{34}, & S_{33}) & \\ 64, & 63, & \ldots, & 34, & 33 & \leftarrow \text{номер разряда } N_2. \end{array}$$

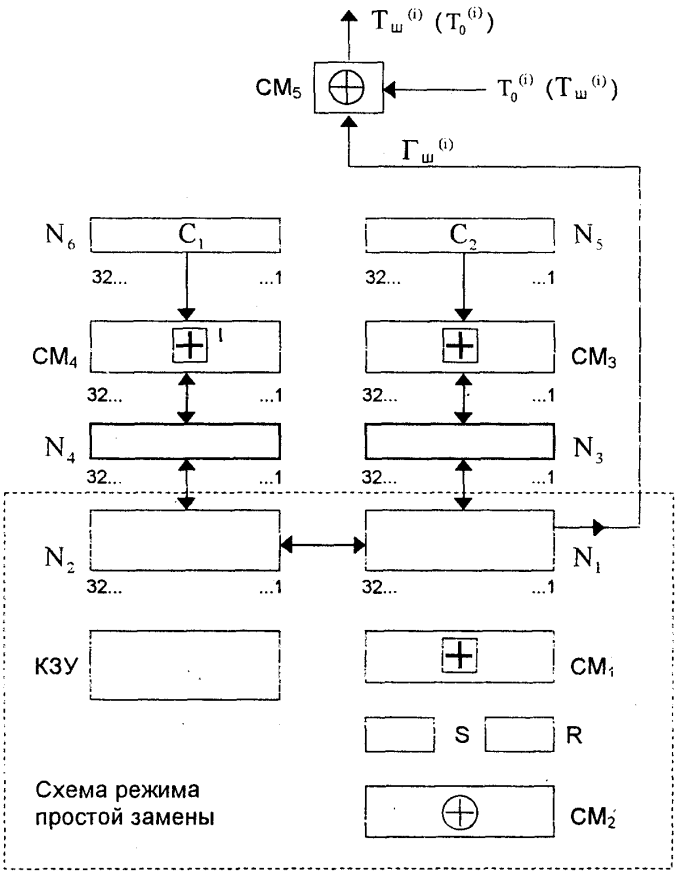

Рис. 3.2. Схема реализации режима гаммирования

Исходное заполнение N_1 и N_2 (синхропосылка $\tilde{S}$ шифруется в режиме простой замены. Результат шифрования

$$A(\tilde{S}) = (Y_0, Z_0)$$

переписывается в 32-разрядные накопители N_3 и N_4 так, что заполнение N_1 переписывается в N_3, а заполнение N_2 – в N_4.

Заполнение накопителя N_4 суммируют по модулю $2^{32} - 1$ в сумматоре CM_4 с 32-разрядной константой C_1 из накопителя N_6. Ре-

зультат записывается в N_4. Заполнение накопителя N_3 суммируется по модулю 2^{32} в сумматоре CM_3 с 32-разрядной константой C_2 из накопителя N_5. Результат записывается в N_3. Заполнение N_3 переписывают в N_1, а заполнение N_4 – в N_2, при этом заполнения N_3, N_4 сохраняются. Заполнение накопителей шифруется в режиме простой замены.

Полученное в результате шифрования заполнение накопителей N_1 и N_2 образует первый 64-разрядный блок гаммы шифра:

$$\Gamma_{\text{ш}}^{(1)} = (\gamma_1^{(1)}, \gamma_2^{(1)}, \dots, \gamma_{63}^{(1)}, \gamma_{64}^{(1)}),$$

который суммируют поразрядно по модулю 2 в сумматоре CM_5 с первым 64-разрядным блоком открытых данных:

$$\text{T}_{\text{o}}^{(1)} = (t_1^{(1)}, t_2^{(1)}, \dots, t_{63}^{(1)}, t_{64}^{(1)}).$$

В результате суммирования по модулю 2 значений $\Gamma_{\text{ш}}^{(1)}$ и $\text{T}_0^{(1)}$ получают первый 64-разрядный блок зашифрованных данных:

$$\text{T}_{\text{ш}}^{(1)} = \Gamma_{\text{ш}}^{(1)} \oplus \text{T}_{\text{o}}^{(1)} = (\tau_1^{(1)}, \tau_2^{(1)}, \dots, \tau_{63}^{(1)}, \tau_{64}^{(1)}),$$

где $\tau_i^{(1)} = t_i^{(1)} \oplus \gamma_i^{(1)}$; $i = 1–64$.

Для получения следующего 64-разрядного блока гаммы шифра $\Gamma_{\text{ш}}^{(2)}$ заполнение N_4 суммируется по модулю $(2^{32} – 1)$ в сумматоре CM_4 с константой C_1 из N_6. Результат записывается в N_4. Заполнение N_3 суммируется по модулю 2^{32} в сумматоре CM_3 с константой C_2 из N_5. Результат записывается в N_3. Новое заполнение N_3 переписывают в N_1, а новое заполнение N_4 – в N_2, при этом заполнения N_3 и N_2 сохраняют. Заполнения N_1, N_2 шифруют в режиме простой замены.

Полученное в результате шифрования заполнение накопителей N_1 и N_2 образует второй 64-разрядный блок гаммы шифра $\Gamma_{\text{ш}}^{(2)}$, который суммируется поразрядно по модулю 2 в сумматоре CM_5 со вторым блоком открытых данных $\text{T}_{\text{o}}^{(2)}$:

$$\text{T}_{\text{ш}}^{(2)} = \Gamma_{\text{ш}}^{(2)} \oplus \text{T}_{\text{o}}^{(2)}.$$

Аналогично вырабатываются блоки гаммы шифра $\Gamma_{ш}^{(3)}, \Gamma_{ш}^{(4)}, \ldots, \Gamma_{ш}^{(m)}$ и шифруются блоки открытых данных $T_{о}^{(3)}, T_{о}^{(3)}, \ldots, T_{о}^{(m)}$.

В канал связи или память ЭВМ передаются синхропосылка $\tilde{S}$ и блоки зашифрованных данных

$$T_{ш}^{(1)}, T_{ш}^{(2)}, \ldots, T_{ш}^{(m)}.$$

Расшифровывание в режиме гаммирования. При расшифровывании криптосхема имеет тот же вид, что и при шифровании (см. рис. 3.2).

Уравнение расшифровывания:

$$T_{о}^{(i)} = T_{ш}^{(i)} \oplus \Gamma_{ш}^{(i)} = T_{ш}^{(i)} \oplus A\left(Y_{i-1} + C_2, Z_{i-1} + C_1\right), \; i = 1, \ldots, m.$$

Следует отметить, что расшифровывание данных возможно только при наличии синхропосылки, которая не является секретным элементом шифра и может храниться в памяти ЭВМ или передаваться по каналам связи вместе с зашифрованными данными.

Рассмотрим реализацию процедуры расшифровывания. В КЗУ вводят 256 бит ключа, с помощью которого осуществляется шифрование данных $T_{о}^{(1)}, T_{о}^{(2)}, \ldots, T_{о}^{(m)}$. В накопители N_1 и N_2 вводится синхропосылка и осуществляется процесс выработки m блоков гаммы шифра $\Gamma_{ш}^{(1)}, \Gamma_{ш}^{(2)}, \ldots, \Gamma_{ш}^{(m)}$. Блоки зашифрованных данных $T_{ш}^{(1)}, T_{ш}^{(2)}, \ldots, T_{ш}^{(m)}$ суммируются поразрядно по модулю 2 в сумматоре CM_5 с блоками гаммы шифра $\Gamma_{ш}^{(1)}, \Gamma_{ш}^{(2)}, \ldots, \Gamma_{ш}^{(m)}$. В результате получаются блоки открытых данных $T_{о}^{(1)}, T_{о}^{(2)}, \ldots, T_{о}^{(m)}$, при этом $T_{о}^{(m)}$ может содержать меньше 64 разрядов.

3.3. Режим гаммирования с обратной связью

Криптосхема, реализующая алгоритм шифрования в режиме гаммирования с обратной связью, показана на рис. 3.3.

Шифрование открытых данных в режиме гаммирования с обратной связью. Открытые данные, разбитые на 64-разрядные блоки $T_{о}^{(1)}, T_{о}^{(2)}, \ldots, T_{о}^{(m)}$, шифруются в режиме гаммирования с обратной

связью путем поразрядного сложения по модулю 2 с гаммой шифра $\Gamma_{ш}$, которая вырабатывается блоками по 64 бита: $\Gamma_{ш}^{(1)}, \Gamma_{ш}^{(2)}, \ldots, \Gamma_{ш}^{(m)}$.

Число двоичных разрядов в блоке $T_0^{(m)}$ может быть меньше 64, при этом неиспользованная для шифрования часть гаммы шифра из блока $\Gamma_{ш}^{(m)}$ отбрасывается.

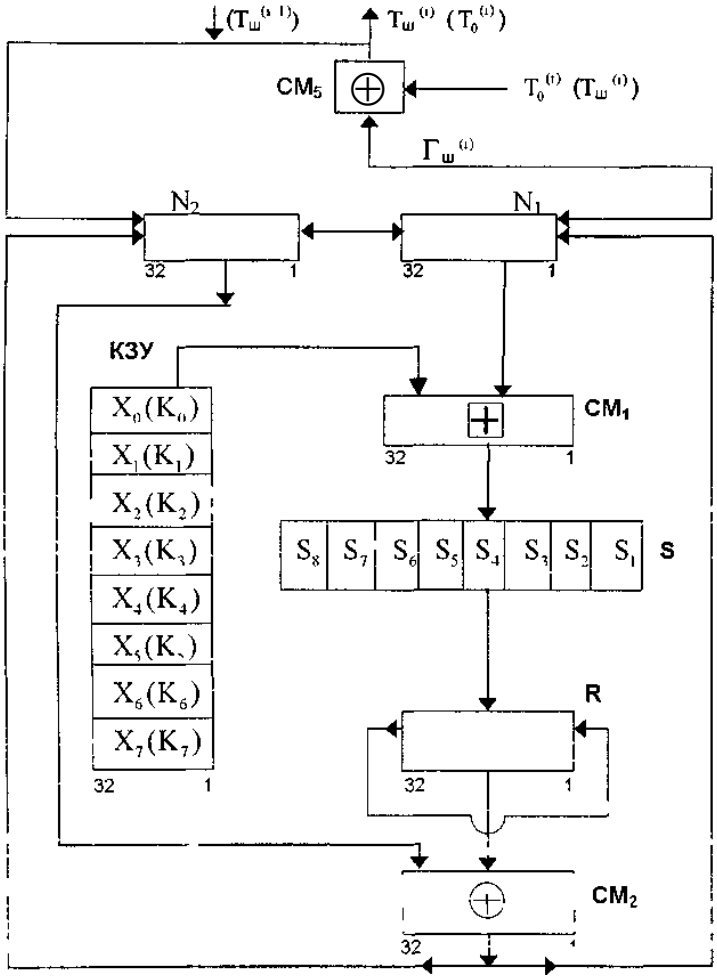

Рис. 3.3. Схема реализации режима гаммирования с обратной связью

Уравнения шифрования в режиме гаммирования с обратной связью имеют вид:

$$\mathrm{T}_{\text{ш}}^{(1)} = A\left(\tilde{S}\right) \oplus \mathrm{T}_{\text{о}}^{(1)} = \Gamma_{\text{ш}}^{(1)} \oplus \mathrm{T}_{\text{о}}^{(1)};$$

$$\mathrm{T}_{\text{ш}}^{(i)} = A\left(\mathrm{T}_{\text{ш}}^{(i-1)}\right) \oplus \mathrm{T}_{\text{о}}^{(i)} = \Gamma_{\text{ш}}^{(i)} \oplus \mathrm{T}_{\text{о}}^{(i)},$$

где $\mathrm{T}_{\text{ш}}^{(i)}$ – i-й 64-разрядный блок шифрованного текста;

$A(*)$ – функция шифрования в режиме простой замены;

m – определяется объемом открытых данных.

$i = 2, \ldots, m$.

Аргументом функции $A(*)$ на первом шаге итеративного алгоритма является 64-разрядная синхропосылка S, а на всех последующих шагах – предыдущий блок зашифрованных данных $\mathrm{T}_{\text{ш}}^{(i-1)}$.

Процедура шифрования данных в режиме гаммирования с обратной связью реализуется следующим образом. В КЗУ вводятся 256 бит ключа, в накопители N_1 и N_2 вводится синхропосылка $\tilde{S} = (S_1, S_2, \ldots, S_{64})$ из 64 бит. Исходное заполнение накопителей N_1 и N_2 шифруется в режиме простой замены. Полученное в результате шифрования заполнение накопителей N_1 и N_2 образует первый 64-разрядный блок гаммы шифра $\Gamma_{\text{ш}}^{(1)} = \grave{A}(\tilde{S})$, который суммируется поразрядно по модулю 2 в сумматоре CM_5 с первым 64-разрядным блоком открытых данных

$$\mathrm{T}_{\text{о}}^{(1)} = \left(t_1^{(1)}, t_2^{(1)}, \ldots, t_{63}^{(1)}, t_{64}^{(1)}\right).$$

В результате получают первый 64-разрядный блок шифрованных данных

$$\mathrm{T}_{\text{ш}}^{(1)} = \Gamma_{\text{ш}}^{(1)} \oplus \mathrm{T}_{\text{о}}^{(1)},$$

где $\mathrm{T}_{\text{ш}}^{(1)} = \left(\tau_1^{(1)}, \tau_2^{(1)}, \ldots, \tau_{63}^{(1)}, \tau_{64}^{(1)}\right)$.

Блок шифрованных данных $\mathrm{T}_{\text{ш}}^{(i)}$ одновременно является также исходным состоянием накопителей N_1 и N_2 для выработки второго блока гаммы шифра $\Gamma_{\text{ш}}^{(2)}$, и поэтому по обратной связи $\mathrm{T}_{\text{ш}}^{(1)}$ записывается в указанные накопители N_1 и N_2.

Заполнение накопителя N_1

$$\begin{array}{c}\left(\tau_{32}^{(1)}, \tau_{31}^{(1)}, \ldots, \tau_{2}^{(1)}, \tau_{1}^{(1)}\right)\\ 32,\ \ 31, \ldots,\ 2,\ \ \ 1 \ \leftarrow \text{номер разряда } N_1\end{array}$$

Заполнение накопителя N_2

$$\begin{array}{c}\left(\tau_{64}^{(1)}, \tau_{63}^{(1)}, \ldots, \tau_{34}^{(1)}, \tau_{33}^{(1)}\right)\\ 32,\ \ 31, \ldots,\ 2,\ \ \ 1 \ \leftarrow \text{номер разряда } N_1\end{array}$$

Заполнение накопителей N_1 и N_2 шифруется в режиме простой замены. Полученное в результате шифрования заполнение накопителей N_1 и N_2 образует второй 64-разрядный блок гаммы шифра $\Gamma_{ш}^{(2)}$, который суммируется поразрядно по модулю 2 в сумматоре CM_5 со вторым блоком открытых данных $T_{o}^{(2)}$:

$$T_{ш}^{(2)} = \Gamma_{ш}^{(2)} \oplus T_{o}^{(2)}.$$

Выработка последующих блоков гаммы шифра $\Gamma_{ш}$ и шифрование соответствующих блоков открытых данных $T_{o}^{(i)}$ ($i = 3, \ldots, m$) производятся аналогично. Если длина последнего m-го блока открытых данных $T_{o}^{(m)}$ меньше 64 разрядов, то из $\Gamma_{ш}^{(m)}$ используется только соответствующее число разрядов гаммы шифра, остальные разряды отбрасываются.

В канал связи или память ЭВМ передаются синхропосылка $\tilde{S}$ и блоки зашифрованных данных $T_{ш}^{(1)}, T_{ш}^{(2)}, \ldots, T_{ш}^{(m)}$.

Расшифровывание в режиме гаммирования с обратной связью. При расшифровывании криптосхема имеет тот же вид, что и при шифровании (см. рис. 3.3).

Уравнения расшифровывания:

$$T_{o}^{(1)} = A(\tilde{S}) \oplus T_{ш}^{(1)} = \Gamma_{ш}^{(1)} \oplus T_{ш}^{(1)},$$

$$T_{o}^{(i)} = \Gamma_{ш}^{(i)} \oplus T_{ш}^{(i)} = A\left(T_{ш}^{(i-1)}\right) \oplus T_{ш}^{(i)},\ i = 2, \ldots, m.$$

Реализация процедуры расшифрования шифрованных данных в режиме гаммирования с обратной связью происходит следующим образом. В КЗУ вводят 256 бит того же ключа, на котором осуществлялось шифрование открытых блоков $T_o^{(1)}, T_o^{(2)}, \ldots, T_o^{(m)}$. В накопители N_1 и N_2 вводится синхропосылка $\tilde{S}$, заполнение которорй шифруется в режиме простой замены. Полученное в результате шифрования заполнение N_1 и N_2 образует первый блок гаммы шифра

$$\Gamma_ш^{(1)} = A(\tilde{S}),$$

который суммируется поразрядно по модулю 2 в сумматоре CM_5 с блоком шифрованных данных $T_ш^{(1)}$. Таким образом получается первый блок открытых данных

$$T_o^{(2)} = \Gamma_ш^{(2)} \oplus T_ш^{(2)}.$$

Блок шифрованных данных $T_ш^{(1)}$ является исходным заполнением накопителей N_1 и N_2 для выработки второго блока гаммы шифра $\Gamma_ш^{(2)}$: $\Gamma_ш^{(2)} = A(T_ш^{(1)})$. Полученное заполнение накопителей N_1 и N_2 шифруется в режиме простой замены. Образованный в результате шифрования блок $\Gamma_ш^{(2)}$ суммируется поразрядно по модулю 2 в сумматоре CM_5 со вторым блоком шифрованных данных $T_ш^{(2)}$. В результате получают второй блок открытых данных. Аналогично в N_1 и N_2 записывают блоки шифрованных данных $T_ш^{(2)}, T_ш^{(3)}, \ldots, T_ш^{(m)}$, из которых в режиме простой замены вырабатываются блоки гаммы шифра $\Gamma_ш^{(3)}, \Gamma_ш^{(4)}, \ldots, \Gamma_ш^{(m)}$. Блоки гаммы шифра суммируются поразрядно по модулю 2 в сумматоре CM_5 с блоками шифрованных данных $T_ш^{(3)}, T_ш^{(4)}, \ldots, T_ш^{(m)}$.

В результате получают блоки открытых данных $T_o^{(3)}, T_o^{(4)}, \ldots, T_o^{(m)}$, при этом последний блок открытых данных $T_o^{(m)}$ может содержать меньше 64 разрядов.

3.4. Режим выработки имитовставки

Имитовставка – это блок из P бит, который вырабатывают по определенному правилу из открытых данных с использованием ключа и затем добавляют к зашифрованным данным для обеспечения их имитозащиты.

Имитозащита – это защита системы шифрованной связи от навязывания ложных данных.

В ГОСТ 28147–89 определяется процесс выработки имитовставки, который единообразен для любого из режимов шифрования данных. Имитовставка $И_P$ вырабатывается из блоков открытых данных либо перед шифрованием всего сообщения, либо параллельно с шифрованием по блокам. Первые блоки открытых данных, которые участвуют в выработке имитовставки, могут содержать служебную информацию (например, адресную часть, время, синхропосылку) и не шифруются.

Значение параметра P (число двоичных разрядов в имитовставке) определяется криптографическими требованиями с учетом того, что вероятность навязывания ложных помех равна $1/2^{\partial}$.

Для выработки имитовставки открытые данные представляют в виде последовательности 64-разрядных блоков $T_o^{(i)}$, $i = 1, \ldots, m$. Первый блок открытых данных $T_o^{(1)}$ подвергают преобразованию $\widetilde{A}(*)$, соответствующему первым 16 циклам алгоритма шифрования в режиме простой замены. В качестве ключа для выработки имитовставки используют ключ длиной 256 бит, по которому шифруют данные.

Полученное после 16 циклов 64-разрядное число $\widetilde{A}(T_o^{(1)})$ суммируют по модулю 2 со вторым блоком открытых данных $T_o^{(2)}$. Результат суммирования $\widetilde{A}(T_o^{(1)}) \oplus T_o^{(2)}$ снова подвергают преобразованию $\widetilde{A}(*)$.

Полученное 64-разрядное число $\widetilde{A}(\widetilde{A}(T_o^{(1)}) \oplus T_o^{(2)})$ суммируют по модулю 2 с третьим блоком $T_o^{(3)}$ и снова подвергают преобразованию $\widetilde{A}(*)$, получая 64-разрядное число $\widetilde{A}(\widetilde{A}(\widetilde{A}(T_o^{(1)}) \oplus T_o^{(2)}) \oplus T_o^{(3)})$, и т. д.

Последний блок $T_o^{(m)}$ (при необходимости дополненный нулями до полного 64-разрядного блока) суммируют по модулю 2 с результатом вычислений на шаге $(m-1)$, после чего шифруют в режиме простой замены, используя преобразование $\widetilde{A}(*)$.

Из полученного 64-разрядного числа выбирают отрезок $И_P$ (имитовставку) длиной P бит:

$$И_P = [a^m_{32-P+1}(16), a^m_{32-P+2}(16), \dots, a^m_{32}(16)],$$

где a^m_i – i-й бит 64-разрядного числа, полученного после 16-го цикла последнего преобразования $\widetilde{A}(*)$, $32 - P + 1 \le i \le 31$.

Имитовставка $И_P$ передается по каналу связи в конце шифрованных данных, т. е.

$$Т^{(1)}_{ш}, Т^{(2)}_{ш}, \dots, Т^{(m)}_{ш}, И_P.$$

Поступившие к получателю шифрованные данные $Т^{(1)}_{ш}, Т^{(2)}_{ш}, \dots, Т^{(m)}_{ш}$ расшифровываются, и из полученных блоков открытых данных $Т^{(1)}_{о}, Т^{(2)}_{о}, \dots, Т^{(m)}_{о}$ аналогичным образом вырабатывается имитовставка $И'_P$, которая сравнивается с $И_P$. В случае несовпадения блок открытых данных считается ложным.

4. СТАНДАРТ ШИФРОВАНИЯ ДАННЫХ DES

4.1. Обобщенная схема алгоритма DES

Алгоритм DES использует комбинацию подстановок и перестановок. DES осуществляет шифрование 64-битовых блоков данных с помощью 64-битового ключа, в котором значащими являются 56 бит, а остальные 8 бит – проверочные биты для контроля на четность. Расшифровывание в DES является операцией, обратной шифрованию, и выполняется путем повторения операций шифрования в обратной последовательности.

Обобщенная схема процесса представлена в виде рис. 4.1.

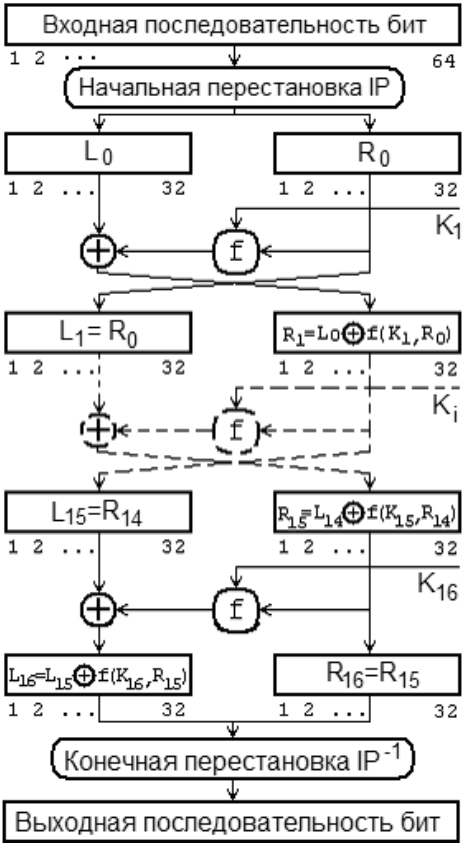

Рис. 4.1. Обобщенная схема шифрования в алгоритме DES

Шифрование в алгоритме DES заключается в начальной перестановке бит 64-битового блока, шестнадцати циклах шифрования и, наконец, в конечной перестановке бит.

Следует отметить, что все приводимые таблицы являются стандартными и должны включаться в реализацию алгоритма DES в неизменном виде. Все перестановки и коды в таблицах подобраны разработчиками таким образом, чтобы максимально затруднить процесс взлома шифра.

Пусть из файла исходного текста считан очередной 64-битовый блок T_o. Этот блок преобразуется с помощью матрицы начальной перестановки IP (табл. 4.1).

Таблица 4.1

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

Биты входного блока T (64 бита) переставляются в соответствии с матрицей IP: бит 58 входного блока T становится битом 1, бит 50 – битом 2 и т. д. Эту перестановку можно описать выражением $T_o = IP(T)$. Полученная последовательность бит T разделяется на две последовательности: L_0 – левые, или старшие, биты, R_0 – правые, или младшие, биты – каждая из которых содержит 32 бита.

Затем выполняется итеративный процесс шифрования, состоящий из 16 шагов (циклов). Пусть T_i – результат *i*-й итерации:

$$T_i = L_i R_i,$$

где $L_i = t_1, t_2, \ldots, t_{32}$ (первые 32 бита);

$R_i = t_{33}, t_{34}, \ldots, t_{64}$ (последние 32 бита). Тогда результат *i*-й итерации описывается следующими формулами:

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

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

Функция f называется функцией шифрования. Ее аргументами являются последовательность R_{i-1}, получаемая на предыдущем шаге итерации, и 48-битовый ключ K_i, который является результатом преобразования 64-битового ключа шифра K. Подробнее функция шифрования f и алгоритм получения ключа K описаны ниже.

На последнем шаге итерации получают последовательности R_{16} и L_{16} (без перестановки местами), которые конкатенируются в 64-битовую последовательность $R_{16}L_{16}$.

По окончании шифрования осуществляется восстановление позиций бит с помощью матрицы обратной перестановки IP^{-1} (табл. 4.2).

Таблица 4.2

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

Процесс расшифровывания данных является инверсным по отношению к процессу шифрования. 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 бит $R_{16}L_{16}$ выполняются те же действия, что и в процессе шифрования, но в обратном порядке.

Итеративный процесс расшифровывания может быть описан следующими формулами:

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

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

Таким образом, для процесса расшифровывания с переставленным входным блоком $R_{16}L_{16}$ на первой итерации используется ключ K_{16}, на второй – K_{15}, а на 16-й – ключ K_1. 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. Результат такого преобразования – исходная последовательность бит (расшифрованное 64-битовое значение).

4.2. Реализация функции шифрования

Схема вычисления функции шифрования $f(R_{i-1},.K_i)$ показана на рис. 4.2.

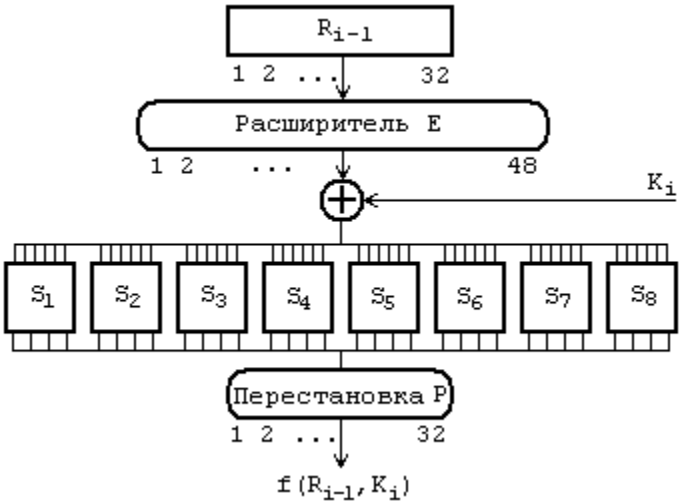

Рис. 5.2. Схема вычисления функции шифрования *f*

Для вычисления значения функции *f* используются:

– функция *E* (расширение 32 бит до 48);

– функция S_1, S_2, ... , S_8 (преобразование 6-битового числа в 4-битовое);

– функция P (перестановка бит в 32-битовой последовательности).

Приведем определения этих функций.

Аргументами функции шифрования f являются R_{i-1} (32 бита) и K_i (48 бит). Результат функции $E(R_{i-1})$ есть 48-битовое число. Функция расширения E, выполняющая расширение 32 бит до 48 (принимает блок из 32 бит и порождает блок из 48 бит), определяется табл. 4.3.

Таблица 4.3

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

В соответствии с табл. 4.3 первые три бита $E(R_{i-1})$ – это биты 32, 1 и 2, а последние – 31, 32 и 1. Полученный результат (обозначим его $E(R_{i-1})$) складывается по модулю 2 с текущим значением ключа K_i и затем разбивается на восемь 6-битовых блоков $B_1, B_2, \ldots, B_8 = E(R_{i-1}) \oplus K_i$. Далее каждый из этих блоков используется как номер элемента в функциях-матрицах $S_1, S_2, \ldots, S_8$, содержащих 4-бито-вые значения (табл. 4.4).

Следует отметить, что выбор элемента в матрице S осуществляется достаточно оригинальным образом. Пусть на вход матрицы S поступает 6-битовый блок $B_j = b_1b_2b_3b_4b_5b_6$, тогда 2-битовое число b_1b_6 указывает номер строки матрицы, а 4-битовое число $b_2b_3b_4b_5$ – номер столбца. Например, если на вход матрицы S_1 поступает 6-битовый блок $B_1 = b_1b_2b_3b_4b_5b_6 = 100110_{(2)}$, то 2-битовое число $b_1b_6 = 10_{(2)} = 2_{(2)}$ указывает строку с номером 2 матрицы S_1, а 4-битовое число $b_2b_3b_4b_5 = 0011_{(2)} = 3_{(10)}$ указывает столбец с номе-

ром 3 матрицы S_1. Это означает, что в матрице S_1 блок $B_1 = 100110$ выбирает элемент на пересечении строки с номером 2 и столбца с номером 3, т. е. элемент $S_{(10)} = 1000_{(2)}$. Совокупность 6-битовых блоков $B_1, B_2, \ldots, B_8$ обеспечивает выбор 4-битового элемента в каждой из матриц $S_1, S_2, \ldots, S_8$.

Таблица 4.4

Функции S																	
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S_1	0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	2	4	1	4	8	13	6	2	11	15	12	9	7	3	10	5	0
	3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2	0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3	0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4	0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5	0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6	0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	1	6
	3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7	0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8	0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

В результате получаем $S_1(B_1)$, $S_2(B_1)$, … , $S_8(B_1)$, т. е. 32-битовый блок (т. к. матрицы S содержат 4-битовые элементы). Этот 32-битовый блок преобразуется с помощью функции перестановки бит P (табл. 4.5).

Таблица 4.5

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

Таким образом, функция шифрования

$$f\left(R_{i-1}, K_i\right) = P\left(S_1(B_1), \ldots, S_8(B_1)\right).$$

4.3. Алгоритм вычисления ключей

Из вышесказанного видно, что на каждой итерации используется новое значение ключа K_i (длиной 48 бит). Новое значение ключа K_i вычисляется из начального ключа K (рис. 4.3).

Ключ K представляет собой 64-битовый блок с 8 битами контроля по четности, расположенными в позициях 8, 16, 24, 32, 40, 48, 56, 64. Для удаления контрольных бит и подготовки ключа к работе используется функция G первоначальной подготовки ключа (табл. 4.6).

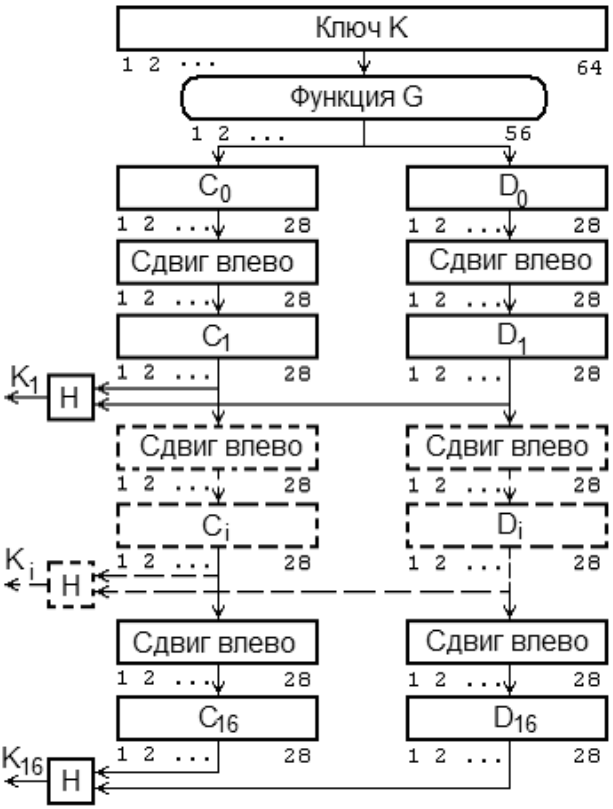

Рис. 4.3. Схема алгоритма вычисления ключей K_i

Таблица 4.6

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

Таблица разделена на две части. Результат преобразования $G(K)$ разбивается на две половины C_0 и D_0, по 28 бит каждая. Первые четыре строки матрицы G определяют, как выбираются биты последовательности C (первым битом C_0 будет бит 57 ключа шифра, затем бит 49 и т. д., а последними битами – биты 44 и 36 ключа).

Следующие четыре строки матрицы G определяют, как выбираются биты последовательности D_0, т. е. последовательность D_0 будет состоять из бит 63, 55, 47, ... , 12, 4 ключа шифра.

Как видно из таблицы, для генерации последовательностей C_0 и D_0 не используются биты 8, 16, 24, 32, 40, 48, 56 и 64 ключа шифра. Эти биты не влияют на шифрование и могут служить для других целей (например, для контроля по четности). Таким образом, в действительности ключ шифра является 56-битовым.

После определения C_0 и D_0 рекурсивно определяются C_i и D_i, $i = 1, 2, \ldots, 16$. Для этого применяются операции циклического сдвига влево на один или два бита в зависимости от номера шага итерации, как показано в табл. 4.7.

Таблица 4.7

Таблица сдвигов для вычисления ключа

Итерация	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Сдвиг влево	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Операции сдвига выполняются для последовательностей C_i и D_i независимо. Например, последовательность C_3 получается посредством циклического сдвига влево на две позиции последовательности C_2, а последовательность D_3 – посредством сдвига влево на две позиции последовательности D_2, C_{16} и D_{16} получаются из C_{15} и D_{15} посредством сдвига влево на одну позицию.

Ключ K_i, определяемый на каждом шаге итерации, есть результат выбора конкретных бит из 56-битовой последовательности C_iD_i и их перестановки. Другими словами, ключ $K_i = H(C_iD_i)$, где функция H определяется матрицей, завершающей обработку ключа (табл. 4.8).

Таблица 4.8

Функция H					
14	17	11	24	1	5
3	28	15	6	21	10
23	19	22	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Как видно из таблицы, первым битом ключа K_i будет 14-й бит последовательности C_iD_i, вторым – 17-й бит, 47-м – 29-й бит C_iD_i, а 48-м – 32-й бит C_iD_i.

4.4. Основные режимы работы алгоритма DES

Чтобы воспользоваться алгоритмом DES для решения разнообразных криптографических задач, разработаны четыре рабочих режима:

1) электронная кодовая книга ECB(Electronic Code Book);
2) сцепление блоков шифра CBC (Cipher Block Chaining);
3) обратная связь по шифротексту CBP (Cipher FeedBack);
4) обратная связь по выходу OFB (Output FeedBack).

Режим «Электронная кодовая книга». Длинный файл разбивают на 64-битовые отрезки (блоки) по 8 байт. Каждый из этих блоков шифруют независимо с использованием одного и того же ключа шифрования (рис. 4.4).

Основное достоинство – простота реализации. Недостаток – относительно слабая устойчивость против квалифицированных криптоаналитиков. Из-за фиксированного характера шифрования при ограниченной длине блока возможно проведение криптоанализа «со словарем». Блок такого размера может повториться в сообщении вследствие большой избыточности в тексте на естественном языке. Это приво-

дит к тому, что идентичные блоки открытого текста в сообщении будут представлены идентичными блоками шифротекста, что дает криптоаналитику некоторую информацию о содержании сообщения.

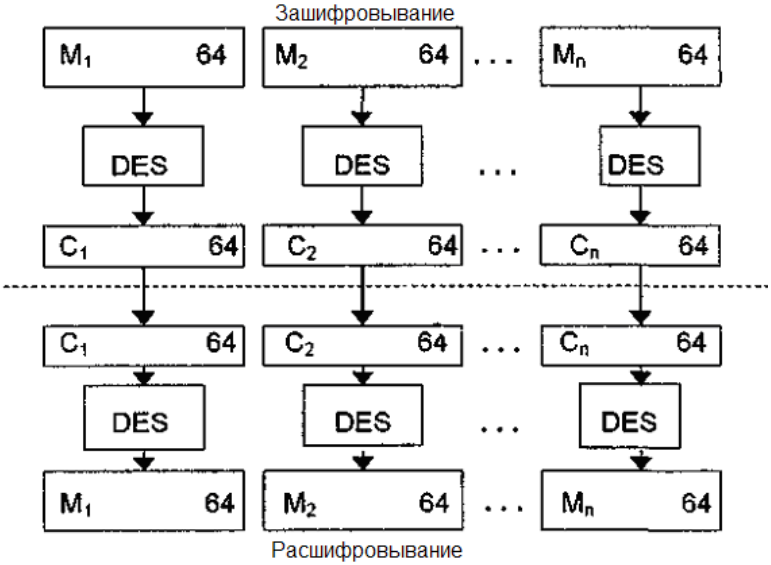

Рис. 4.4. Схема алгоритма DES в режиме электронной кодовой книги

Режим «Сцепление блоков шифра». 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 (рис. 4.5). 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.

Таким образом, для всех $i = 1, \ldots, n$ (n – число блоков) результат шифрования C определяется следующим образом: $C_i = \mathrm{DES}(M_i \oplus C_{i-1})$, где $C_0 = \mathrm{IV}$ – начальное значение шифра, равное начальному вектору (вектору инициализации).

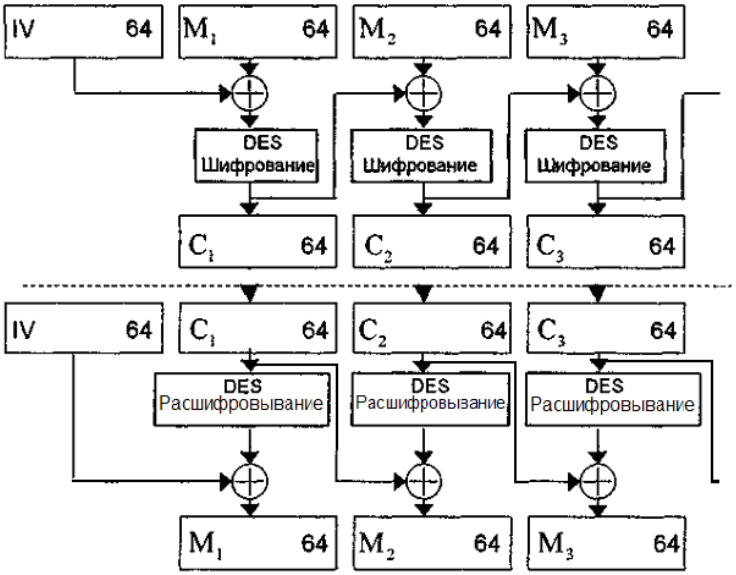

Рис. 4.5. Схема алгоритма DES в режиме сцепления блоков шифра

Очевидно, что последний 64-битовый блок шифротекста является функцией секретного ключа, начального вектора и каждого бита открытого текста независимо от его длины. Этот блок шифротекста называют *кодом аутентификации сообщения* (КАС).

Код КАС может быть легко проверен получателем, владеющим секретным ключом и начальным вектором, путем повторения процедуры, выполненной отправителем. Посторонний не может осуществить генерацию КАС, который воспринялся бы получателем как подлинный, чтобы добавить его к ложному сообщению либо отделить КАС от истинного сообщения для использования его с измененным или ложным сообщением.

Достоинство данного режима в том, что он не позволяет накапливаться ошибкам при передаче.

Блок M_i является функцией C_{i-1} и C_i, поэтому ошибка при передаче приведет к потере только двух блоков исходного текста.

Режим «Обратная связь по шифротексту». В этом режиме размер блока может отличаться от 64 бит (рис. 4.6). Файл, подле-

жащий шифрованию (расшифровыванию), считывается последовательными блоками длиной k бит (k = 1, ... , 64).

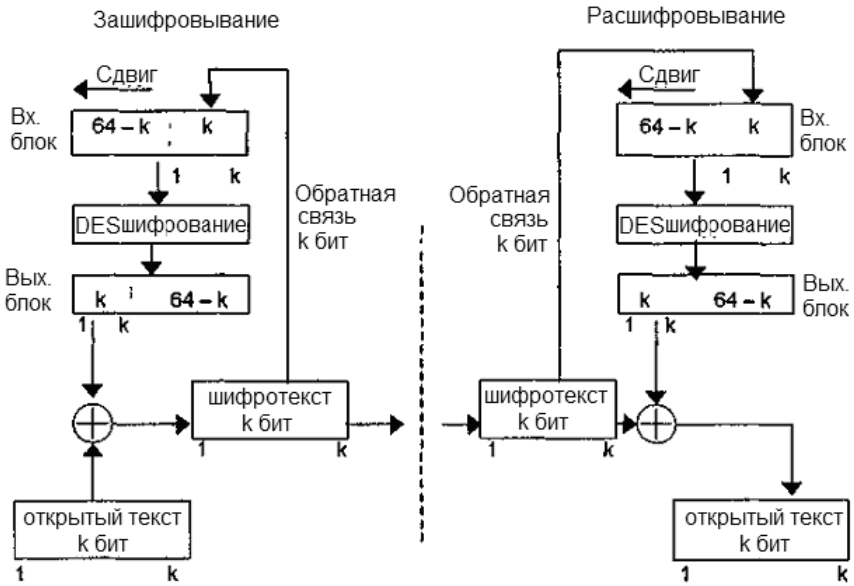

Рис. 4.6. Схема алгоритма DES в режиме обратной связи по шифротексту

Входной блок (64-битовый регистр сдвига) вначале содержит вектор инициализации, выровненный по правому краю. Предположим, что в результате разбиения на блоки получили n блоков длиной k бит каждый (остаток дописывается нулями или пробелами). Тогда для любого $i = 1, \ldots, n$ блок шифротекста $C_i = M_i \oplus P_{i-1}$, где P_{i-1} обозначает k старших бит предыдущего зашифрованного блока.

Обновление сдвигового регистра осуществляется путем удаления его старших k бит и записи C_i в регистр. Восстановление зашифрованных данных также выполняется относительно просто: P_{i-1} и C_i вычисляются аналогичным образом и $M_i = C_i \oplus P_{i-1}$.

Режим «Обратная связь по выходу». Этот режим тоже использует переменный размер блока и сдвиговый регистр, инициализируемый так же, как в режиме CBP, – входной блок вначале содержит

вектор инициализации IV, выровненный по правому краю (рис. 4.7). При этом для каждого сеанса шифрования данных необходимо использовать новое начальное состояние регистра, которое должно пересылаться по каналу открытым текстом.

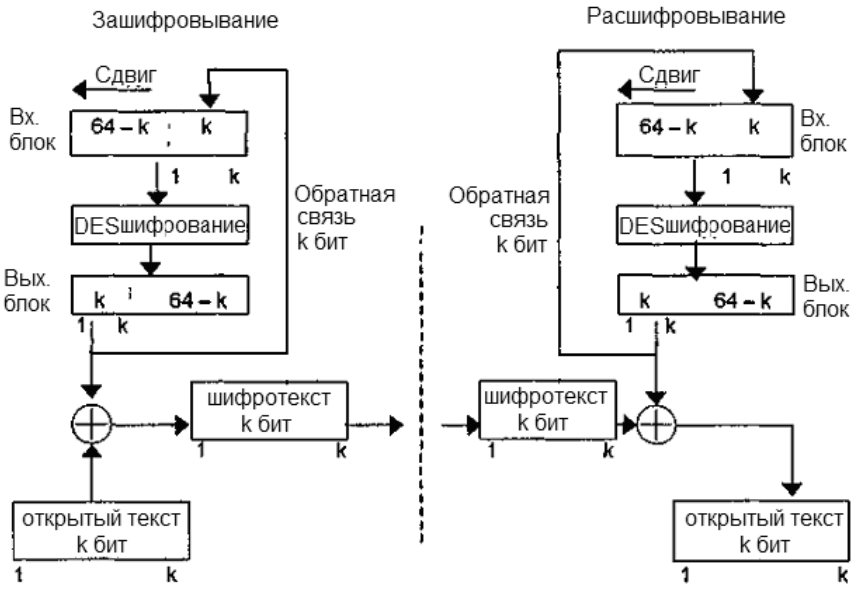

Рис. 4.7. Схема алгоритма DES в режиме обратной связи по выходу

Предположим $M = M_1 M_2 \ldots M_n$ для всех $i = 1, \ldots, n$ $C_i = M_i \oplus P_i$, где P_i – старшие k бит операции $\mathrm{DES}(C_{i-1})$. Отличие от режима обратной связи по шифротексту состоит в методе обновления сдвигового регистра.

Это осуществляется путем отбрасывания старших k бит и дописывания справа P_i.

5. АСИММЕТРИЧНЫЕ КРИПТОСИСТЕМЫ

5.1. Концепция криптосистемы с открытым ключом

Эффективными системами криптографической защиты данных являются асимметричные криптосистемы, называемые также криптосистемами с открытым ключом. В таких системах для шифрования данных используется один ключ, а для расшифровывания другой (отсюда и название – асимметричные). Первый ключ является открытым и может быть опубликован для использования всеми пользователями системы, которые шифруют данные. Расшифровывание данных с помощью открытого ключа невозможно. Для этого используется второй ключ, который является секретным. Разумеется, ключ расшифровывания не может быть определен из ключа шифрования.

Обобщенная схема асимметричной криптосистемы с открытым ключом показана на рис. 5.1.

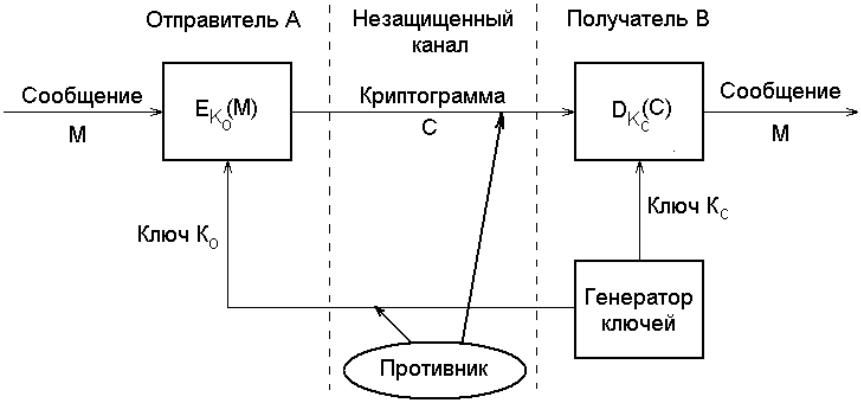

Рис. 5.1. Обобщенная схема асимметричной криптосистемы с открытым ключом

В этой криптосистеме применяют два различных ключа: K_o – открытый ключ отправителя А; K_c – секретный ключ получателя В. Генератор ключа целесообразно располагать на стороне получателя В, чтобы не пересылать секретный ключ K_c по незащищенному каналу. Значения ключей K_o, K_c – зависят от начального состояния генератора ключей.

Раскрытие секретного ключа K_c по известному ключу K_o должно быть вычислительно неразрешимой задачей.

Характерные особенности асимметричных криптосистем:

1) открытый ключ K_o и криптограмма C могут быть отправлены по незащищенному каналу, т. е. могут стать известны противнику;

2) алгоритмы шифрования $E_{K_o}(M) \to C$ и расшифровывания $D_{K_c}(C) \to M$ являются открытыми;

3) защита информации в асимметричной криптосистеме основана на секретности ключа K_c.

У. Диффи и М. Хелман сформулировали требования, выполнение которых обеспечивает безопасность асимметричной криптосистемы:

1) вычисление пары ключей (K_o, K_c) получателем В на основе начального условия должно быть простым;

2) отправитель А, зная открытый ключ K_o и сообщение M, может легко вычислить криптограмму $C = E_{K_o}(M)$;

3) получатель В, используя секретный ключ K_c и криптограмму C, может легко восстановить исходное сообщение $M = D_{K_c}(C) = D_{K_c}(E_{K_o}(M))$;

4) противник, зная открытый ключ K_c, при попытке вычислить секретный ключ K_c наталкивается на непреодолимую вычислительную проблему;

5) противник, зная пару (K_o, C), при попытке вычислить исходное сообщение M, наталкивается на непреодолимую вычислительную проблему.

5.2. Однонаправленные функции

Концепция асимметричных криптосистем с открытым ключом основана на применении однонаправленных функций, пример которых приведен ниже.

Пусть X и Y – некоторые производные множества. Функция $f: X \to Y$ является однонаправленной, если для всех $x \in X$ можно легко вычислить функцию $y = f(x)$, где $y \in Y$. И в тоже время для большинства $y \in Y$ достаточно сложно получить значение $x \in X$, такое, что $f(x) = y$. При этом полагаем, что существует по крайней мере одно такое значение x.

Основным критерием отнесения функции f к классу ОФ является отсутствие эффективных алгоритмов обратного преобразования $Y \to X$. Примеры однонаправленных функций.

1. *Целочисленное произведение двух больших чисел.*

Прямое преобразование – вычисление произведения двух больших чисел P и Q, т. е. нахождение значения $N = P \cdot Q$, является относительно несложной задачей для ЭВМ. Обратное преобразование – разложение на множители большого целого числа, т. е. нахождение деталей P и Q большого $N = P \cdot Q$, является практически неразрешимой задачей при достаточно больших значениях N. По оценке теории чисел для разложения целого $N \approx 2^{664}$ потребуется около 10^{23} операций, т. е. задача практически неразрешима для ЭВМ.

2. *Модульная экспонента с фиксированным основанием и модулем.*

Пусть A и N – целые числа, такие, что $1 \le A \le N$. Модульная экспонента с основанием A по модулю N представляет собой функцию $f_{A,N}(x) = A^x \bmod N$, где x – целое число, $1 \le x \le N-1$. Существуют эффективные алгоритмы, позволяющие достаточно быстро вычислить значение функции $f_{A,N}(x)$. Обратное преобразование, т. е. нахождение x из соотношения $A^x \bmod N = Y$, представляет собой трудновыполнимую задачу, т. к. для $y = A^x$ существует обратная функция $x = \log_A Y$. Часто нахождение аргумента x по известным y, A и N называют *задачей дискретного логарифмирования*. Следует иметь в виду, что $y \in Z_n$, где $Z_n = \{1, 2, \ldots, N-1\}$.

При целых числах порядка $A \approx 2^{664}$ и $N \approx 2^{664}$ решением задачи дискретного логарифмирования (нахождение показателя степени x для известного y) потребуется 10^{26} операций, т. е. для модульной экспоненты на 10^3 сложнее вычислять обратное преобразование, чем для целочисленного произведения. Однако до сих пор не доказано, что не существует эффективного логарифма за приемлемое время. Тем не менее модульная экспонента отнесена к однонаправленным функциям, условно и широко используется на практике.

Кроме однонаправленных функций рассмотренного типа применяются однонаправленные функции с секретом (потайным ходом). Функция $f: X \to Y$ относится к классу ОФ с секретом в том случае, если она является однонаправленной и, кроме того, возможно эффек-

тивное вычисление обратной функции, если известен «потайной ход» (секретное число, строка или другая информация о данной функции).

5.3. Элементы теории чисел

Определения

Число a называется простым, если оно не имеет других натуральных делителей, кроме 1 и a.

Например, 17, 23.

Числа a и b называются взаимно простыми, если наибольший общий множитель этих чисел $(a, b) = 1$.

Например, 8 и 9.

Модулярная арифметика

В модулярной арифметике все арифметические действия выполняются как в обычной арифметике с учетом того, что получаемые числа не могут превышать некоторой величины, называемой модулем.

Например,

$$(3 + 14) \bmod 12 = 5;$$

$$(3 + 14) \equiv 5 \bmod 12.$$

В общем случае $a \equiv r \bmod n$. Читается a сравнимо с r по модулю n. Это справедливо, если $a = n \cdot k + r$, где $k = 0, 1, 2, \ldots$. Отсюда $r = a - n \cdot k$ называется вычетом числа a по модулю n, $(0 \le r < n)$. Справедливо:

$$(a \pm b) \bmod n = (a \bmod n \pm b \bmod n) \bmod n;$$

$$(a \cdot b) \bmod n = (a \bmod n \cdot b \bmod n) \bmod n;$$

$$a \cdot (b \pm c) \bmod n = ((a \cdot b) \bmod n \pm (a \cdot c) \bmod \mathrm{n}) \bmod n.$$

Использование модулярной арифметики позволяет оперировать с очень большими числами, например, при возведении в степень

$$a^8 \bmod n = \left((a^2 \bmod n)^2 \bmod n\right)^2 \bmod n.$$

Малая теорема Ферма. Если n - простое и НОД $(a, n) = 1$, то $a^{n-1} \equiv 1 \bmod n$.

Функция Эйлера. Количество положительных целых, меньших n, которые взаимно просты с n, определяется с помощью функции Эйлера $\varphi(n)$:

Модуль	n простое	n^2	n^m	$p \cdot q$ (p и q простые)
$\varphi(n)$	$n-1$	$n(n-1)$	$n^{m-1}(n-1)$	$(p-1)\cdot(q-1)$

Обобщение Эйлера малой теоремы Ферма: если НОД $(a, n) = 1$, то $a^{\varphi(n)} \equiv 1 \bmod n$.

Нахождение обратных величин

Если задано уравнение $(a \cdot x) \bmod n = 1$, то величина $a^{-1} \equiv x \bmod n$ называется обратной величиной a по модулю n.

Обратная величина существует, если a и n – взаимно простые числа.

Способы нахождения обратных чисел

1. *Перебором возможных значений*.

Подставляя вместо x числа $1, 2, \ldots, n-1$, добиваемся выполнения исходного уравнения.

Пример 5.1. $(5 \cdot x) \bmod 7 = 1$, $x = (5^{-1}) \bmod 7 = 3$, т. к. $(5 \cdot 3) \bmod 7 = 15 \bmod 7 = 15 \bmod 7 = (15 - 7 \cdot 2) \bmod 7 = 1$.

2. *С помощью функции Эйлера* $\varphi(n)$.

$$(a^{-1}) \bmod n = (a^{\varphi(n)-1}) \bmod n.$$

Пример 5.2.

$$x = (5-1) \bmod 7 = (5^{6-1}) \bmod 7 = ((5^2) \bmod 7 \cdot (5^3) \bmod 7) \bmod 7 = (4 \cdot 6) \bmod 7 = 3.$$

3. С помощью алгоритма Евклида.

Алгоритм Евклида применяется для нахождения НОД чисел a и b. Однако его расширенный вариант можно использовать и для вычисления обратной величины.

Основной вариант. Даны a и b, $a > b$. Алгоритм имеет итерационный характер:

$$a = b \cdot q_1 + r_1,\ 0 < r_1 < b;$$

$$b = r_1 \cdot q_2 + r_2,\ 0 < r_2 < r_1;$$

$$r_1 = r_2 \cdot q_3 + r_3,\ 0 < r_3 < r_2;$$

$$\vdots$$

$$r_{k-2} = r_{k-1} \cdot q_k + r_k,\ 0 < r_k < r_{k-1};$$

$$r_{k-1} = r_k \cdot q_{k+1},\ r_{k+1} = 0;$$

$$(a, b) = r_k,$$

где q_i, r_i – частное и остаток на i-м шаге алгоритма. На первом шаге делимое – a, делитель – b, частное – q_1, остаток – r_1. На i-м, $i > 1$ шаге алгоритма: делимое – делитель $i-1$-го шага, делитель – остаток $i-1$-го шага (r_{i-1}), частное – q_i, остаток – r_i.

Пример 5.3. Пусть $a = 1071$ и $b = 693$. Найти НОД (a, b).

$$1071 = 693 \cdot 1 + 378, \text{ т. е. } q_1 = 1,\ r_1 = 378;$$
$$693 = 378 \cdot 1 + 315, \text{ т. е. } q_2 = 1,\ r_2 = 315;$$
$$378 = 315 \cdot 1 + 63, \text{ т. е. } q_3 = 1,\ r_3 = 63;$$
$$315 = 63 \cdot 5, \text{ т. е. } q_4 = 5,\ r_4 = 0.$$

То есть на четвертом шаге остаток от деления $r_4 = 0$, следовательно, алгоритм останавливается и НОД$(a, b) = 63$.

Доказано, что при неотрицательных a и b можно найти такие целые числа u_1, u_2, u_3, что будет выполняться

$$a \cdot u_1 + b \cdot u_2 = u_3 = (a, b).$$

Если выбрать $b = n$ и a, n – взаимно простые числа, т. е. $(a, n) = 1$, тогда

$$a \cdot u_1 + n \cdot u_2 = 1;$$

$$(a \cdot u_1 + n \cdot u_2) \bmod n = (a \cdot u_1) \bmod n = 1;$$

$$(a^{-1}) \bmod n = u_1 \bmod n\ .$$

То есть для нахождения обратной величины необходимо вычислить $u_1 \bmod n$. Эта задача решается в ходе вычисления НОД(a, n) в соответствии с алгоритмом Евклида. Дополнительно на каждом шаге вычисляются координаты двух векторов:

$$\vec{u} = (u_1, u_2, u_3);\ \vec{v}(v_1, v_2, v_3).$$

Алгоритм вычисления u_1 представлен ниже.

1. Начальные установки:

$\vec{u}_0 = (0, 1, n)$, т. е. $u_1 = 0;\ u_2 = 1;\ u_3 = n$, при этом $a \cdot 0 + b \cdot 1 = n$, т. е. $b = n$;

$\vec{v}_0 = (1, 0, a)$, т. е. $v_1 = 1;\ v_2 = 0;\ v_3 = a$, при этом $a \cdot 1 + n \cdot 0 = a$.

2. Проверяем, выполняется ли $u_3 = 1$, если да, то алгоритм заканчивается.

3. Делим n на a (u_3 на v_3) и определяем $q_1 = \left[\frac{u_3}{v_3}\right]$ и значения векторов $\vec{u}_1 = \vec{v}_0;\ \vec{v}_1 = \vec{u}_0 - q_1 \cdot \vec{v}_0$.

4. Вернуться к шагу 2.

На каждом шаге при расчетах используются результаты предыдущего

$$q_i = \left[\frac{u_3}{v_3}\right]_{i-1},\quad \vec{u}_i = \vec{v}_{i-1},\ \vec{v}_i = \vec{u}_{i-1} - q_i \cdot \vec{v}_{i-1}.$$

При $u_3 = 1$ вычисления заканчиваются $(a^{-1}) \bmod n = u_1 \bmod n$, где u_1 значение u_1, полученное на последнем шаге.

Пример 5.4. Пусть $n = 23$ и $a = 5$. Найти число x, обратное числу a по модулю n, т. е. найти $5^{-1} \bmod 23$.

Используя расширенный алгоритм Евклида, выполним вычисления.

q	u_1	u_2	u_3	v_1	v_2	v_3
–	0	1	$n = 23$	1	0	$a = 5$
4	1	0	5	–4	1	3
1	–4	1	3	5	–1	2
1	5	–1	2	–9	2	1
–	–9	2	1	–	–	–

При $u_1 = -9, u_2 = 2, u_3 = 1$ выполняется уравнение $a \cdot u_1 + n \cdot u_2 = 1$, $a \cdot (-9) + n \cdot 2 = 5 \cdot (-9) + 23 \cdot 2 = 1$ и $a^{-1} \bmod n = 5^{-1} \bmod 23 = (-9) \bmod 23 = 14$. Итак, $x = 5^{-1} \bmod 23 = (-9) \bmod 23 = 14$.

5.4. Криптосистема RSA

Последовательность действий абонентов криптосистемы RSA

Действия получателя криптограммы B

1. В генерирует два произвольных больших простых числа P и Q. Эти числа должны быть примерно одинаковыми, размерностью 100–150 десятичных разрядов. Они должны быть секретными.

2. В вычисляет значение модуля $n = P \cdot Q$ и функции Эйлера $\varphi(n) = (P - 1) \cdot (Q - 1)$ и выбирает значение открытого ключа K_o с соблюдением условий:

$$1 < K_o \le \varphi(n);$$

$$(K_o, \varphi(n)) = 1,$$

т. е. K_o и $\varphi(n)$ должны быть взаимно простыми.

3. В вычисляет значение секретного ключа K_c, используя расширенный алгоритм Евклида:

$$K_c = \left(K_o^{-1}\right) \bmod \varphi(n).$$

4. В посылает А пару чисел n, K_o по открытому каналу.

Действия отправителя криптограммы A

1. Разбивает исходный текст M на блоки M_i, $i = 1, 2, \ldots, m$, т. е. $M = M_1, M_2, \ldots, M_m$. Величина $M_i < n$.

2. Шифрует каждое число M_i по формуле $C_i = (M_i^{K_0}) \bmod n$ и отправляет криптограмму $C = C_1, C_2, \ldots, C_m$.

3. Получатель В, получив криптограмму, расшифровывает каждый блок секретным ключом K_c, $M_i = \left(C_i^{K_c}\right) \bmod n$, и восстанавливает весь текст $M = M_1, M_2, \ldots, M_m$.

Покажем, что при расшифровании восстанавливается исходный текст. Согласно обобщению Эйлером малой теоремы Ферма: если НОД(a, n) = 1, то $a^{\varphi(n)} \equiv 1 \bmod n$ или $a^{\varphi(n)+1} \equiv a \bmod n$. Открытый K_o и закрытый K_c ключи в алгоритме связаны соотношением $K_o \cdot K_c \equiv 1 \bmod \varphi(n)$ или $K_o \cdot K_c = k \cdot \varphi(n) + 1$ для некоторого целого k. Таким образом, процесс шифрования, а затем расшифрования некоторого сообщения M_i выглядит следующим образом:

$$\left(\left(M_i^{K_o}\right) \bmod n\right)^{K_c} \bmod n = \left(M_i^{K_o \cdot K_c}\right) \bmod n = \left(M_i^{k \cdot \varphi(n)+1}\right) \bmod n = M_i.$$

В процессе применения RSA злоумышленник может иметь C_i, K_o, n – и организовать дешифрование двумя способами.

1. По C_i, K_o, n получить M_i. Для этого он решает задачу вычисления M_i из уравнения $C_i = M_i^{K_o} \bmod n$. Эта задача вычислительно трудна.

2. По n вычислить P и Q, затем найти $\varphi(n)$ и вычислить $K_c = \left(K_o^{-1}\right) \bmod \varphi(n)$, и дешифровать сообщение $M_i = \left(C_i^{K_c}\right) \bmod n$.

Однако задача разложения большого числа на простые множители вычислительно сложна.

Пользователи A и B должны быстро вычислять K_o, шифровать и расшифровывать.

Вычисление K_o с использованием алгоритма Евклида – довольно быстрый процесс и не представляет трудности. Шифрование и расшифровывание – возведение большого числа в большую степень – требует определенных затрат времени, но, с учетом наличия быстрых алгоритмов и быстродействия современных компьютеров, это приемлемая процедура.

5.5. Криптосистема Эль-Гамаля

Схема Эль-Гамаля, предложенная в 1985 г., может быть использована как для шифрования, так и для цифровых подписей. Безопасность схемы Эль-Гамаля обусловлена сложностью вычисления дискретных логарифмов в конечном поле.

Для того чтобы генерировать пару ключей (открытый ключ – секретный ключ), сначала выбирают некоторое большое простое

число P и большое целое число G, причем $G < P$. Числа P и G могут быть распространены среди группы пользователей. Затем выбирают случайное целое число X, причем $X < P$. Число X является секретным ключом и должно храниться в секрете. Далее вычисляют $Y = G^X \bmod P$. Число Y является открытым ключом.

Для того чтобы зашифровать сообщение M, выбирают случайное целое число $1 < K < P - 1$ такое, что числа K и $(P - 1)$ являются взаимно простыми. Затем вычисляют числа $a = G^K \bmod P$, $b = \left(Y^K \cdot M\right) \bmod p$. Пара чисел (a, b) является шифротекстом. Заметим, что длина шифротекста вдвое больше длины исходного открытого текста M.

Для того чтобы расшифровать шифротекст (a, b), вычисляют

$$M = (b / a^X) \bmod P.$$

Справедливость этого равенства следует из

$$a^X \equiv G^{KX} \bmod P, \quad b / a^X \equiv Y^K M / a^X \equiv G^{KX} M / G^{KX} \equiv M \bmod P.$$

6. ЭЛЕКТРОННАЯ ЦИФРОВАЯ ПОДПИСЬ

6.1. Общие сведения

При обмене сообщениями через ТКС возникает задача подтверждения их подлинности, т. е. подтверждения авторства и целостности. Такая же проблема существует и при переходе от юридически значимых бумажных документов к электронным. Сообщения, для которых эта проблема актуальна, будут в дальнейшем называться *электронными документами*.

Целью аутентификации электронных документов является их защита от возможных видов злоумышленных действий, к которым относятся:

– *активный перехват* – нарушитель, подключившийся к сети, перехватывает документы и изменяет их;

– *маскарад* – абонент С посылает документ абоненту В от имени абонента А;

– *ренегатство* – абонент А заявляет, что не посылал сообщения абоненту В, хотя на самом деле послал;

– *подмена* – абонент В изменяет или формирует новый документ и заявляет, что получил его от абонента А;

– *повтор* – абонент С повторяет ранее переданный документ, который абонент А посылал абоненту В.

В обычной (бумажной) информатике эти проблемы решаются за счет того, что информация в документе и рукописная подпись автора жестко связаны с физическим носителем (бумагой). В электронных документах на машинных носителях такой связи нет.

Естественно, что для электронных документов традиционные способы установления подлинности по рукописной подписи и оттиску печати на бумажном документе совершенно непригодны, поэтому для подтверждения подлинности документа используется специфическая криптографическая процедура, называемая *электронной цифровой подписью* (ЭЦП).

ЭЦП функционально аналогична обычной рукописной подписи и обладает ее основными достоинствами:

– удостоверяет, что подписанный текст исходит от лица, поставившего подпись;

– не дает этому лицу возможности самому отказаться от обязательств, связанных с подписанным текстом;

– гарантирует целостность подписанного текста.

ЭЦП представляет собой относительно небольшое количество дополнительной цифровой информации, передаваемой вместе с подписываемым текстом.

Технология ЭЦП включает две процедуры:

1) процедуру постановки подписи;

2) процедуру проверки подписи.

В процедуре постановки подписи используется секретный ключ отправителя сообщения, в процедуре проверки подписи – открытый ключ отправителя.

При формировании ЭЦП отправитель прежде всего вычисляет хэш-функцию $h(M)$ подписываемого документа M. Вычисленное значение хэш-функции $h(M)$ представляет собой один короткий блок информации m, характеризующий весь документ M в целом. Затем число m «шифруется» секретным ключом отправителя. Получаемая при этом пара чисел представляет собой ЭЦП для данного документа M. В принципе можно обойтись без предварительного хэширования документа, а «шифровать» весь документ. Однако в этом случае придется иметь дело с гораздо большим по размерам файлом. Употребление слова «шифровать» здесь весьма условное и справедливо при использовании алгоритма RSA, для других алгоритмов точнее говорить «преобразовывать».

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

Принципиальным моментом в системе ЭЦП является невозможность подделки ЭЦП пользователя без знания его секретного ключа подписывания.

В качестве подписываемого документа может быть использован любой файл. Подписанный файл создается из неподписанного путем добавления в него одной или более электронных подписей.

Каждая подпись содержит следующую информацию:

– дату подписи;

– срок окончания действия ключа данной подписи;

– информацию о лице, подписавшем файл (Ф.И.О., должность, краткое наименование фирмы);

– идентификатор подписавшего (имя открытого ключа);

– собственно цифровую подпись.

6.2. Однонаправленные хэш-функции

Хэш-функция предназначена для сжатия подписываемого документа M до нескольких десятков или сотен бит. Хэш-функция h принимает в качестве аргумента сообщение (документ) M произвольной длины и возвращает хэш-значение $h(M) = H$ фиксированной длины. Обычно хэшированная информация является сжатым двоичным представлением основного сообщения произвольной длины. Следует отметить, что значение хэш-функции $h(M)$ зависит от документа M и не позволяет восстановить сам документ M.

Хэш-функция должна удовлетворять целому ряду условий:

– она должна быть чувствительна к всевозможным изменениям в тексте M: вставкам, выбросам, перестановкам и т. п.;

– должна обладать свойством необратимости, т. е. задача подбора документа M', который обладал бы требуемым значением хэш-функции, должна быть вычислительно неразрешима;

– вероятность того, что значения хэш-функции двух различных документов (вне зависимости от их длин) совпадут, должна быть ничтожно мала.

Большинство хэш-функций строится на основе однонаправленной функции f, аргументами которой являются две величины: блок исходного документа M_i и хэш-значение H_{i-1} предыдущего блока документа (рис. 6.1): $H_i = f\left(M_i, H_{i-1}\right)$.

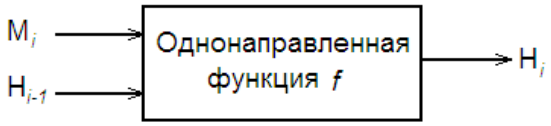

Рис. 6.1. Общая схема вычисления однонаправленной хэш-функции

Хэш-значение, вычисляемое при вводе последнего блока текста, становится хэш-значением всего сообщения M. В результате одно-

направленная хэш-функция всегда формирует выход фиксированной длины n (независимо от длины входного текста).

Часто функции хэширования строят, используя в качестве однонаправленной функции – симметричный блочный алгоритм шифрования (DES, ГОСТ 28147–89) в режиме с обратной связью, принимая последний блок шифротекста за хэш-значение всего документа. Так как длина блока в указанных алгоритмах невелика (64 бита), то часто в качестве хэш-значения используют два блока шифротекста. Одна из возможных схем хэширования на основе блочного алгоритма шифрования изображена на рис. 6.2

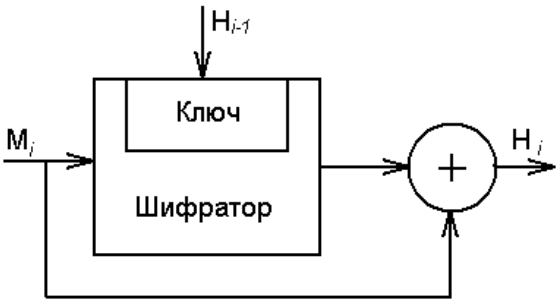

Рис. 6.2. Схема вычисления однонаправленной функции хэширования на базе блочного алгоритма шифрования

6.3. Алгоритм электронной цифровой подписи RSA

Первой и наиболее известной во всем мире конкретной системой ЭЦП стала система RSA, математическая схема которой была разработана в 1977 г. в Массачуссетском технологическом институте США.

Сначала необходимо вычислить пару ключей: секретный и открытый. Для этого отправитель (автор) электронных документов вычисляет два больших простых числа P и Q, затем находит их произведение $n = P \cdot Q$ и значение функции Эйлера $\varphi(n) = (P - 1) \cdot (Q - 1)$. Далее отправитель вычисляет число K_o из условий: $K_o \le \varphi(n)$, НОД $(K_o, \varphi(n)) = 1$ и число K_c из условий $K_c < n$, $K_o \cdot K_c \equiv 1 \bmod (\varphi(n))$.

Пара чисел (K_o, n) является открытым ключом. Эту пару автор передает партнерам по переписке для проверки его цифровых подписей. Число K_c сохраняется автором как секретный ключ для под-

писывания. Обобщенная схема формирования и проверки цифровой подписи RSA показана на рис. 6.3.

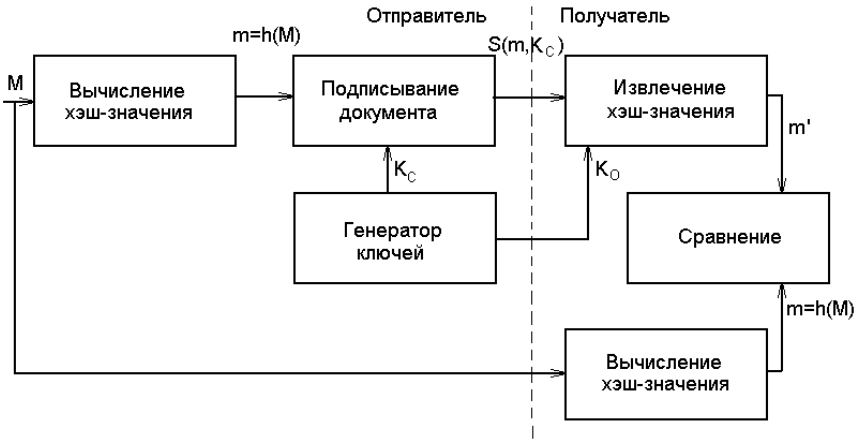

Рис. 6.3. Обобщенная схема алгоритма ЭЦП RSA

Допустим, что отправитель хочет подписать сообщение M перед его отправкой. Сначала сообщение M (блок информации, файл, таблица) сжимают с помощью хэш-функции h в целое число $m = h(M)$. Затем вычисляют цифровую подпись S под электронным документом M, используя хэш-значение m и секретный ключ K_c: $S = m^{K_c} \bmod n$. Пара (M, S) передается партнеру-получателю как электронный документ M, подписанный цифровой подписью S, причем подпись S сформирована обладателем секретного ключа K_c. После приема пары (M, S) получатель вычисляет хэш-значение сообщения M двумя разными способами. Прежде всего он восстанавливает хэш-значение m', применяя криптографическое преобразование подписи S с использованием открытого ключа K_o: $m' = S^{K_o} \bmod n$. Кроме того, он находит результат хэширования принятого сообщения M с помощью такой же хэш-функции h: $m = h(M)$. Если соблюдается равенство вычисленных значений, т. е. $m' = S^{K_o} \bmod n = h(M)$, то получатель признает пару (M, S) подлинной. Доказано, что только обладатель секретного ключа K_c может сформировать цифровую подпись S по документу M, а определить секретное число K_c по открытому числу K_o

не легче, чем разложить модуль N на множители. Кроме того, можно строго математически доказать, что результат проверки цифровой подписи S будет положительным только в том случае, если при вычислении S был использован секретный ключ K_c, соответствующий открытому ключу K_o. Поэтому открытый ключ K_o иногда называют «идентификатором» подписавшего.

Недостатки алгоритма цифровой подписи RSA.

1. При вычислении модуля n, ключей K_c и K_o для системы цифровой подписи RSA необходимо проверять большое количество дополнительных условий, что сложно сделать. Невыполнение любого из условий делает возможным фальсификацию цифровой подписи со стороны того, кто обнаружит такое невыполнение при подписании важных документов, а допускать такую возможность нельзя даже теоретически.

2. Для обеспечения криптостойкости цифровой подписи RSA по отношению к попыткам фальсификации на уровне, например, национального стандарта США на шифрование информации (алгоритм DES), т. е. 10^{18}, необходимо использовать при вычислениях n, K_c и K_o целые числа не менее 2^{512} (или около 10^{154}) каждое, что требует больших вычислительных затрат, превышающих на 20–30 % вычислительные затраты других алгоритмов цифровой подписи при сохранении того же уровня криптостойкости.

3. Цифровая подпись RSA уязвима к так называемой мультипликативной атаке. Алгоритм цифровой подписи RSA позволяет злоумышленнику без знания секретного ключа K_c сформировать подписи под теми документами, у которых результат хэширования можно вычислить как произведение результатов хэширования уже подписанных документов.

Пример 6.1. Допустим, что злоумышленник может сконструировать три сообщения M_1, M_2 и M_3, у которых хэш-значения $m_1 = h(M_1)$, $m_2 = h(M_2)$, $m_3 = h(M_3)$, причем $m_3 = m_1 \cdot m_2 (\mathrm{mod}\, n)$. Допустим также, что для двух сообщений M_1 и M_2 получены законные подписи $S_1 = m_1^{K_c} (\mathrm{mod}\, n)$ и $S_2 = m_2^{K_c} (\mathrm{mod}\, n)$.

Тогда злоумышленник может легко вычислить подпись S_3 для документа M_3, даже не зная секретного ключа

$$S_3 = S_1 \cdot S_2 (\mathrm{mod}\, n).$$

Действительно,

$$S_1 \cdot S_2 (\text{mod } n) = m_1^{K_c} m_2^{K_c} (\text{mod } n) = (m_1 \cdot m_2)^{K_c} (\text{mod } n) =$$
$$= m_3^{K_c} (\text{mod } n) = S_3.$$

6.4. Алгоритм цифровой подписи Эль-Гамаля (EGSA)

Название EGSA происходит от слов El Gamal Signature Algorithm (алгоритм цифровой подписи Эль-Гамаля). Идея EGSA основана на том, что для обоснования практической невозможности фальсификации цифровой подписи может быть использована более сложная вычислительная задача, чем разложение на множители большого целого числа, – задача дискретного логарифмирования. Кроме того, Эль-Гамалю удалось избежать явной слабости алгоритма цифровой подписи RSA, связанной с возможностью подделки цифровой подписи под некоторыми сообщениями без определения секретного ключа.

Для генерации пары ключей (открытого и секретного) сначала выбирают некоторое большое простое целое число P и большое целое число G, причем $G < P$. Отправитель и получатель подписанного документа используют при вычислениях одинаковые большие целые числа P (10^{308} или 2^{1024}) и G (10^{156} или 2^{512}), которые не являются секретными. Отправитель выбирает случайное целое число K_c, $1 < K_c < P - 1$ и вычисляет $K_o = G^{K_c} \text{mod } P$. Число K_o является открытым ключом, используемым для проверки подписи отправителя. Число K_o открыто передается всем потенциальным получателям документов. Число K_c является секретным ключом отправителя для подписывания документов и должно храниться в секрете. Для того чтобы подписать сообщение M, сначала отправитель хэширует его с помощью хэш-функции $h(*)$ в целое число $m = h(M)$, $1 < m < (P - 1)$ и генерирует случайное целое число K, $1 < K < (P - 1)$, такое, что K и $(P - 1)$ являются взаимно простыми. Затем отправитель вычисляет целое число a: $a = G^K \text{mod } P$ и, применяя расширенный алгоритм Евклида, вычисляет с помощью секретного ключа K_c целое число b из уравнения $m = K_c \cdot a + K \cdot b (\text{mod}(P - 1))$.

Пара чисел (a, b) образует цифровую подпись S: $S = (a, b)$, проставляемую под документом M. Тройка чисел (M, a, b) передается получателю, в то время как пара чисел (K_c, K) держится в секрете.

После приема подписанного сообщения (M, a, b) получатель должен проверить, соответствует ли подпись $S = (a, b)$ сообщению M. Для этого получатель сначала вычисляет по принятому сообщению M число $m = h(M)$, т. е. хэширует принятое сообщение M. Затем получатель вычисляет значение $A = K_{\text{о}}^{a} \cdot a^{b} (\text{mod } P)$ и признает сообщение M подлинным, если, и только если $A = G^{m} \text{mod } P$. Иначе говоря, получатель проверяет справедливость соотношения $K_{\text{о}}^{a} \cdot a^{b} (\text{mod } P) = G^{m} (\text{mod } P)$.

Можно строго математически доказать, что последнее равенство будет выполняться тогда, и только тогда, когда подпись $S = (a, b)$ под документом M получена с помощью именно того секретного ключа $K_{\text{с}}$, из которого был получен открытый ключ $K_{\text{о}}$. Таким образом, можно надежно удостовериться, что отправителем сообщения M был обладатель именно данного секретного ключа $K_{\text{с}}$, не раскрывая при этом сам ключ, и что отправитель подписал именно этот конкретный документ M.

Следует отметить, что выполнение каждой подписи по методу Эль-Гамаля требует нового значения K, причем это значение должно выбираться случайным образом. Если нарушитель раскроет когда-либо значение K, повторно используемое отправителем, то он сможет раскрыть секретный ключ $K_{\text{с}}$ отправителя. Следует отметить, что схема Эль-Гамаля является характерным примером подхода, который допускает пересылку сообщения M в открытой форме вместе с присоединенным аутентификатором (a, b). В таких случаях процедура установления подлинности принятого сообщения состоит в проверке соответствия аутентификатора сообщению.

Схема цифровой подписи Эль-Гамаля имеет ряд преимуществ по сравнению со схемой цифровой подписи RSA:

1. При заданном уровне стойкости алгоритма цифровой подписи целые числа, участвующие в вычислениях, имеют запись на 25 % короче, что уменьшает сложность вычислений почти в два раза и позволяет заметно сократить объем используемой памяти.

2. При выборе модуля P достаточно проверить, что это число является простым и что у числа $(P - 1)$ имеется большой простой множитель.

3. Процедура формирования подписи по схеме Эль-Гамаля не позволяет вычислять цифровые подписи под новыми сообщениями без знания секретного ключа (как в RSA).

Однако алгоритм цифровой подписи Эль-Гамаля имеет и некоторые недостатки по сравнению со схемой подписи RSA. В частности, длина цифровой подписи получается в 1,5 раза больше, что, в свою очередь, увеличивает время ее вычисления.

Пример *6.2.* Выберем числа $P = 11$, $G = 2$ и секретный ключ $K_c = 8$. Вычисляем значение открытого ключа $K_o = G^{K_c} \bmod P = 2^8 \bmod 11 = 3$. Предположим, что исходное сообщение M характеризуется хэш-значением $m = 5$. Для того чтобы вычислить цифровую подпись для сообщения M, имеющего хэш-значение $m = 5$, сначала выберем случайное целое число $K = 9$. Убедимся, что числа K и $(P - 1)$ являются взаимно простыми. Действительно, НОД (9, 10) = 1. Далее вычисляем элементы a и b подписи: $a = G^k \bmod P = 2^9 \bmod 11 = 6$, элемент b определяем из уравнения $m = K_c a + kb (\bmod (P - 1))$ используя расширенный алгоритм Евклида. При $m = 5$, $a = 6$, $K_c = 8$, $P = 11$ получаем $5 = 6 \cdot 8 + 9 \cdot b \pmod{10}$ или $9 \cdot b = -43 \pmod{10}$. Решение $b = 3$. Цифровая подпись представляет собой пару $a = 6$, $b = 3$. Далее отправитель передает подписанное сообщение. Приняв подписанное сообщение и открытый ключ $K_o = 3$, получатель вычисляет хэш-значение для сообщения M: $m = 5$, а затем

$$K_0^a \cdot a^b (\bmod P) = 3^6 \cdot 6^3 (\bmod 11) = 10;$$

$$G^m \bmod P = 2^5 \bmod 11 = 10.$$

Так как эти два целых числа равны, принятое получателем сообщение признается подлинным.

6.5. Белорусские стандарты ЭЦП и функции хэширования

Белорусские стандарты, регламентирующие использование электронной цифровой подписи, официальное название которых «Процедура выработки и проверки ЭЦП» и «Функция хэширования», были разработаны группой белорусских специалистов в 1999 г. и официально приняты в 2000 г.

В этих стандартах наряду с элементами классических процедур ЭЦП используются современные идеи, позволяющие увеличить крип-

тостойкость и быстродействие. Так, открытый ключ и секретный ключ связаны известным соотношением

$$K_o = (a^{K_c}) \bmod P,$$

которое позволяет легко вычислить K_o по K_c но очень сложно решение обратной задачи – вычислить K_c по K_o. К подписываемому сообщению добавляется случайная компонента t, что усложняет возможный подбор хэш-значения злоумышленником по известному тексту сообщения.

Обозначения, принятые в стандарте СТБ-1176.02–99

B_p – множество, состоящее из чисел 1, 2, … , $p - 1$;

$c := d$ – присвоение параметру c значения d;

$c \bmod d$ – остаток от деления c на d, где c – натуральное число или ноль; а d – натуральное число;

$c^{-1} \bmod d$ – натуральное число b такое, что $b < d$ и $(cb) \bmod d = 1$, где c и d – взаимно простые числа;

$\lceil c \rceil$ – наименьшее целое число, не меньшее чем c;

$\lfloor c \rfloor$ – наибольшее целое число, не большее чем c;

$c = \sum_{i=0}^{k-1} c_i (2^b)^i$ – разложение неотрицательного целого числа c по основанию 2^b, где k и b – натуральные числа;

c_i – целое число, $0 \le c_i \le 2^b$;

$\oplus$ – бинарная операция, определенная на множестве неотрицательных целых чисел по формуле $d \oplus b = \sum_{i=0}^{k-1} ((d_i + b_i) \bmod 2) 2^i$, где $d = \sum_{i=0}^{k-1} d_i 2^i$, $b = \sum_{i=0}^{k-1} b_i 2^i$, $d_0, \ldots, d_{k-1}, b_0, \ldots, b_{k-1} \in \{0, 1\}$;

$\circ$ – операция $\circ$: $B_p \cdot B_p \to B_p$ определяется для любых $c \in B_p$ и $d \in B_p$ по формуле $c \circ d = (cd(2^{l+2})^{-1}) \bmod p$;

$C^{(k)}$ – степень числа на основе операции $\circ$ определяется индуктивно по формуле $c^{(k)} = \begin{cases} c & , \ k+1 \\ c^{(k-1)} \circ c, & k > 1 \end{cases}$, где k – натуральное число;

h – функция хэширования, процедура вычисления значений которой соответствует СТБ.

Процедура выработки ЭЦП

1. Выбираются параметры l и r, которые определяют уровень криптографической стойкости ЭЦП. Число l является длиной записи числа p в системе счисления по основанию 2, r является длиной записи числа q в системе счисления по основанию 2.

2. В соответствии с выбранными l и r генерируются простые числа p и q такие, что q делит $p - 1$ нацело.

3. Генерируется случайное число d, $0 < d < p$.

4. Вычисляется $a = d^{\left(\frac{p-1}{q}\right)}$. Если $a \equiv 2^{l+2} \bmod p$, то перейти к пункту 3.

5. Генерируется случайное число x, $0 < x < q$, которое является секретным ключом.

6. Вычисляется число $y = a^{(x)}$, которое является открытым ключом.

7. Генерируется случайное число k, $0 < k < q$.

8. Вычисляется $t = a^{(k)}$. Далее число t разлагается по основанию 2^8, т. е. $t = \sum_{i=0}^{n-1} t_i (2^8)^i$. Таким образом, получаются коэффициенты t_0, t_1, … , t_{n-1}.

9. Формируется последовательность $M_t = (t_0, t_1, \ldots, t_{n-1}, m_1, m_2, \ldots, m_z)$, состоящая из коэффициентов $t_0, t_1, \ldots, t_{n-1}$ и блоков открытого текста $m_1, m_2, \ldots, m_z$.

10. Вычисляется значение хэш-функции $U = h(M_t)$. Если $U = 0$, то перейти к пункту 6.

11. Вычисляется $V = (k - x\,U) \bmod q$. Если $V = 0$, то перейти к пункту 6.

12. Вычисляется $S = U \cdot 2^r + V$. ЭЦП последовательности M_t есть число S.

13. Отправляется M_t, S.

Процедура проверки ЭЦП

1. Вычисляется $V = S \bmod 2^r$.

2. Вычисляется $U = (S - V)/2^r$.

3. Если хотя бы одно из условий $0 < U < 2^r$ и $0 < V < q$ не выполнено, то ЭЦП считается недействительной и работа алгоритма завершается.

4. Вычисляется $t' = a^{(V)} \circ y^{(U)}$.

5. Число t' разлагается по основанию 2^8, т. е. $t' = \sum_{i=0}^{n-1} t_i'(2^8)^i$. Таким образом, получаются коэффициенты t_0', t_1', … , t_{n-1}'.

6. Формируется последовательность $M_t' = (t_0', t_1', \dots, t_{n-1}', m_1, m_2, \dots, m_z)$, состоящая из коэффициентов t_0', t_1', … , t_{n-1}' и блоков открытого текста $m_1, m_2, \dots, m_z$.

7. Вычисляется хэш-функция $W = h(M_t')$.

8. Проверяется условие $W = U$. При совпадении W и U принимается решение о том, что ЭЦП была создана при помощи личного ключа подписи x, связанного с открытым ключом проверки подписи y. Таким образом? ЭЦП и последовательность M_t не были изменены с момента их создания. В противном случае подпись считается недействительной.

Стандарт «Процедура выработки и проверки ЭЦП» содержит алгоритмы и процедуры выработки и проверки электронной цифровой подписи, а также подробные инструкции:

- по выбору величин r и l (размер p и q);
- генерации p и q;
- генерации a.

7. АУТЕНТИФИКАЦИЯ ПОЛЬЗОВАТЕЛЕЙ В ТКС

7.1. Общие сведения

Аутентификация означает установление подлинности. Она обеспечивает работу в сети только санкционированных пользователей. Чаще всего проводится при входе в сеть, т. е. после процесса идентификации, во время которого пользователь сообщает свой идентификатор (называет себя), но может – и во время работы. В процедуре аутентификации участвуют две стороны: пользователь доказывает свою подлинность, а сеть проверяет это доказательство и принимает решение. В качестве доказательства используют:

– знание секрета (пароля);

– владение уникальным предметом (физическим ключом);

– предъявление биометрической характеристики (отпечаток пальца, рисунок радужной оболочки глаза, голос).

Наиболее распространенное средство аутентификации – пароль. Он используется как при входе в систему, так и в процессе работы. Пароль может вводиться с клавиатуры, с различных носителей цифровой информации или комбинировано. При использовании паролей необходимо соблюдать требования: по правилам генерации (длина, случайность символов), хранения (хранить в защищенном месте), использования (в зашифрованном виде), отзыва.

В качестве субъектов аутентификации могут выступать не только пользователи, но и различные устройства или процессы. Причем сам процесс может носить обоюдный характер, обе стороны должны доказать свою подлинность. Например, пользователь, обращающийся к корпоративному серверу, должен убедится, что имеет дело с сервером своего предприятия. В этом случае процедура называется *взаимной аутентификацией*.

7.2. Удаленная аутентификация пользователей с использованием пароля

Процесс удаленной аутентификации обычно выполняют в начале сеанса связи. Рассмотрим аутентификацию с использованием пароля.

Пусть стороны А и В знают друг друга и имеют одинаковый секретный ключ K_{AB}. Пользователь А вводит свой идентификатор

(PIN), после чего программа, используя ключ и PIN, вырабатывает пароль *P*, который вместе с PIN передается по сети к пользователю B. Пользователь B по PIN находит в своей базе ключ K_{AB}, с помощью которого вырабатывает *P*. После чего сравнивает значение полученного пароля и выработанного. Схема описанной аутентификации приведена на рис. 7.1.

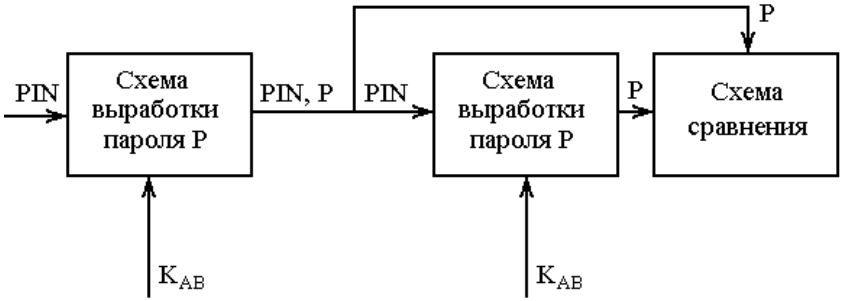

Рис. 7.1. Схема аутентификации с использованием пароля

Рассмотренная схема имеет существенный недостаток: злоумышленник может перехватить пароль *P* и PIN и позднее использовать для своей аутентификации. Для устранения этого недостатка применяют механизм отметки времени («временной штемпель»). Его суть заключается в том, что при выработке пароля наряду с ключом используется текущее время в виде некоторого интервала, в пределах которого пароль действителен. Аналогично вырабатывается пароль на стороне B, в этом случае устаревшим паролем нельзя воспользоваться.

7.3. **Удаленная аутентификация пользователей с использованием механизма запроса-ответа**

Если пользователь A хочет быть уверен, что сообщения, получаемые им от пользователя B, не являются ложными, он включает в посылаемое для B сообщение непредсказуемый элемент – запрос *x* (например, некоторое случайное число). При ответе пользователь B должен выполнить над этим элементом некоторую операцию (например, вычислить функцию $f(x)$). Это невозможно осуществить заранее, так как пользователю B неизвестно, какое случайное число

x придет в запросе. Получив ответ с результатом действий В, пользователь А может быть уверен, что В – подлинный. Недостаток этого метода – возможность установления закономерности между запросом и ответом.

Механизм запрос-ответ используется в более сложной процедуре аутентификации – «*рукопожатии*».

Процедура базируется на указанном выше механизме и заключается во взаимной проверке ключей, используемых сторонами. Иначе говоря, стороны признают друг друга законными партнерами, если докажут, что обладают правильными ключами. Процедуру «рукопожатия» обычно применяют в компьютерных сетях при организации сеанса связи между пользователями, пользователем и хост-компьютером, между хост-компьютерами и т. д.

Рассмотрим в качестве примера процедуру «рукопожатия» для двух пользователей А и В. Это допущение не влияет на общность рассмотрения. Такая же процедура используется, когда вступающие в связь стороны не являются пользователями. Пусть применяется симметричная криптосистема. Пользователи А и В разделяют один секретный ключ K_{AB}. Вся процедура показана на рис. 7.2.

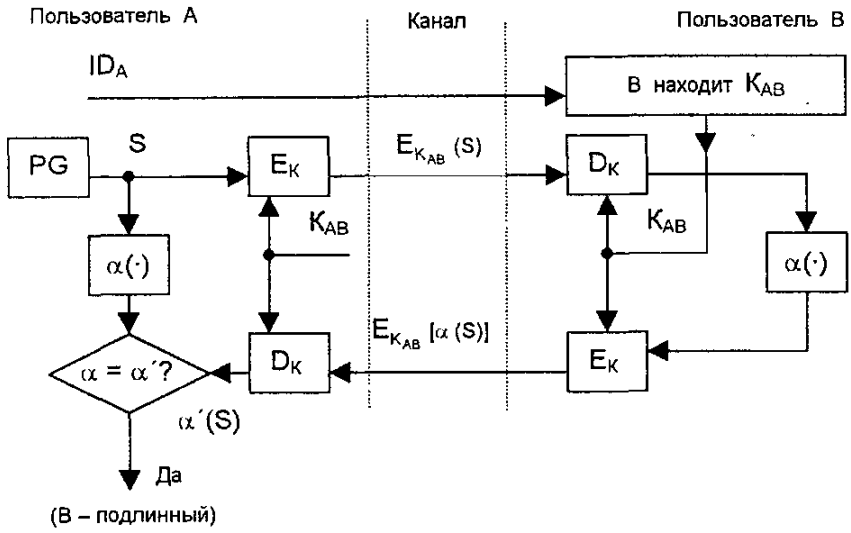

Рис. 7.2. Схема процедуры рукопожатия (пользователь А проверяет подлинность пользователя В)

Пусть пользователь А инициирует процедуру «рукопожатия», отправляя пользователю В свой идентификатор ID_A в открытой форме. Пользователь В, получив идентификатор ID_A, находит в базе данных секретный ключ K_{AB} и вводит его в свою криптосистему.

Тем временем пользователь А генерирует случайную последовательность S с помощью псевдослучайного генератора PG и отправляет ее пользователю В в виде криптограммы $E_{K_{AB}}(S)$. Пользователь В расшифровывает эту криптограмму и раскрывает исходный вид последовательности S. Затем оба пользователя А и В преобразуют последовательность S, используя открытую одностороннюю функцию $a(*)$.

Пользователь В шифрует сообщение $a(S)$ и отправляет эту криптограмму пользователю А. Наконец, пользователь А расшифровывает эту криптограмму и сравнивает полученное сообщение $a'(S)$ с исходным $a(S)$. Если они равны, пользователь А признает подлинность пользователя В.

Очевидно, пользователь В проверяет подлинность пользователя А таким же способом. Обе операции образуют процедуру «рукопожатия», которая обычно выполняется в самом начале любого сеанса связи между любыми двумя сторонами в компьютерных сетях.

Достоинством модели «рукопожатия» является то, что ни один из участников сеанса связи не получает никакой секретной информации во время процедуры подтверждения подлинности.

Процедура «рукопожатия» была рассмотрена в предположении, что пользователи А и В доверяют друг другу и имеют общий секретный сеансовый ключ. Однако нередки ситуации, когда пользователи должны осуществить взаимную аутентификацию, не доверяя друг другу и не обмениваясь никакой конфиденциальной информацией.

7.4. Протоколы идентификации с нулевой передачей знаний

Описанная выше ситуация возникает при использовании интеллектуальных карт (смарт-карт) для разнообразных коммерческих, гражданских и военных применений (кредитные карты, карты социального страхования, карты доступа в охраняемое помещение, компьютерные пароли, ключи и т. п.) Во многих приложениях главная проблема заключается в том, чтобы при предъявлении интеллектуальной карты оперативно обнаружить обман и отказать обманщику в допуске, ответе или обслуживании.

Для безопасного использования интеллектуальных карт разработаны протоколы идентификации с нулевой передачей знаний. Секретный ключ владельца карты становится неотъемлемым признаком его личности. Доказательство знания секретного ключа с нулевой передачей этого знания служит доказательством подлинности личности владельца карты.

Упрощенная схема идентификации с нулевой передачей знаний

Схему идентификации с нулевой передачей знаний предложили в 1986 г. У. Фейге, А. Фиат и А. Шамир. Она является наиболее известным доказательством идентичности с нулевой передачей конфиденциальной информации.

Рассмотрим упрощенный вариант схемы идентификации с нулевой передачей знаний для более четкого выявления ее основной концепции. Прежде всего выбирают случайное значение модуля n, который является произведением двух больших простых чисел. Он должен иметь длину 512–1024 бит. Это значение n может быть представлено группе пользователей, которым необходимо доказывать свою подлинность. В процессе идентификации участвуют две стороны:

– сторона А, доказывающая свою подлинность;

– сторона В, проверяющая представляемое стороной А доказательство.

Для того чтобы сгенерировать открытый и секретный ключи для стороны А, доверенный арбитр (Центр) выбирает некоторое число V, которое является квадратичным вычетом по модулю n. Иначе говоря, выбирается такое число V, что сравнение $x^2 \equiv V \bmod n$ имеет решение и существует целое число $V^{-1} \bmod n$.

Выбранное значение V является открытым ключом для А. Затем вычисляют наименьшее значение S, для которого $S = sqrt\left(V^{-1}\right) (\bmod n)$. Это значение S является секретным ключом для А.

Теперь можно приступить к выполнению протокола идентификации.

1. Сторона А выбирает некоторое случайное число r, где $r < n$. Затем она вычисляет $x = r^2 \bmod n$ и отправляет x стороне В.

2. Сторона В посылает А случайный бит b.

3. Если $b = 0$, тогда A отправляет r стороне B. Если $b = 1$, то A отправляет стороне B $y = r \cdot S \bmod n$.

4. Если $b = 0$, то сторона B проверяет, что $x = r^2 \bmod n$, чтобы убедиться, что A знает $sqrt(x)$. Если $b = 1$, сторона B проверяет, что $x = y^2 \cdot V \bmod n$, чтобы быть уверенной, что A знает $sqrt(V^{-1})$.

Эти шаги образуют один цикл протокола, называемый *аккредитацией*. Стороны A и B повторяют этот цикл t раз при разных случайных значениях r и b до тех пор, пока B не убедится, что A знает значение S.

Если сторона A не знает значения S, она может выбрать такое значение r, которое позволит ей обмануть сторону B, если B отправит ей $b = 0$, либо A может выбрать такое r, которое позволит обмануть B, если B отправит ей $b = 1$. Но этого невозможно сделать в обоих случаях. Вероятность того, что A обманет B в одном цикле, составляет 1/2. Вероятность обмануть B в t циклах равна $(1/2)^t$.

Для того чтобы этот протокол работал, сторона A не должна повторно использовать значение r. Если A поступила бы таким образом, а сторона B отправила бы стороне A на шаге 2 другой случайный бит b, то B имела бы оба ответа A. После этого B может вычислить значение S и для A все закончено.

Параллельная схема идентификации с нулевой передачей знаний

Параллельная схема идентификации позволяет увеличить число аккредитаций, выполняемых за один цикл, и тем самым уменьшить длительность процесса идентификации.

Как и в предыдущем случае, сначала генерируется число n как произведение двух больших чисел. Для того чтобы сгенерировать открытый и секретный ключи для стороны A, сначала выбирают k различных чисел $V_1, V_2, \ldots, V_k$, где каждое V_i является квадратичным вычетом по модулю n. Иначе говоря, выбирают значение V_i таким, что сравнение $x^2 \equiv V_i \bmod n$ имеет решение и существует $V_i^{-1} \bmod n$. Полученная строка $V_1, V_2, \ldots, V_k$ является открытым ключом. Затем вычисляют такие наименьшие значения S_i, что $S_i = sqrt\left(V_i^{-1}\right) \bmod n$. Строка $S_1, S_2, \ldots, S_k$ является секретным ключом стороны A.

Процесс идентификации состоит из нескольких действий.

1. Сторона А выбирает некоторое случайное число r, где $r < n$. Затем она вычисляет $x = r^2 \bmod n$ и посылает x стороне В.

2. Сторона В отправляет стороне А некоторую случайную двоичную строку из K бит: $b_1, b_2, \ldots, b_k$.

3. Сторона А вычисляет $y = r \cdot (S^{b_1} \cdot S^{b_2} \cdot \ldots \cdot S^{b_k}) \bmod n$.

Перемножаются только те значения S_i, для которых $b_i = 1$. Например, если $b_i = 1$, то сомножитель S_i входит в произведение, если же $b_i = 0$, то S_i не входит в произведение и т. д. Вычисленное значение y отправляется стороне В.

4. Сторона В проверяет, что $x = y^2 \cdot \left(V_1^{b_1} \cdot V_2^{b_2} \cdot \ldots \cdot V_k^{b_k}\right) \bmod n$.

Фактически сторона В перемножает только те значения V_i, для которых $b_i = 1$. Стороны А и В повторяют этот протокол t раз, пока В не убедится, что А знает $S_1, S_2, \ldots, S_k$.

Вероятность того, что А может обмануть В, равна $(1/2)^{Kt}$. Рекомендуется в качестве контрольного значения брать вероятность обмана В равной $(1/2)^{20}$ при $K = 5$ и $t = 4$.

Стороны А и В повторяют этот протокол t раз, каждый раз с разным случайным числом r, пока сторона В не будет удовлетворена.

При малых значениях величин, как в данном примере, не достигается настоящей безопасности. Но если n представляет собой число длиной 512 бит и более, сторона В не сможет узнать ничего о секретном ключе стороны А, кроме того факта, что сторона А знает этот ключ.

8. УПРАВЛЕНИЕ КРИПТОГРАФИЧЕСКИМИ КЛЮЧАМИ

Любая криптографическая система основана на использовании криптографических ключей. В симметричной криптосистеме отправитель и получатель сообщения используют один и тот же секретный ключ. Этот ключ должен быть неизвестен остальным и периодически обновляться одновременно у отправителя и получателя. Процесс распределения (рассылки) секретных ключей между участниками информационного обмена в симметричных криптосистемах имеет весьма сложный характер.

Асимметричная криптосистема предполагает использование двух ключей – открытого и личного (секретного). Открытый ключ можно разглашать, а личный – надо хранить в тайне. При обмене сообщениями необходимо пересылать только открытый ключ. Важным требованием является обеспечение подлинности отправителя сообщения. Это достигается путем взаимной аутентификации участников информационного обмена.

Под *ключевой информацией* понимают совокупность всех действующих в системе ключей. Если не обеспечено достаточно надежное управление ключевой информацией, то, завладев ею, злоумышленник получает неограниченный доступ ко всей информации.

Управление ключами – информационный процесс, включающий реализацию следующих основных функций:

- генерацию ключей;
- хранение ключей;
- распределение ключей.

8.1. **Генерация ключей**

Безопасность любого криптографического алгоритма определяется используемым криптографическим ключом. Надежные криптографические ключи должны иметь достаточную длину и случайные значения битов.

Для получения ключей используются аппаратные и программные средства генерации случайных значений ключей. Как правило, применяют датчики псевдослучайных чисел (ПСЧ). Однако степень слу-

чайности генерации чисел должна быть достаточно высокой. Идеальными генераторами являются устройства на основе «натуральных» случайных процессов, например на основе белого радиошума.

Если ключ не меняется регулярно, это может привести к его раскрытию и утечке информации. Регулярную замену можно осуществить, используя процедуру модификации ключа.

Модификация ключа – это генерирование нового ключа из предыдущего значения ключа с помощью односторонней (однонаправ-ленной) функции. Участники информационного обмена разделяют один и тот же ключ и одновременно вводят его значение в качестве аргумента в одностороннюю функцию, получая один и тот же результат. Затем берут определенные биты из этих результатов, чтобы создать новое значение ключа.

Процедура модификации ключа работоспособна, но надо помнить, что новый ключ безопасен в той же мере, в какой был безопасен прежний ключ. Если злоумышленник сможет добыть прежний ключ, то он сможет выполнить процедуру модификации ключа.

Генерация ключей для асимметричных криптосистем с открытыми ключами намного сложнее, потому что эти ключи должны обладать определенными математическими свойствами, т. е. быть очень большими и простыми и т. д.

8.2. **Хранение ключей**

Под функцией *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. Любая информация об используемых ключах должна быть защищена, в частности храниться в зашифрованном виде.

Необходимость в хранении и передаче ключей, зашифрованных с помощью других ключей, приводит к концепции иерархии ключей. Суть концепции состоит в том, что вводится иерархия ключей: главный ключ (ГК), ключ шифрования ключей (КК), ключ шифрования данных (КД). Иерархия ключей может быть:

– двухуровневой (КК / КД);
– трехуровневой (ГК / КК / КД).

Нижним уровнем являются рабочие или сеансовые КД, которые используются для шифрования данных, персональных идентификационных номеров (PIN) и аутентификации сообщений. Когда эти ключи надо зашифровать с целью защиты при передаче или хранении, используют ключи шифрования ключей. Они никогда не должны использоваться как сеансовые (рабочие) КД, и наоборот.

Такое разделение функций необходимо для обеспечения максимальной безопасности. Фактически концепция устанавливает, что различные типы рабочих ключей (например, для шифрования данных, для аутентификации и т. д.) должны всегда шифроваться с помощью различных версий ключей шифрования ключей.

В частности, ключи шифрования ключей, используемые для пересылки ключей между двумя узлами сети, известны так же, как и ключи обмена между узлами сети. Обычно в канале используются два ключа для обмена между узлами сети, по одному в каждом направлении. Поэтому каждый узел сети будет иметь ключ отправления для обмена с узлами сети и ключ получения для каждого канала, поддерживаемого другим узлом сети.

На верхнем уровне иерархии ключей располагается главный ключ, мастер-ключ. Его применяют для шифрования КК, когда требуется сохранить их на диске. Обычно в каждом компьютере используется только один мастер-ключ.

Мастер-ключ распространяется между участниками обмена при личном контакте, чтобы исключить его перехват и/или компрометацию. Раскрытие противником значения мастер-ключа полностью уничтожает защиту компьютера.

Значение мастер-ключа фиксируется на длительное время (до нескольких недель или месяцев, поэтому его генерация и хранение являются критическими вопросами криптографической защиты. На практике мастер-ключ компьютера создается истинно случайным выбором из всех возможных значений ключей. Главный ключ помещают в защищенный по считыванию и записи и от механических воздействий блок криптографической системы таким образом, чтобы раскрыть значение этого ключа было невозможно. Однако должен существовать способ проверки, является ли значение ключа правильным.

Проблема аутентификации мастер-ключа может быть решена различными путями. Один из способов аутентификации показан на рис. 8.1.

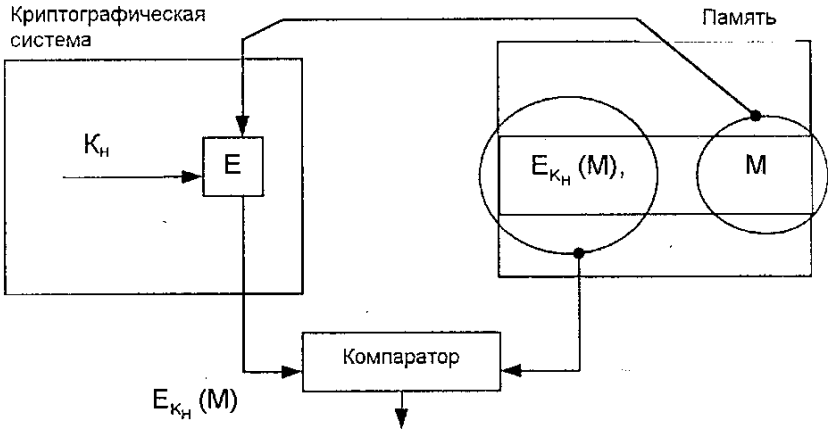

Рис. 8.1. Схема аутентификации мастер-ключа хост-компьютера

Администратор, получив новое значение мастер-ключа $K_н$ хост-компьютера, шифрует некоторое сообщение M ключом $K_н$. Пара (криптограмма $E_{K_н}(M)$, сообщение M) помещается в память компьютера. Всякий раз, когда требуется аутентификация мастер-ключа хост-компьютера, берется сообщение M из памяти и подается в криптографическую систему. Получаемая криптограмма сравнивается с криптограммой, хранящейся в памяти. Если они совпадают, считается, что данный ключ является правильным.

Рабочие ключи (например, сеансовый) обычно создаются с помощью псевдослучайного генератора и могут храниться в незащищенном месте. Это возможно, поскольку такие ключи генерируются в форме соответствующих криптограмм, т. е. генератор ПСЧ выдает вместо ключа K_S его криптограмму $E_K(K_S)$, получаемую с помощью мастер-ключа K хост-компьютера. Расшифровывание такой криптограммы выполняется только перед использованием ключа K_S.

Схема защиты рабочего (сеансового) ключа показана на рис. 8.2. Чтобы зашифровать сообщение M ключом K_S, на соответствующие входы криптографической системы подается криптограмма $E_K(K_S)$

и сообщение *M*. Криптографическая система сначала восстанавливает ключ K_S, а затем шифрует сообщение *M*, используя открытую форму сеансового ключа K_S.

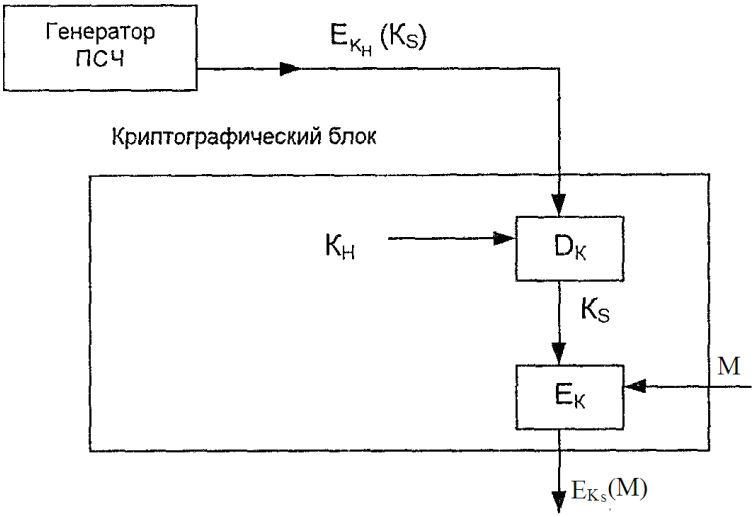

Рис. 8.2. Схема защиты сеансового ключа K_S

Таким образом, безопасность сеансовых ключей зависит от безопасности криптографической системы. Криптографический блок может быть спроектирован как единая СБИС и помещен в физически защищенное место. Важным условием безопасности информации является периодическое обновление ключевой информации в криптосистеме. При этом должны переназначаться как рабочие ключи, так и мастер-ключи. В особо ответственных системах обновление ключевой информации (сеансовых ключей) желательно делать ежедневно. Вопрос обновления ключевой информации тесно связан с третьим элементом управления ключами – распределением ключей.

8.3. Распределение ключей

Распределение ключей – самый ответственный процесс в управлении ключами. К нему предъявляются следующие требования:

– оперативность и точность распределения;

– скрытность распределяемых ключей.

Распределение ключей между пользователями компьютерной сети реализуется двумя способами:

– использованием одного или нескольких центров распределения ключей;

– прямым обменом сеансовыми ключами между пользователями сети.

Недостаток первого подхода состоит 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 надежно установить подлинность субъектов сети.

В обоих случаях должна быть обеспечена подлинность сеанса связи. Это можно осуществить, используя уже рассмотренные ранее процедуры аутентификации, с использованием механизма запроса-ответа или механизма отметки времени.

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 основу решения задачи распределения ключей целесообразно положить принцип отделения процедуры подтверждения подлинности партнеров от процедуры собственно 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.

8.3.1. Распределение ключей с участием центра распределения ключей

Каждый из участников сеанса А и В имеет мастер-ключ K_i для объекта i ($i = A; B$), известный только ему и ЦРК. Эти мастер-ключи генерируются в ЦРК и распределяются каждому объекту при личном контакте. Главный ключ используется для шифрования сеансового ключа, когда последний передается по сети. Сеансовый ключ K_S генерируется в ЦРК и используется участниками сеанса А и В для защиты сообщений при передаче по линиям связи. Для предотвращения фальсифицированных повторных вызовов на стадии распределения ключей в этом протоколе применяются отметки времени T.

Участник А инициирует фазу распределения ключей, посылая в ЦРК по сети идентификаторы ID_A и ID_B:

$$\text{(1) A} \rightarrow \text{ЦРК: } ID_\text{A}, ID_\text{B}, E_{K_\text{A}}(ID_\text{B}, \text{«Прошу связь с B»}).$$

Идентификатор ID_A посылается в явном виде, поэтому менеджер ЦРК будет знать, какой мастер-ключ необходим для расшифровывания зашифрованной части сообщения. Для этого в ЦРК имеются таблицы идентификаторов и соответствующих им мастер-ключей. Любая попытка злоумышленника изменить ID_A приведет к получению неправильного ключа для расшифровывания и, следовательно, к выявлению нарушителя службой ЦРК.

Если сообщение правильное, менеджер ЦРК разыскивает мастер-ключ K_A, а также вычисляет сеансовый ключ K_S. Затем участнику А посылается ответное сообщение:

$$\text{(2) ЦРК} \rightarrow \text{A: } E_{K_\text{A}}(T, K_S, ID_\text{B}, E_{K_\text{B}}(T, K_S, ID_\text{A})).$$

Это сообщение может расшифровать только А, поскольку оно зашифровано ключом K_A. Участник А проверяет отметку времени T, чтобы убедиться, что это сообщение не является повтором предыдущей процедуры распределения ключей. Идентификатор ID_A убеждает А, что требуемый адресат был действительно указан в сообщении (1). Участник А сохраняет у себя сеансовый ключ K_S и посылает участнику В часть сообщения, зашифрованную мастер-клю-чом объекта В, а также случайное число r_1, зашифрованное сеансовым ключом K_S:

$$\text{(3) A} \rightarrow \text{B: } E_{K_\text{B}}(T, K_S, ID_\text{A}), E_{K_S}(r_1).$$

В этом случае только участник В может расшифровать сообщение (3). Участник В получает отметку времени T, сеансовый ключ K_S и идентификатор ID_A. Если отметка времени T верна, то В уверен, что никто, кроме А, не может быть вызывающим объектом. Фактически, верная отметка времени и уникальный идентификатор участника А, зашифрованные ключом K_B, обеспечивают подтверждение подлинности А по отношению к В. Далее В извлекает из сообщения случайное число r_1, выполняя расшифровывание сеансовым ключом K_S. Для взаимного подтверждения подлинности участник В посылает А сообщение, зашифрованное ключом K_S и содержащее некоторую функцию от случайного числа $f(r_1)$, например, $r_1 - 1$:

$$(4)\ \text{B} \rightarrow \text{A}: E_{K_S}(r_1 - 1).$$

Если после расшифровывания сообщения (4) участник А получает правильный результат, он знает, что объект на другом конце линии связи действительно В.

Если все шаги успешно выполнены, то этих взаимных подтверждений достаточно, чтобы установить связь, которая будет проходить под сеансовым ключом K_S. Следует отметить, что в этом протоколе необходим обмен с ЦРК для получения сеансового ключа каждый раз, когда А желает установить связь с В. Данный протокол обеспечивает надежное соединение объектов А и В при условии, что ни один из ключей не скомпрометирован и ЦРК защищен.

Протокол для асимметричных криптосистем с использованием сертификатов открытых ключей

В этом протоколе используется идея сертификатов открытых ключей. Хотя открытые ключи предполагаются известными всем, посредничество ЦРК позволяет подтвердить их подлинность. Без такого посредничества злоумышленник может снабдить А своим открытым ключом, который А будет считать ключом участника В. Затем злоумышленник может подменить собой В и установить связь с А, и его никто не сможет выявить.

Сертификатом открытого ключа C называется сообщение ЦРК, удостоверяющее подлинность некоторого открытого ключа объекта. Например, сертификат открытого ключа для пользователя А,

обозначаемый C_A, содержит отметку времени T, идентификатор ID_A и открытый ключ K_{A_o}, зашифрованные секретным ключом ЦРК K_c, т. е. $C_A = E_{K_c}(T, ID_A, K_{A_o})$.

Отметка времени T используется для подтверждения актуальности сертификата и тем самым предотвращает повторы прежних сертификатов, которые содержат открытые ключи и для которых соответствующие секретные ключи несостоятельны/

Секретный ключ ЦРК K_c известен только менеджеру ЦРК, открытый ключ ЦРК K_o – участникам А и В. ЦРК поддерживает таблицу открытых ключей всех объектов сети, которые он обслуживает.

Вызывающий объект А инициирует стадию установления ключа, запрашивая у ЦРК сертификат своего открытого ключа и открытого ключа участника В:

(1) А → ЦРК: ID_A, ID_B, «Вышлите сертификаты ключей А и В»,

где ID_A, ID_B – уникальные идентификаторы, соответственно, участников А и В.

Менеджер ЦРК отвечает сообщением:

$$(2)\ \text{ЦРК} \rightarrow \text{А}: E_{K_c}(T, ID_A, K_{A_o}), E_{K_c}(T, ID_B, K_{B_o}).$$

Участник А, используя открытый ключ ЦРК K_o, расшифровывает ответ ЦРК и проверяет оба сертификата. Идентификатор ID_B убеждает А, что личность вызываемого участника правильно зафиксирована в ЦРК и K_{B_o} – действительно открытый ключ участника В, поскольку оба зашифрованы ключом K_c.

Следующий шаг протокола включает установление связи А с В:

$$(3)\ \text{А} \rightarrow \text{В}: C_A, E_{K_{A_c}}(T), E_{K_{B_o}}(r_1),$$

где C_A – сертификат открытого ключа пользователя А;

$E_{K_{A_c}}(T)$ – отметка времени, зашифрованная секретным ключом участника А и являющаяся подписью участника А, поскольку никто другой не может создать такую подпись;

r_1 – случайное число, генерируемое А и используемое для обмена с В, в ходе процедуры проверки подлинности.

Если сертификат C_A и подпись А верны, то участник В уверен, что сообщение пришло от А. Часть сообщения $E_{K_{B_o}}(r_1)$ может расшифровать только В, поскольку никто другой не знает секретного ключа K_{B_c}, соответствующего открытому ключу K_{B_o}. Участник В расшифровывает значение числа r_1 и, чтобы подтвердить свою подлинность, посылает участнику А сообщение:

$$(4)\ \text{B} \rightarrow \text{A}: E_{K_{A_o}}(r_1).$$

Участник А восстанавливает значение r_1, расшифровывая сообщение с использованием своего секретного ключа K_{A_c}. Если это ожидаемое значение r_1, то А получает подтверждение, что вызываемый участник действительно В.

Протокол, основанный на симметричном шифровании, функционирует быстрее, чем протокол, основанный на криптосистемах с открытыми ключами. Однако способность систем с открытыми ключами генерировать цифровые подписи, обеспечивающие различные функции защиты, компенсирует избыточность требуемых вычислений.

8.3.2. Прямой обмен ключами между пользователями

При использовании для информационного обмена криптосистемы с симметричным секретным ключом два пользователя, желающие обменяться криптографически защищенной информацией, должны обладать общим секретным ключом. Для этого пользователи должны обменяться общим ключом по безопасному каналу связи. Если пользователи меняют ключ достаточно часто, то доставка ключа превращается в серьезную проблему.

Для решения этой проблемы можно применить два способа:

– использование криптосистемы с открытым ключом для шифрования и передачи секретного ключа симметричной криптосистемы;

– использование системы открытого распределения ключей Диффи-Хелмана.

Первый способ был уже рассмотрен ранее. Второй способ основан на применении системы открытого распределения ключей. Эта система позволяет пользователям обмениваться ключами по незащищенным каналам связи.

Алгоритм открытого распределения ключей Диффи-Хелмана

Алгоритм Диффи-Хелмана был первым алгоритмом с открытыми ключами (предложен в 1976 г.). Его безопасность обусловлена трудностью вычисления дискретных логарифмов в конечном поле, в отличие от легкости дискретного возведения в степень в том же конечном поле.

Предположим, что два пользователя A и B хотят организовать защищенный коммуникационный канал.

1. Обе стороны заранее уславливаются о модуле n (n должно быть простым числом, $(n-1)/2$ также должно быть простым) и элементе g ($1 \le g \le n-1$). Как правило, эти значения являются общими для всех пользователей системы.

2. Затем пользователи A и B независимо друг от друга выбирают собственные секретные ключи X_A и X_B (X_A, X_B) – случайные большие целые числа, которые хранятся пользователями A и B в секрете).

3. Далее пользователь A вычисляет открытый ключ $Y_A = g^{X_A} (\mathrm{mod}\ n)$, а пользователь B – открытый ключ $Y_B = g^{X_B} (\mathrm{mod}\ n)$.

4. Затем стороны A и B обмениваются вычисленными значениями открытых ключей Y_A и Y_B по незащищенному каналу. (Считается, что все данные, передаваемые по незащищенному каналу связи, могут быть перехвачены злоумышленником.)

5. Далее пользователи A и B вычисляют общий секретный ключ.

Пользователь A: $K = (Y_B)^{X_A} = (g^{X_B})^{X_A} \mathrm{mod}\ n$,

пользователь B: $K = (Y_A)^{X_B} = (g^{X_A})^{X_B} \mathrm{mod}\ n$. При этом $K = K'$, так как $(g^{X_B})^{X_A} \mathrm{mod}\ n = (g^{X_A})^{X_B} \mathrm{mod}\ n$.

Ключ K может использоваться в качестве общего секретного ключа (ключа шифрования ключей) в симметричной криптосистеме.

Алгоритм открытого распределения ключей Диффи-Хелмана позволяет обойтись без защищенного канала для передачи ключей. Однако, работая с этим алгоритмом, необходимо иметь гарантию того, что пользователь A получил открытый ключ именно от пользователя B, и наоборот. Эта проблема решается с помощью электронной подписи, которой подписываются сообщения об открытом ключе.

Метод Диффи-Хелмана дает возможность шифровать данные при каждом сеансе связи на новых ключах. Это позволяет не хранить

секреты на дискетах или других носителях. Не следует забывать, что любое хранение секретов повышает вероятность попадания их в руки конкурентов или противника.

ЛИТЕРАТУРА

1. Голиков, В.Ф. Криптографическая защита информации в телекоммуникационных системах: в 2 ч. / В.Ф. Голиков, А.В. Курилович. – Минск: БГУИР, 2006. – Ч. 1. – 54 с.

2. Голиков, В.Ф. Криптографическая защита информации в телекоммуникационных системах: 2 ч. / В.Ф. Голиков, А.В. Курило-вич. – Минск: БГУИР, 2008. – Ч. 2. – 29 с.

3. Основы криптологии / Ю.С. Харин [и др.]. – Минск: Новое знание, 2003. – 381 с.

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

5. Шнайер, Б. Прикладная криптография / Б. Шнайер. – М.: Триумф, 2002. – 758 с.

6. Асосков, А.В. Поточные шифры / А.В. Асосков. – М.: Кудиц-образ, 2003. – 333 с.

Учебное издание

ГОЛИКОВ Владимир Федорович

БЕЗОПАСНОСТЬ ИНФОРМАЦИИ И НАДЕЖНОСТЬ КОМПЬЮТЕРНЫХ СИСТЕМ

Методическое пособие
для студентов специальностей
1-40 01 01 «Программное обеспечение информационных технологий»
и 1-53 01 02 «Автоматизированные системы обработки информации»
всех форм обучения

В 2 частях

Ч а с т ь 2

КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА ИНФОРМАЦИИ

Редактор Т.В. Кипель
Компьютерная верстка Н.А. Школьниковой

Подписано в печать 22.05.2012.
Формат 60×84 1/16. Бумага офсетная.
Отпечатано на ризографе. Гарнитура Таймс.
Усл. печ. л. 5,29. Уч.-изд. л. 4,14. Тираж 100. Заказ 752.

Издатель и полиграфическое исполнение:
Белорусский национальный технический университет.
ЛИ № 02330/0494349 от 16.03.2009.
Проспект Независимости, 65. 220013, Минск.