А.В. Яковлев, А.А. Безбогов, В.В. Родин, В.Н. Шамкин

КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА ИНФОРМАЦИИ

• ИЗДАТЕЛЬСТВО ТГТУ •

А.В. ЯКОВЛЕВ КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА ИНФОРМАЦИИ

Министерство образования и науки Российской Федерации

ГОУ ВПО "Тамбовский государственный технический университет"

А.В. Яковлев, А.А. Безбогов, В.В. Родин, В.Н. Шамкин

КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА ИНФОРМАЦИИ

Утверждено Ученым советом университета в качестве учебного пособия

Тамбов
Издательство ТГТУ
2006

УДК 681.322.067

ББК З973.26-018.2я73
К824

Рецензенты:

Заведующий кафедрой КРЭМС Заслуженный деятель науки и техники РФдоктор технических наук, профессор
Ю.Л. Муромцев

Директор РУНЦ по ИБ
Заслуженный работник высшей школы, профессор
Ю.Ф. Мартемьянов

К824 Криптографическая защита информации : учебное пособие / А.В. Яковлев, А.А. Безбогов, В.В. Родин, В.Н. Шамкин. – Тамбов : Изд-во Тамб. гос. техн. ун-та, 2006. – 140 с. – 100 экз. – ISBN 5-8265-0503-6.

Содержит материал в соответствии с Государственным образовательным стандартом по специальности 090105 "Комплексное обеспечение информационной безопасности автоматизированных систем".

Рассматриваются, классифицируются и анализируются основные алгоритмы шифрования, оценивается их эффективность, надежность и особенности реализации.

Пособие может быть полезно при курсовом и дипломном проектировании, аспирантам, а также кругу читателей, интересующихся современными проблемами криптографической защиты информации.

УДК 681.322.067
ББК Z973.26-018.2я73

ISBN 5-8265-0503-6

Учебное издание

ЯКОВЛЕВ Алексей Вячеславович
БЕЗБОГОВ Александр Александрович
РОДИН Владимир Валентинович,
ШАМКИН Валерий Николаевич

КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА ИНФОРМАЦИИ

Учебное пособие

Редактор З.Г. Чернова

Компьютерное макетирование Е.В. Кораблевой

Подписано в печать 25.09.2006
Формат 60 × 84/16. Бумага офсетная. Гарнитура Times New Roman.
8,0 уч.-изд. л. Тираж 100 экз. Заказ № 413

Издательско-полиграфический центр ТГТУ
392000, Тамбов, Советская, 106, к. 14

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В течение многих лет криптография служила исключительно военным целям. Сегодня обычные пользователи получают возможность обращаться к средствам, позволяющим им обезопасить себя от несанкционированного доступа к конфиденциальной информации, применяя методы компьютерной криптографии.

В настоящем учебном пособии последовательно рассматриваются, классифицируются и анализируются основные алгоритмы шифрования, их эффективность, надежность и особенности реализации.

Вначале рассмотрены криптографические системы, приведена краткая история криптографии, введены основные понятия и определения, указаны требования к криптографическим системам, представлены сведения о криптоанализе и рассмотрена классификация методов криптографического закрытия информации.

Далее описана наиболее распространенная в настоящее время симметричная криптосистема. Приведены элементы теории, а также методы и алгоритмы шифрования. Для алгоритмов блочного шифрования представлены способы генерирования блочного ключа и режимы применения блочных шифров. Для потоковых шифров приведены методы и алгоритмы шифрования. Рассмотрены также комбинированные методы симметричного шифрования.

Асимметричные алгоритмы шифрования, получающие все более широкое распространение, представлены краткими теоретическими сведениями об асимметричном шифровании, обсуждением некоторых распространенных криптосистем с асимметричным шифрованием, примерами применения асимметричных алгоритмов шифрования.

Обсуждены вопросы, относящиеся к электронной цифровой подписи, как к эффективному средству криптозащиты. Рассмотрена постановка задачи верификации сообщений с помощью электронной цифровой подписи, приведены алгоритмы электронной цифровой подписи, основанные как на симметричных, так и на асимметричных криптосистемах, представлены алгоритмы формирования функций хэширования, удовлетворяющих условиям использования в процессе аутентификации сообщений.

Представлена методика управления криптографическими ключами. Рассмотрены обычная система управления ключами, управление ключами, основанное на системах с открытым ключом, протокол обмена секретным ключом, использование сертификатов, протоколы аутентификации и анонимное распределение ключей.

В пособии содержатся сведения по технической реализации алгоритмов.

В конце каждой главы приведены контрольные вопросы.

1. КРИПТОГРАФИЧЕСКИЕ СИСТЕМЫ

1.1. ИСТОРИЯ КРИПТОГРАФИИ

С распространением письменности в человеческом обществе появилась потребность в обмене письмами и сообщениями, что вызвало необходимость сокрытия содержимого письменных сообщений от посторонних. Методы сокрытия содержимого письменных сообщений можно разделить на три группы. К первой группе относятся методы маскировки или *стеганографии*, которые осуществляют сокрытие самого факта наличия сообщения; вторую группу составляют различные методы тайнописи или *криптографии* (от греческих слов *ktyptos* – тайный и *grapho* – пишу); методы третьей группы ориентированы на создание специальных технических устройств, засекречивания информации.

История криптографии – ровесница истории человеческого языка. Более того, первоначально письменность сама по себе была своеобразной криптографической системой, так как в древних обществах ею владели только избранные.

Развитию тайнописи способствовали войны. Письменные приказы и донесения обязательно шифровались, чтобы пленение курьеров не позволило противнику получить важную информацию. Например, римский император Цезарь пользовался в своей военной и личной переписке шифром, сущность которого состояла в замене каждой буквы латинского языка на следующую букву алфавита. Тогда знаменитая фраза: "*VENI*, *VIDI*, *VICI*" ("Пришел, увидел, победил"), которой Цезарь, известил одного из своих друзей в Риме о быстро одержанной им победе, в зашифрованном виде будет иметь следующий вид: "*XFOJ*, *XJEJ*, *XJDJ*".

Практически одновременно с криптографией стал развиваться и криптоанализ – наука о раскрытии шифров (ключей) по шифртексту.

В истории криптографии условно можно выделить четыре этапа: наивный, формальный, научный; компьютерный.

Для ***наивной криптографии*** (до начала XVI в.) характерно использование любых, обычно примитивных, способов запутывания противника относительно содержания шифруемых текстов. На начальном этапе для защиты информации использовались методы *кодирования* и *стеганографии,* которые родственны, но не тождественны криптографии.

Большинство из используемых шифров сводились к *перестановке* или *моноалфавитной подстановке.* Одним из первых зафиксированных примеров является шифр Цезаря, состоящий в замене каждой буквы исходного текста на другую, отстоящую от нее в алфавите на определенное число позиций. Другой шифр, полибианский квадрат, авторство которого приписывается греческому писателю Полибию, является общей моноалфавитной подстановкой, которая проводится с помощью случайно заполненной алфавитом квадратной таблицей (для греческого алфавита размер составляет 5 × 5). Каждая буква исходного текста заменяется на букву, стоящую в квадрате снизу от нее.

Этап ***формальной криптографии*** (конец XV – начало XX вв.) связан с появлением формализованных и относительно стойких к ручному криптоанализу шифров. В европейских странах это произошло в эпоху Возрождения, когда развитие науки и торговли вызвало спрос на надежные способы защиты информации. Важная роль на этом этапе принадлежит Леону Батисте Альберти, итальянскому архитектору, который одним из первых предложил *многоалфавитную подстановку.* Данный шифр, получивший имя дипломата XVI в. Блеза Вижинера, состоял в последовательном "сложении" букв исходного текста с ключом (процедуру можно облегчить с помощью специальной таблицы). Его работа "Трактат о шифре" (1466 г.) считается первой научной работой по криптологии.

Одной из первых печатных работ, в которой обобщены и сформулированы известные на тот момент алгоритмы шифрования, является труд "Полиграфия" (1508 г.) немецкого аббата Иоганна Трисемуса. Ему принадлежат два небольших, но важных открытия: способ заполнения полибианского квадрата (первые позиции заполняются с помощью легко запоминаемого

ключевого слова, остальные – оставшимися буквами алфавита) и шифрование пар букв (биграмм).

Простым, но стойким способом многоалфавитной замены (подстановки биграмм) является шифр Плейфера, который был открыт в начале XIX в. Чарльзом Уитстоном. Уитстону принадлежит и важное усовершенствование – шифрование "двойным квадратом". Шифры Плейфера и Уитстона использовались вплоть до первой мировой войны, так как с трудом поддавались ручному криптоанализу.

В XIX в. голландец Керкхофф сформулировал главное требование к криптографическим системам, которое остается актуальным и поныне: *секретность шифров должна быть основана на секретности ключа, но не алгоритма*.

Наконец, последним словом в донаучной криптографии, которое обеспечило еще более высокую криптостойкость, а также позволило автоматизировать (в смысле механизировать) процесс шифрования стали *роторные криптосистемы.*

Одной из первых подобных систем стала изобретенная в 1790 г. Томасом Джефферсоном, будущим президентом США, механическая машина. Многоалфавитная подстановка с помощью роторной машины реализуется вариацией взаимного положения вращающихся роторов, каждый из которых осуществляет "прошитую" в нем подстановку.

Практическое распространение роторные машины получили только в начале XX в. Одной из первых практически используемых машин, стала немецкая *Enigma*, разработанная в 1917 г. Эдвардом Хеберном и усовершенствованная Артуром Кирхом. Роторные машины активно использовались во время второй мировой войны. Помимо немецкой машины Enigma использовались также устройства *Sigaba* (США), *Typex* (Великобритания), *Red*, *Orange* и *Purple* (Япония). Роторные системы – вершина формальной криптографии, так как относительно просто реализовывали очень стойкие шифры. Успешные криптоатаки на роторные системы стали возможны только с появлением ЭВМ в начале 40-х гг.

Главная отличительная черта ***научной криптографии*** (1930 – 60-е гг.) – появление криптосистем со строгим математическим обоснованием криптостойкости. К началу 30-х гг. окончательно сформировались разделы математики, являющиеся научной основой криптологии: теория вероятностей и математическая статистика, общая алгебра, теория чисел, начали активно развиваться теория алгоритмов, теория информации, кибернетика. Своеобразным водоразделом стала работа Клода Шеннона "Теория связи в секретных системах" (1949), которая подвела научную базу под криптографию и криптоанализ. С этого времени стали говорить о КРИПТОЛОГИИ (от греческого *kryptos* – тайный и *logos* – сообщение) – науке о преобразовании информации для обеспечения ее секретности. Этап развития криптографии и криптоанализа до 1949 г. стали называть донаучной криптологией. Шеннон ввел понятия "рассеивание" и "перемешивание", обосновал возможность создания сколь угодно стойких криптосистем.

В 1960-х гг. ведущие криптографические школы подошли к созданию *блочных шифров,* еще более стойких по сравнению с роторными криптосистемами, однако допускающих практическую реализацию только в виде цифровых электронных устройств.

Компьютерная криптография (с 1970-х гг.) обязана своим появлением вычислительным средствам с производительностью, достаточной для реализации криптосистем, обеспечивающих при большой скорости шифрования на несколько порядков более высокую криптостойкость, чем "ручные" и "механические" шифры.

Первым классом криптосистем, практическое применение которых стало возможно с появлением мощных и компактных вычислительных средств, стали блочные шифры. В 70-е гг. был разработан *американский стандарт шифрования DES* (принят в 1978 г.). Один из его авторов, Хорст Фейстель (сотрудник *IBM*), описал модель блочных шифров, на основе которой были построены другие, более стойкие симметричные криптосистемы, в том числе *отечественный стандарт шифрования* ГОСТ 28147–89.

С появлением *DES* обогатился и криптоанализ, для атак на американский алгоритм был создано несколько новых видов криптоанализа (линей-

ный, дифференциальный и т.д.), практическая реализация которых опять же была возможна только с появлением мощных вычислительных систем.

В середине 70-х гг. XX столетия произошел настоящий прорыв в современной криптографии – *появление асимметричных криптосистем,* которые не требовали передачи секретного ключа между сторонами. Здесь отправной точкой принято считать работу, опубликованную Уитфилдом Диффи и Мартином Хеллманом в 1976 г. под названием "Новые направления в современной криптографии". В ней впервые сформулированы принципы обмена шифрованной информацией без обмена секретным ключом. Независимо к идее асимметричных криптосистем подошел Ральф Меркли. Несколькими годами позже Рон Ривест, Ади Шамир и Леонард Адлеман открыли систему *RSA*, первую практическую асимметричную криптосистему, стойкость которой была основана на проблеме факторизации больших простых чисел. Асимметричная криптография открыла сразу несколько новых прикладных направлений, в частности системы *электронной цифровой подписи* (ЭЦП) и *электронных денег.*

В 1980–90-е гг. появились совершенно новые направления криптографии: вероятностное шифрование, квантовая криптография и другие. Осознание их практической ценности еще впереди. Актуальной остается и задача совершенствования симметричных криптосистем. В этот же период были разработаны нефейстелевские шифры (*SAFER*, *RC*6 и др.), а в 2000 г. после открытого международного конкурса был принят новый национальный стандарт шифрования США – *AES.*

Криптография является одним из наиболее мощных средств обеспечения конфиденциальности и контроля целостности информации. Во многих отношениях она занимает центральное место среди программно-технических регуляторов безопасности. Например, для портативных компьютеров, физически защитить которые крайне трудно, только криптография позволяет гарантировать конфиденциальность информации даже в случае кражи.

Подробнее об увлекательной истории криптографии можно прочитать в литературе [1, 10, 14].

1.2. ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ

Защита данных с помощью шифрования – одно из возможных решений проблемы безопасности. Зашифрованные данные становятся доступными только тем, кто знает, как их расшифровать, и поэтому похищение зашифрованных данных абсолютно бессмысленно для несанкционированных пользователей.

Наукой, изучающей математические методы защиты информации путем ее преобразования, является ***криптология***. Криптология разделяется на два направления – *криптографию* и *криптоанализ.*

Криптография изучает методы преобразования информации, обеспечивающие ее конфиденциальность и аутентичность.

Под *конфиденциальностью* понимают невозможность получения информации из преобразованного массива без знания дополнительной информации (ключа).

Аутентичность информации состоит в *подлинности авторства* и *целостности.*

Криптоанализ объединяет математические методы нарушения конфиденциальности и аутентичности информации без знания ключей.

Существует ряд смежных, но не входящих в криптологию отраслей знания. Так обеспечением *скрытности* информации в информационных массивах занимается *стеганография.* Обеспечение целостности информации в условиях случайного воздействия находится в ведении *теории помехоустойчивого кодирования.* Наконец, смежной областью по отношению к криптологии являются *математические методы сжатия информации.*

Современная криптография включает в себя четыре крупных раздела: симметричные криптосистемы, криптосистемы с открытым ключом, системы электронной подписи, управление ключами.

Основные направления использования криптографических методов – передача конфиденциальной информации по каналам связи (например, электронная почта), установление подлинности передаваемых сообщений,

хранение информации (документов, баз данных) на носителях в зашифрованном виде.

В качестве информации, подлежащей шифрованию и расшифрованию, а также электронной подписи будут рассматриваться *тексты* (*сообщения*), построенные на некотором *алфавите.* Под этими терминами понимается следующее.

Алфавит – конечное множество используемых для кодирования информации знаков.

***Текст* (*сообщение*)** – упорядоченный набор из элементов алфавита. В качестве примеров алфавитов, используемых в современных ИС, можно привести следующие:

- алфавит Z_{33} – 32 буквы русского алфавита (исключая "ё") и пробел;
- алфавит Z_{256} – символы, входящие в стандартные коды ASCII и КОИ-8;
- двоичный алфавит – $Z_2 = \{0, 1\}$;
- восьмеричный или шестнадцатеричный алфавит.

Коды и шифры использовались задолго до появления ЭВМ. С теоретической точки зрения не существует четкого различия между кодами и шифрами. Однако в современной практике различие между ними является достаточно четким. Коды оперируют *лингвистическими элементами*, разделяя шифруемый текст на такие смысловые элементы, как слова и слоги. В шифре всегда различают два элемента: *алгоритм и ключ*.

Алгоритм позволяет использовать сравнительно короткий ключ для шифрования сколь угодно большого текста.

Определим ряд терминов, используемых в криптологии.

Под *шифром* понимается совокупность обратимых преобразований множества открытых данных на множество зашифрованных данных, заданных алгоритмом криптографического преобразования.

Шифр – это совокупность инъективных отображений множества открытых текстов во множество шифрованных текстов, проиндексированная элементами из множества ключей: $\{Fk : \boldsymbol{X} \rightarrow \boldsymbol{S}, K \in \boldsymbol{K}\}$.

Криптографическая система, или ***шифр*** представляет собой семейство T обратимых преобразований открытого текста в шифрованный. Членам этого семейства можно взаимно однозначно сопоставить число k, называемое *ключом.* Преобразование T_k определяется соответствующим алгоритмом и значением ключа k.

Ключ – конкретное секретное состояние некоторых параметров алгоритма криптографического преобразования данных, обеспечивающее выбор одного варианта из совокупности всевозможных для данного алгоритма. Секретность ключа должна обеспечивать невозможность восстановления исходного текста по шифрованному.

Пространство ключей K – это набор возможных значений ключа. Обычно ключ представляет собой последовательный ряд букв алфавита. Следует отличать понятия "ключ" и "пароль". *Пароль* также является секретной последовательностью букв алфавита, однако используется не для шифрования (как ключ), а для аутентификации субъектов.

Криптосистемы подразделяются на *симметричные* и *асимметричные* [или *с открытым* (*публичным*) *ключом*].

В симметричных криптосистемах для зашифрования и для расшифрования используется один и тот же ключ.

В *системах с открытым ключом* используются два ключа *открытый* (*публичный*) и *закрытый* (*секретный*), которые математически связаны друг с другом. Информация зашифровывается с помощью открытого ключа, который доступен всем желающим, а расшифровывается с помощью закрытого ключа, известного только получателю сообщения.

Термины *распределение ключей* и *управление ключами* относятся к процессам системы обработки информации, содержанием которых является выработка и распределение ключей между пользователями.

Электронной* (*цифровой*) *подписью называется присоединяемое к тексту его криптографическое преобразование, которое позволяет при получении текста другим пользователем проверить авторство и целостность сообщения.

Зашифрованием данных называется процесс преобразования открытых данных в зашифрованные с помощью шифра, а ***расшифрованием*** данных – процесс преобразования закрытых данных в открытые с помощью шифра. Вместо термина "открытые данные" часто употребляются термины "*открытый текст*" и "*исходный текст*", а вместо "зашифрованные данные" – "*шифрованный текст*".

Дешифрованием называется процесс преобразования закрытых данных в открытые при неизвестном ключе и, возможно, неизвестном алгоритме, т.е. методами криптоанализа.

Шифрованием называется процесс зашифрования или расшифрования данных. Также термин шифрование используется как синоним зашифрования. Однако неверно в качестве синонима шифрования использовать термин "*кодирование*" (а вместо "*шифра*" – "*код*"), так как под кодированием обычно понимают представление информации в виде знаков (букв алфавита).

Криптостойкостью называется характеристика шифра, определяющая его стойкость к дешифрованию. Обычно эта характеристика определяется периодом времени, необходимым для дешифрования.

Гаммирование – процесс наложения по определенному закону гаммы шифра на открытые данные.

Гамма шифра – псевдослучайная двоичная последовательность, вырабатываемая по заданному алгоритму, для зашифрования открытых данных и расшифрования зашифрованных данных.

Имитозащита – защита от навязывания ложных данных. Для обеспечения имитозащиты к зашифрованным данным добавляется имитовставка, представляющая собой последовательность данных фиксированной длины, полученную по определенному правилу из открытых данных и ключа.

Криптографическая защита – это защита данных с помощью криптографического преобразования, под которым понимается преобразование данных *шифрованием* и (или) выработкой *имитовставки*.

Синхропосылка – исходные открытые параметры алгоритма криптографического преобразования.

Уравнение зашифрования (расшифрования) – соотношение, описывающее процесс образования зашифрованных (открытых) данных из открытых (зашифрованных) данных в результате преобразований, заданных алгоритмом криптографического преобразования.

1.3. ТРЕБОВАНИЯ К КРИПТОГРАФИЧЕСКИМ СИСТЕМАМ

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

Независимо от способа реализации для современных криптографических систем защиты информации сформулированы следующие общепринятые требования:

- стойкость шифра противостоять криптоанализу должна быть такой, чтобы вскрытие его могло быть осуществлено только решением задачи полного перебора ключей и должно либо выходить за пределы возможностей современных компьютеров (с учетом возможности организации сетевых вычислений) или требовать создания использования дорогих вычислительных систем;
- криптостойкость обеспечивается не секретностью алгоритма, а секретностью ключа (разделяет криптосистемы общего использования (алгоритм доступен потенциальному нарушителю) и ограниченного использования (алгоритм держится в секрете));
- зашифрованное сообщение должно поддаваться чтению только при наличии ключа;
- шифр должен быть стойким даже в случае, если нарушителю известно достаточно большое количество исходных данных и соответствующих им зашифрованных данных;
- незначительное изменение ключа или исходного текста должно приводить к существенному изменению вида зашифрованного текста;

– структурные элементы алгоритма шифрования должны быть неизменными;

– шифртекст не должен существенно превосходить по объему исходную информацию; дополнительные биты, вводимые в сообщение в процессе шифрования, должны быть полностью и надежно скрыты в шифрованном тексте;

– ошибки, возникающие при шифровании, не должны приводить к искажениям и потерям информации;

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

– любой ключ из множества возможных должен обеспечивать равную криптостойкость (обеспечение линейного (однородного) пространства ключей);

– время шифрования не должно быть большим;

– стоимость шифрования должна быть согласована со стоимостью закрываемой информации.

1.4. КРАТКИЕ СВЕДЕНИЯ О КРИПТОАНАЛИЗЕ

Знание некоторых положений криптоанализа необходимо для глубокого понимания криптографии.

Главным действующим лицом в криптоанализе выступает *нарушитель* (или *криптоаналитик*). Под ним понимают лицо (группу лиц), целью которых является прочтение или подделка защищенных криптографическими методами сообщений.

В отношении нарушителя принимается ряд допущений, которые, как правило, кладутся в основу математических или иных моделей:

1. Нарушитель знает алгоритм шифрования (или выработки ЭЦП) и особенности его реализации в конкретном случае, но не знает секретного ключа.

2. Нарушителю доступны все зашифрованные тексты. Нарушитель может иметь доступ к некоторым исходным текстам, для которых известны соответствующие им зашифрованные тексты.

3. Нарушитель имеет в своем распоряжении вычислительные, людские, временные и иные ресурсы, объем которых оправдан потенциальной ценностью информации, которая будет добыта в результате криптоанализа.

Попытку прочтения или подделки зашифрованного сообщения, вычисления ключа методами криптоанализа называют *криптоатакой* или *атакой на шифр.* Удачную криптоатаку называют *взломом.*

Криптостойкостью называется характеристика шифра, определяющая его стойкость к расшифрованию без знания ключа (т.е. криптоатаке). Показатель криптостойкости – главный параметр любой криптосистемы. В качестве показателя криптостойкости можно выбрать:

– количество всех возможных ключей или вероятность подбора ключа за заданное время с заданными ресурсами;

– количество операций или время (с заданными ресурсами), необходимое для взлома шифра с заданной вероятностью;

– стоимость вычисления ключевой информации или исходного текста.

Все эти показатели должны учитывать также уровень возможной криптоатаки.

Однако следует понимать, что эффективность защиты информации криптографическими методами зависит не только от криптостойкости шифра, но и от множества других факторов, включая вопросы реализации криптосистем в виде устройств или программ. При анализе криптостойкости шифра необходимо учитывать и *человеческий фактор.* Например, подкуп конкретного человека, в руках которого сосредоточена необходимая информация, может стоить на несколько порядков дешевле, чем создание суперкомпьютера для взлома шифра

Современный криптоанализ опирается на такие математические науки как теория вероятностей и математическая статистика, алгебра, теория чисел, теория алгоритмов и ряд других. Все методы криптоанализа в целом укладываются в четыре направления.

1. *Статистический криптоанализ* – исследует возможности взлома криптосистем на основе изучения статистических закономерностей исходных и зашифрованных сообщений. Его применение осложнено тем, что в

реальных криптосистемах информация перед шифрованием подвергается сжатию (превращая исходный текст в случайную последовательность символов), или в случае гаммирования используются псевдослучайные последовательности большой длины.

2. *Алгебраический криптоанализ* – занимается поиском математически слабых звеньев криптоалгоритмов. Например, в 1997 г. в эллиптических системах был выявлен класс ключей, существенно упрощавший криптоанализ.

3. *Дифференциальный* (*или разностный*) *криптоанализ* – основан на анализе зависимости изменения шифрованного текста от изменения исходного текста. Впервые использован Мерфи, улучшен Бихэмом и Шамиром для атаки на *DES*.

4. *Линейный криптоанализ* – метод, основанный на поиске линейной аппроксимации между исходным и шифрованным текстом. Предложенный Мацуи, также впервые был применен при взломе *DES*. Как и дифференциальный анализ в реальных криптосистемах может быть применен только для анализа отдельных блоков криптопреобразований.

Опыт взломов криптосистем (в частности, конкурсов, которые регулярно устраивает *RSA Data Security*) показывает, что главным методом остается "лобовая" атака – проба на ключ. Также как показывает опыт криптосистемы больше страдают от небрежности в реализации.

Принято различать несколько *уровней криптоатаки* в зависимости от объем информации, доступной криптоаналитику. Можно выделить три уровня криптоатаки по нарастанию сложности.

1. Атака по шифрованному тексту (Уровень КА1) – нарушителю доступны все или некоторые зашифрованные сообщения.

2. Атака по паре "исходный текст – шифрованный текст" (Уровень КА2) – нарушителю доступны все или некоторые зашифрованные сообщения и соответствующие им исходные сообщения.

3. Атака по выбранной паре "исходный текст – шифрованный текст" (Уровень КА3) – нарушитель имеет возможность выбирать исходный текст, получать для него шифрованный текст и на основе анализа зависимостей между ними вычислять ключ.

Все современные криптосистемы обладают достаточной стойкостью даже к атакам уровня КА3, т.е. когда нарушителю доступно по сути шифрующее устройство.

1.5. КЛАССИФИКАЦИЯ МЕТОДОВ КРИПТОГРАФИЧЕСКОГО ЗАКРЫТИЯ ИНФОРМАЦИИ

В настоящее время известно большое число методов криптографического закрытия информации. Классификация методов шифрования (криптоалгоритмов) может быть осуществлена по следующим признакам:

- *по типу ключей*: симметричные криптоалгоритмы; асимметричные криптоалгоритмы;
- *по размеру блока информации*: потоковые шифры; блочные шифры;
- *по характеру воздействий, производимых над данными*: метод замены (перестановки), метод подстановки; аналитические методы, аддитивные методы (гаммирование), комбинированные методы.

Кодирование может быть смысловое, символьное, комбинированное.

Закрытие информации другими способами может достигаться с помощью стеганографии, сжатия/расширения, рассечения/разнесения.

Одна из схем классификации методов криптографического закрытия информации приведена на рис. 1.1.

Контрольные вопросы

1. Назовите и охарактеризуйте основные этапы развития криптографии.
2. Дайте определение криптологии, криптографии и криптоанализа.
3. Сформулируйте основные направления в современной криптографии.
4. Дайте определение основных понятий криптографической защиты информации.
5. Сформулируйте основные требования к криптосистемам.
6. **Приведите классификацию методов криптографического преобразования информации.**

7. Каковы перспективы криптозащиты информации в КС.

2. СИММЕТРИЧНЫЕ КРИПТОСИСТЕМЫ

2.1. ОСНОВЫ ТЕОРИИ К. ШЕННОНА

Клод Шеннон рассмотрел модель (рис. 2.1), в которой источник сообщений порождает открытый текст *X*. Источник ключей генерирует ключ *Z*.

Шифратор преобразовывает открытый текст *X* с помощью ключа *Z* в шифртекст *Y*: $Y = TzX$.

Дешифратор, получив зашифрованное сообщение *Y*, выполняет обратную операцию: $X = Tx(-1)\ Y$.

Задачей криптоаналитика противника является получение открытого текста и ключа на основе анализа шифртекста.

Шеннон рассмотрел вопросы теоретической и практической секретности. Для определения теоретической секретности Шеннон сформулировал следующие вопросы:

1. Насколько устойчива система, если криптоаналитик противника не ограничен временем и обладает всеми необходимыми средствами для анализа криптограмм?

2. Имеет ли криптограмма единственное решение?

3. Какой объем шифртекста необходимо перехватить криптоаналитику, чтобы решение стало единственным?

Для ответа на эти вопросы Шеннон ввел понятие совершенной секретности с помощью следующего условия: для всех *Y* апостериорные вероятности равны априорным вероятностям, т.е. перехват зашифрованного сообщения не дает криптоаналитику противника никакой информации. По теореме Байеса

$$P_y(X) = P(X)P_x(Y)/P(Y),$$

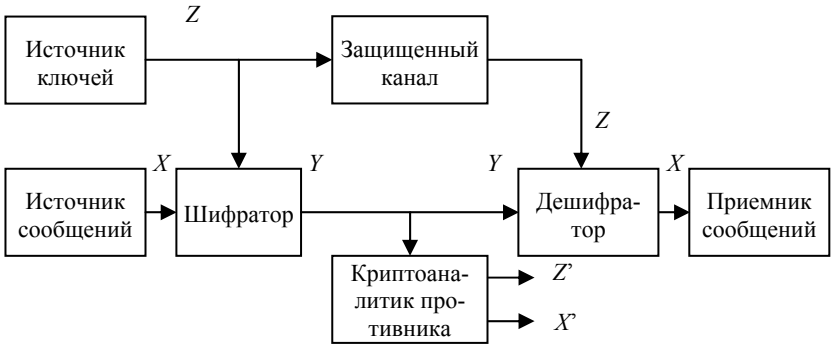


Рис. 2.1. Общая схема передачи шифрованных сообщений

где $P(X)$ – априорная вероятность сообщения *X*; $P_x(Y)$ – условная вероятность криптограммы *Y* при условии, что выбрано сообщение *X*, т.е. сумма вероятностей всех тех ключей, которые переводят сообщение *X* в криптограмму *Y*; $P(Y)$ – вероятность получения криптограммы *Y*; $P_y(X)$ – апостериорная вероятность сообщения *X* при условии, что перехвачена криптограмма *Y*. Для совершенной секретности значения $P_y(X)$ и $P(X)$ должны быть равны для всех *X* и *Y*.

Для противодействия методам статистического анализа криптограмм Шеннон предложил использовать два метода: рассеивание и перемешивание.

2.2. СИММЕТРИЧНЫЕ МЕТОДЫ ШИФРОВАНИЯ

2.2.1. ОСНОВНЫЕ КЛАССЫ СИММЕТРИЧНЫХ КРИПТОСИСТЕМ

Под симметричными криптографическими системами понимаются такие криптосистемы, в которых для шифрования и расшифрования используется один и тот же ключ, хранящийся в секрете (рис. 2.2).

Для пользователей это означает, что прежде, чем начать использовать систему, необходимо получить общий секретный ключ так, чтобы исклю-

чить к нему доступ потенциального злоумышленника. Все многообразие симметричных криптосистем основывается на следующих базовых классах.

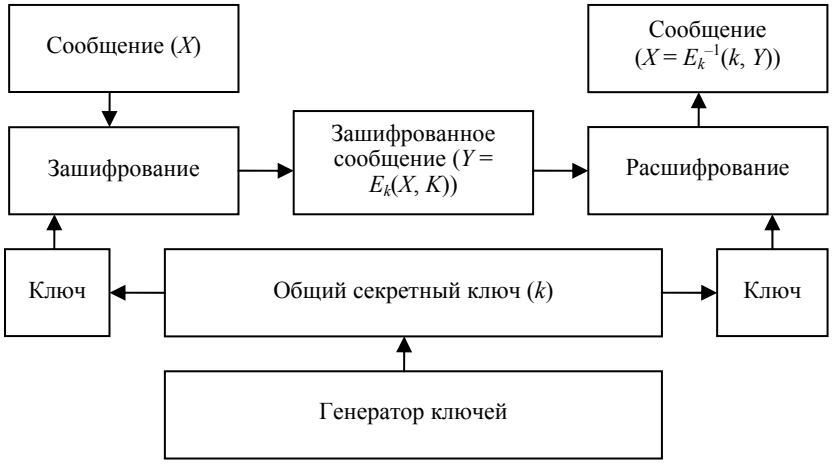


Рис. 2.2. Использование симметричного метода шифрования

Моно- и многоалфавитные подстановки. Моноалфавитные подстановки – это наиболее простой вид преобразований, заключающийся в замене символов исходного текста на другие (того же алфавита) по более или менее сложному правилу. В случае моноалфавитных подстановок каждый символ исходного текста преобразуется в символ шифрованного текста по одному и тому же закону. При многоалфавитной подстановке закон преобразования меняется от символа к символу. Один и тот же шифр может рассматриваться и как моно- и как многоалфавитный в зависимости от определяемого алфавита. Например, шифр Плейфера (подстановка биграмм) с точки зрения обычного алфавита является моноалфавитным, а с точки зрения алфавита биграмм – многоалфавитным.

Перестановки – также несложный метод криптографического преобразования, заключающийся в перестановке местами символов исходного текста по некоторому правилу. Шифры перестановок в настоящее время не используются в чистом виде, так как их криптостойкость недостаточна.

Блочные шифры – семейство обратимых преобразований блоков (частей фиксированной длины) исходного текста. Фактически блочный шифр – это система подстановки на алфавите блоков (она может быть моно- или многоалфавитной в зависимости от *режима* блочного шифра). В настоящее время блочные шифры наиболее распространены на практике. Российский и американский стандарты шифрования (ГОСТ 28147–89 "Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования" и *DES*) относятся именно к этому классу шифров.

Гаммирование – преобразование исходного текста, при котором символы исходного текста складываются (по модулю, равному мощности алфавита) с символами псевдослучайной последовательности, вырабатываемой по некоторому правилу. Собственно говоря, гаммирование нельзя целиком выделить в отдельный класс криптографических преобразований, так как эта псевдослучайная последовательность может вырабатываться, например, с помощью блочного шифра. В случае, если последовательность является истинно случайной (например, снятой с физического датчика) и каждый ее фрагмент используется только один раз, то имеет место криптосистема с одноразовым ключом.

Далее речь будет идти о защите сообщений, хотя события могут развиваться не только в пространстве, но и во времени, когда шифруются и расшифровываются никуда не перемещающиеся файлы.

Основным недостатком симметричного шифрования является то, что секретный ключ должен быть известен и отправителю, и получателю. С одной стороны, это ставит новую проблему рассылки ключей. С другой стороны, получатель на основании наличия шифрованного и расшифрованного сообщения не может доказать, что он получил это сообщение от конкретного отправителя, поскольку такое же сообщение он мог сгенерировать

и сам.

2.2.2. ОБЩИЕ СВЕДЕНИЯ О БЛОЧНЫХ ШИФРАХ

Под *N*-разрядным блоком будем понимать последовательность из нулей и единиц длины *N* [ГОСТ Р 34.10–2001]:

$$x = (x_0, x_1, ..., x_{N-1}) \in Z_{2,N},$$

где x в $Z_{2,N}$ можно интерпретировать как вектор и как двоичное представление целого числа:

$$\|x\| = \sum_{i=0}^{N-1} x_i 2^{N-i-1}.$$

Например, если $N = 4$, то

(0,0,0,0)⇨0	(0,0,0,1)⇨1	(0,0,1,0)⇨2	(0,0,1,1)⇨3
(0,1,0,0)⇨4	(0,1,0,1)⇨5	(0,1,1,0)⇨6	(0,1,1,1)⇨7
(1,0,0,0)⇨8	(1,0,0,1)⇨9	(1,0,1,0)⇨10	(1,0,1,1)⇨11
(1,1,0,0)⇨12	(1,1,0,1)⇨13	(1,1,1,0)⇨14	(1,1,1,1)⇨15

Блочным шифром будем называть элемент

$$\pi \in SYM(Z_{2,N}); \quad \pi : x \to y = \pi(x),$$

где $x = (x_0, x_1, ..., x_{N-1})$, $y = (y_0, y_1, ..., y_{N-1})$. Хотя блочные шифры являются частными случаями подстановок их следует рассматривать особо, так как, во-первых, большинство симметричных шифров, используемых в системах передачи информации, являются блочными и, во-вторых, блочные шифры удобнее всего описывать в алгоритмическом виде, а не как обычные подстановки.

Если исходный текст шифруется подстановкой π, выбранной из полной симметрической группы, то злоумышленник, изучающий соответствие между подмножествами исходного и шифрованного текстов

$$x_i \leftrightarrow y_i, \quad 0 \le i \le m,$$

не в состоянии на основе этой информации определить исходный текст, соответствующий $y \notin \{y_i\}$.

Если для шифрования исходного текста используется подсистема π из $\Pi \in SYM(Z_{2,N})$, то получающуюся в результате систему подстановок Π называют системой *блочных шифров* или *системой блочных подстановок*. Блочный шифр представляет собой частный случай моноалфавитной подстановки с алфавитом $Z_{2^N} = Z_{2,N}$. Если информация исходного текста не может быть представлена *N*-разрядными блоками, как в случае стандартного алфавитно-цифрового текста, то первое, что нужно сделать, это перекодировать исходный текст именно в этот формат, причем с практической точки зрения неважно, какой из способов перекодирование был применен.

Ключевой системой блочных шифров является подмножество Π[*K*] симметрической группы $SYM(Z_{2,N})$

$$\Pi[K] = \{\pi\{k\} : k \in K\},$$

индексируемое по параметру $k \in K$; k является ключом, а K – пространством ключей. При этом не требуется, чтобы различные ключи соответствовали различным подстановкам $Z_{2,N}$.

Ключевая система блочных шифров Π[*K*] используется следующим образом. Пользователь *i* и пользователь *j* некоторым образом заключают соглашение относительно ключа *k* из *K*, выбирая, таким образом, элемент из Π[*K*] и передавая текст, зашифрованный с использованием выбранной подстановки. Запись $y = \pi\{k, x\}$ используется для обозначения *N*-разрядного блока шифрованного текста, который получен в результате шифрования *N*-разрядного блока исходного текста x с использованием под-

становки $\pi\{k\}$, соответствующей ключу k. Предположим, что злоумышленнику:

- известно пространство ключей K;
- известен алгоритм определения подстановки $\pi\{k\}$ по значению ключа k;
- неизвестно, какой именно ключ k выбрал пользователь.
- Какими возможностями располагает злоумышленник? Он может:
- получить ключ вследствие небрежности пользователя i или пользователя j;
- перехватить шифрованный текст y, передаваемый пользователем i пользователю j, и производить пробы на все возможные ключи из K до получения читаемого сообщения исходного текста;
- получить соответствующие исходный и шифрованный тексты $x \to y$ и воспользоваться методом пробы на ключ;
- получить соответствующие исходный и шифрованный тексты и исследовать соотношение исходного текста x и шифрованного текста y для определения ключа k;
- организовать каталог N-разрядных блоков с записью частот их появления в исходном или шифрованном тексте. Каталог дает возможность производить поиск наиболее вероятных слов, используя, например, следующую информацию: листинг на языке ассемблера характеризуется сильно выраженным структурированным форматом, цифровое представление графической и звуковой информации имеет ограниченный набор знаков.

Предположим, что $N = 64$ и каждый элемент $SYM(Z_{2,N})$ может быть использован как подстановка, так что $K = SYM(Z_{2,N})$. Тогда:

- существует 2^{64} 64-разрядных блоков, но злоумышленник не может поддерживать каталог с $2^{64} \approx 1{,}8 \cdot 10^{19}$ строками;
- проба на ключ при числе ключей, равном $(2^{64})!$, практически невозможна; соответствие исходного и шифрованного текстов для некоторых N-разрядных блоков $\pi\{k,\ x_i\}$, $0 \le i < m$, не дает злоумышленнику информации относительно значения $\pi\{k,\ x\}$ для $x \notin \{x_i\}$.

Системы шифрования с блочными шифрами, алфавитом $Z_{2,64}$ и пространством ключей $K = SYM(Z_{2,64})$ являются неделимыми в том смысле, что поддержание каталога частот появления букв для 64-разрядных блоков или проба на ключ при числе ключей 2^{64} выходит за пределы возможностей злоумышленника. Следует сравнить эту проблему с той задачей, с которой сталкивается злоумышленник в процессе криптоанализа текста, зашифрованного подстановкой Цезаря с алфавитом {А, ..., Я, _}; для определения ключа подстановки Цезаря требуется лишь $\log_2 32 = 5$ бит, в то время как для пространства ключей $K = SYM(Z_{2,64})$ требуется 2^{64} бит.

К сожалению, разработчик и злоумышленник находятся в одинаковом положении: разработчик не может создать систему, в которой были бы реализованы все $2^{64}!$ подстановок $SYM(Z_{2,64})$, а злоумышленник не может испытать такое число ключей. Остается согласиться с тем, что не каждый элемент из $SYM(Z_{2,64})$ будет использован в качестве подстановки.

Таким образом, требования к хорошему блочному шифру формулируются следующим образом. Необходимы:

- достаточно большое N (64 или более) для того, чтобы затруднить составление и поддержание каталога (в новом стандарте шифрования США N не менее 128);
- достаточно большое пространство ключей для того, чтобы исключить возможность подбора ключа;
- сложные соотношения $\pi\{k,\ x\} : x \to y = \pi\{k,\ x\}$ между исходным и шифрованным текстами с тем, чтобы аналитические и (или) статистические методы определения исходного текста и (или) ключа на основе соответствия исходного и шифрованного текстов были бы по возможности нереализуемы.

2.1. Характеристики криптографических алгоритмов

Название алгоритма	Автор	Размер блока	Длина ключа

IDEA	*Xuejia Lia and James Massey*	64 бита	128 бит
CAST128		64 бита	128 бит
BlowFish	*Bruce Schneier*	64 бита	128…448 бит
ГОСТ	НИИ	64 бита	256 бит
TwoFish	*Bruce Schneier*	128 бит	128…256 бит
MARS	Корпорация *IBM*	128 бит	128…1048 бит

2.2.3. ГЕНЕРИРОВАНИЕ БЛОЧНЫХ ШИФРОВ

Одним из наиболее распространенных способов задания блочных шифров является использование так называемых сетей Фейстеля. Сеть Фейстеля представляет собой общий метод преобразования произвольной функции (обычно называемой *F*-функцией) в перестановку на множестве блоков. Эта конструкция была изобретена Хорстом Фейстелем и была использована в большом количестве шифров, включая *DES* и ГОСТ 28147–89. *F*-функция, представляющая собой основной строительный блок сети Фейстеля, всегда выбирается нелинейной и практически во всех случаях необратимой.

Формально *F*-функцию можно представить в виде отображения

$$F : Z_{2,N/2} Z_{2,k} \to Z_{2,N/2},$$

где N – длина преобразуемого блока текста (должна быть четной); k – длина используемого блока ключевой информации.

Пусть теперь X – блок текста, представим его в виде двух подблоков одинаковой длины $X = \{A, B\}$. Тогда одна итерация (или раунд) сети Фейстеля определяется как

$$X_{i+1} = B_1 \| (F(B_1, k_i) \oplus A_i),$$

где $X_i = \{A_i, \ B_i\}$, $\|$ операция конкатенации, а $\oplus$ – побитовое исключающее ИЛИ.

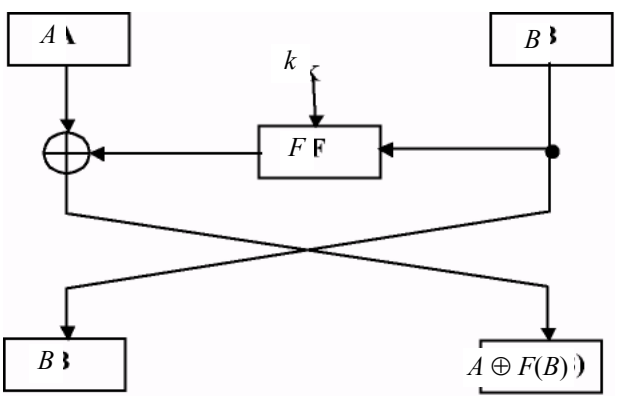


Рис. 2.3. Структура итерации сети Фейстеля

Структура итерации сети Фейстеля представлена на рис. 2.3. Сеть Фейстеля состоит из некоторого фиксированного числа итераций, определяемого соображениями стойкости разрабатываемого шифра, при этом на последней итерации перестановка местами половин блока текста не производится, так как это не влияет на стойкость шифра.

Данная структура шифров обладает рядом достоинств, а именно:

- процедуры шифрования и расшифрования совпадают, с тем исключением, что ключевая информация при расшифровании используется в обратном порядке;
- для построения устройств шифрования можно использовать те же блоки в цепях шифрования и расшифрования.

Недостатком является то, что на каждой итерации изменяется только половина блока обрабатываемого текста, что приводит к необходимости увеличивать число итераций для достижения требуемой стойкости.

В отношении выбора *F*-функции каких-то четких стандартов не суще-

ствует, однако, как правило, эта функция представляет собой последовательность зависящих от ключа нелинейных замен, перемешивающих перестановок и сдвигов.

Другим подходом к построению блочных шифров является использование обратимых зависящих от ключа преобразований. В этом случае на каждой итерации изменяется весь блок и, соответственно, общее количество итераций может быть сокращено. Каждая итерация представляет собой последовательность преобразований (так называемых "слоев"), каждое из которых выполняет свою функцию. Обычно используются слой нелинейной обратимой замены, слой линейного перемешивания и один или два слоя подмешивания ключа K недостаткам данного подхода можно отнести то, что для процедур шифрования и расшифрования в общем случае нельзя использовать одни и те же блоки, что увеличивает аппаратные и/или программные затраты на реализацию.

2.3. АЛГОРИТМЫ БЛОЧНОГО ШИФРОВАНИЯ

2.3.1. АЛГОРИТМ *DES* И ЕГО МОДИФИКАЦИИ

Американский стандарт криптографического закрытия данных *DES* (*Data Encryption Standard*), принятый в 1978 г., является типичным представителем семейства блочных шифров и одним из наиболее распространенных криптографических стандартов на шифрование данных, применяемых в США. Этот шифр допускает эффективную аппаратную и программную реализацию, причем возможно достижение скоростей шифрования до нескольких мегабайт в секунду.

Первоначально метод, лежащий в основе данного стандарта, был разработан фирмой *IBM* для своих целей. Он был проверен Агентством Национальной Безопасности США, которое не обнаружило в нем статистических или математических изъянов.

Стандарт *DES* используется федеральными департаментами и агентствами для защиты всех достаточно важных данных в компьютерах (исключая некоторые данные, методы защиты которых определяются специальными актами). Его применяют многие негосударственные институты, в том числе большинство банков и служб обращения денег. Оговоренный в стандарте алгоритм криптографической защиты данных опубликован для того, чтобы большинство пользователей могли использовать проверенный и апробированный алгоритм с хорошей криптостойкостью. Однако, с одной стороны, публикация алгоритма нежелательна, поскольку может привести к попыткам дешифрования закрытой информации, но, с другой стороны, это не столь существенно поскольку стандартный алгоритм шифрования данных должен обладать такими характеристиками, чтобы его опубликование не сказалось на его криптостойкости.

DES имеет блоки по 64 бит и основан на 16-кратной перестановке данных, также для шифрования использует ключ в 56 бит. Существует несколько режимов *DES*: *Electronic Code Book* (*ECB*) и *Cipher Block Chaining* (*CBC*).

56 бит – это 8 семибитовых *ASCII* символов, т.е. пароль не может быть больше чем 8 букв. Если вдобавок использовать только буквы и цифры, то количество возможных вариантов будет существенно меньше максимально возможных 2^{56}.

Шифр *DES* представляет собой результат 33 отображений:

$$X_{i+1} = B_1 \| (F(B_1, k_i) \oplus A_i),$$

где *IP* (*Initial Permutation* – исходная перестановка) представляет собой проволочную коммутацию с инверсией IP^{-1}:

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,

композиция $\Theta \times \pi_{T_i}$, где Θ – перестановка местами правой и левой половин блока данных, представляет собой одну итерацию Фейстеля. В последнем цикле шифрования по алгоритму *DES* перестановка местами половин

блока не производится.

Подстановки π_{T_i}, $1< i < 16$, описываются следующим образом (см. рис. 2.4).

Шаг 1. На i-м цикле входной блок x^i длиной 64 символа

$$x^i = (x_{i,0},\ x_{i,1},\ ...,\ x_{i,63})$$

делится на два блока по 32 символа:

$$X = (x_{i,0},\ x_{i,1},\ ...,\ x_{i,31}) \text{ и } X' = (x'_{i,0},\ x'_{i,1},\ ...,\ x'_{i,31}).$$

Правый блок X' разбивается на восемь блоков по четыре символа:

$x'_{i,0}$	$x'_{i,1}$	$x'_{i,2}$	$x'_{i,3}$
$x'_{i,4}$	$x'_{i,5}$	$x'_{i,6}$	$x'_{i,7}$
$x'_{i,8}$	$x'_{i,9}$	$x'_{i,10}$	$x'_{i,11}$
$x'_{i,12}$	$x'_{i,13}$	$x'_{i,14}$	$x'_{i,15}$
$x'_{i,16}$	$x'_{i,17}$	$x'_{i,18}$	$x'_{i,19}$
$x'_{i,20}$	$x'_{i,21}$	$x'_{i,22}$	$x'_{i,23}$
$x'_{i,24}$	$x'_{i,25}$	$x'_{i,26}$	$x'_{i,27}$
$x'_{i,28}$	$x'_{i,29}$	$x'_{i,30}$	$x'_{i,31}$

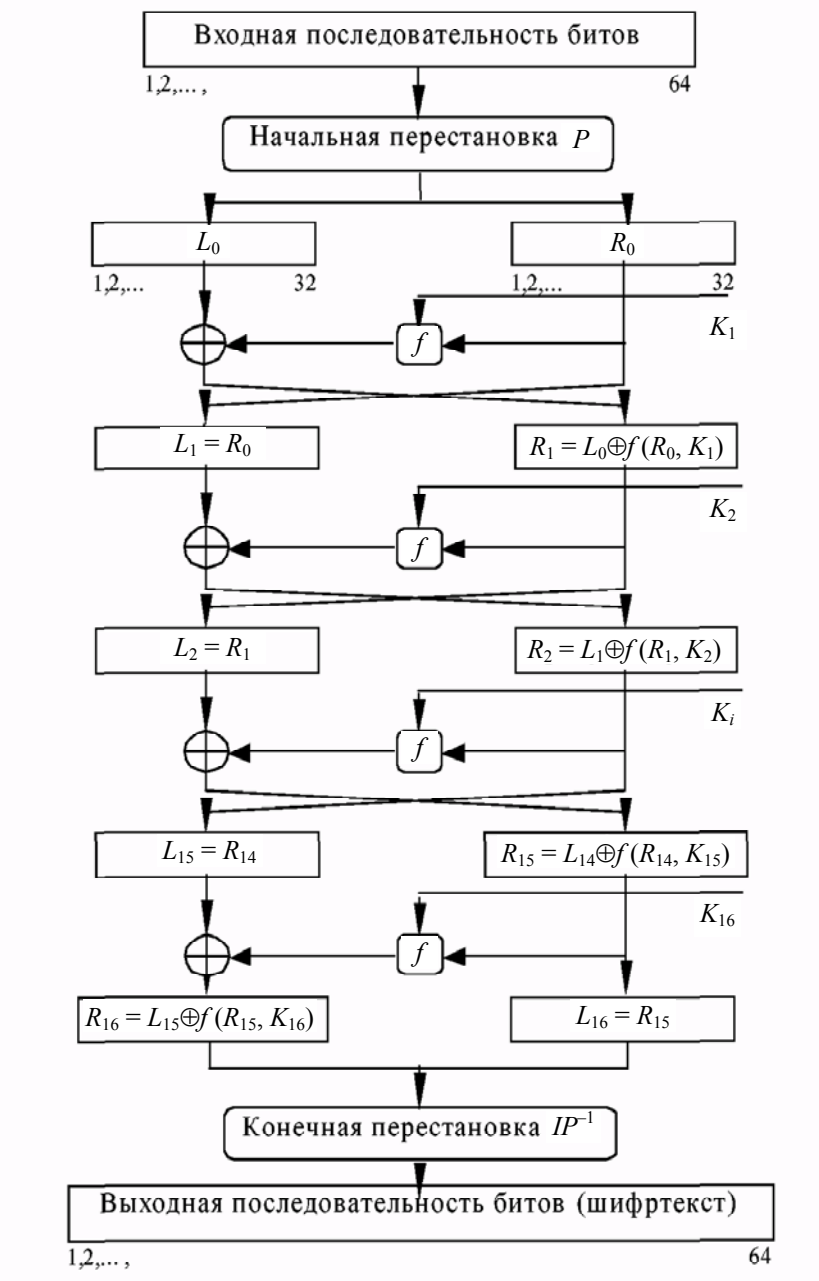


Рис. 2.4. Схема алгоритма шифрования *DES*

Эти восемь блоков путем копирования крайних элементов преобразуются в восемь блоков из шести символов:

$x'_{i,31}$	$x'_{i,0}$	$x'_{i,1}$	$x'_{i,2}$	$x'_{i,3}$	$x'_{i,4}$
$x'_{i,3}$	$x'_{i,4}$	$x'_{i,5}$	$x'_{i,6}$	$x'_{i,7}$	$x'_{i,8}$
$x'_{i,7}$	$x'_{i,8}$	$x'_{i,9}$	$x'_{i,10}$	$x'_{i,11}$	$x'_{i,12}$
$x'_{i,11}$	$x'_{i,12}$	$x'_{i,13}$	$x'_{i,14}$	$x'_{i,15}$	$x'_{i,16}$
$x'_{i,15}$	$x'_{i,16}$	$x'_{i,17}$	$x'_{i,18}$	$x'_{i,19}$	$x'_{i,20}$
$x'_{i,19}$	$x'_{i,20}$	$x'_{i,21}$	$x'_{i,22}$	$x'_{i,23}$	$x'_{i,24}$
$x'_{i,23}$	$x'_{i,24}$	$x'_{i,25}$	$x'_{i,26}$	$x'_{i,27}$	$x'_{i,28}$
$x'_{i,27}$	$x'_{i,28}$	$x'_{i,29}$	$x'_{i,30}$	$x'_{i,31}$	$x'_{i,0}$

Шаг 2. На i-циклической итерации 48 разрядов ключа

$$(k_{i,0}, k_{i,1}, ..., k_{i,48})$$

поразрядно суммируются (по модулю 2) с полученными выше 48 разрядами данных.

Шаг 3. j-й блок из шести символов ($0 < j < 8$) подается на вход блока подстановки (S-бокс) $S[j]$, который имеет шестиразрядный вход и четырехразрядный выход и представляет собой четыре преобразования из $Z_{2,4}$ в $Z_{2,4}$; два крайних разряда входного блока служат для выборки одного из этих преобразований. Каждая из восьми подстановок $S[0]$, $S[1]$, ..., $S[7]$ осуществляется с использованием четырех строк и 16 столбцов матрицы с элементами {0, 1, ..., 15}. Каждый из массивов размерностью 4 × 16 определяет подстановку на множестве $Z_{2,4}$ следующим образом. Если входом является блок из шести символов (z_0, z_1, z_2, z_3, z_4, z_5), то две крайние позиции (z_0, z_5) интерпретируются как двоичное представление целых чисел из набора {0, 1, 2, 3}. Эти целые определяют номер строки (от 0 до 3). Оставшиеся четыре символа (z_1, z_2, z_3, z_4) интерпретируются как двоичное представление целых чисел из набора {0, 1, ..., 15} и служат для определения столбца в массиве (от 0 до 15). Таким образом, входной блок (0, 0, 1, 0, 1, 1) соответствует строке 1 и столбцу 5.

Шаг 4. 32 разряда, составляющие выход S-бокса, подаются на вход блока проволочной коммутации (P-бокса):

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

Шаг 5. Компоненты правого входного 32-разрядного блока X', преобразованного в $T(X')$, поразрядно суммируются по модулю 2 с компонентами левого входного 32-разрядного блока X.

На каждой итерации используется 48-разрядный подключ $(k_{i,0}, k_{i,1}, ..., k_{i,48})$. Поскольку входным ключом *DES* является 56-разрядный блок $k = (k_{i,0}, k_{i,1}, ..., k_{i,56})$, то каждый его разряд используется многократно.

Какие именно разряды ключа используются на i-циклической итерации, определяется по следующему алгоритму:

- прежде всего 64 разряда ключа преобразуются в 56 путем выбрасывания каждого восьмого бита (который может использоваться для контроля целостности ключа);
- производится начальная перестановка КР-1 56-разрядного ключа пользователя $k = (k_{i,0}, k_{i,1}, ..., k_{i,56})$:

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,
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.

Получаемый в результате 56-разрядный блок рассматривается как два 28-разрядных блока: левый – C_0 и правый – D_0;

- производится левый циклический сдвиг блоков C_0 и D_0 $s[1]$ раз для получения блоков C_1 и D_1;
- из сцепления блоков (C_1 и D_1) выбираются 48 разрядов с помощью перестановки КР-2. Эти разряды используются на первой итерации;

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

используемые на i-й циклической итерации разряды ключа определяются методом индукции. Для получения блоков C_i и D_i производим левый циклический сдвиг блоков C_{i-1} и D_{i-1} на $s[i]$ позиций:

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
S	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

и вновь применяем КР-2 для получения очередной порции ключа.

Инверсией *DES* (обеспечивающей расшифрование зашифрованных посредством *DES* данных) является

$$DES = IP^{-1} \times \pi_{T_1} \times \Theta \times ... \pi_{T_{16}} \times IP .$$

Расшифрование зашифрованного посредством *DES* текста осуществляется с использованием тех же блоков благодаря обратимости преобразо-

вания.

Анализ алгоритма *DES* на эффективность показывает, что поскольку длина блоков исходного текста равна 64, поддержка каталогов частот использования блоков является для злоумышленника задачей, выходящей за пределы современных технических возможностей.

Однако, данный алгоритм, являясь первым опытом стандарта шифрования имеет ряд недостатков. За время, прошедшее после создания *DES*, компьютерная техника развилась настолько быстро, что оказалось возможным осуществлять исчерпывающий перебор ключей и тем самым раскрывать шифр. Стоимость этой атаки постоянно снижается. В 1998 г. была построена машина стоимостью около 100 000 долларов, способная по данной паре <исходный текст, шифрованный текст> восстановить ключ за среднее время в трое суток. Таким образом, *DES*, при его использовании стандартным образом, уже стал далеко не оптимальным выбором для удовлетворения требованиям скрытности данных.

Большое количество предложений по усовершенствованию *DES* отчасти компенсируют указанные недостатки, рассмотрим два из них.

Наиболее широко известным предложением по усилению *DES* является так называемый "тройной *DES*", одна из версий которого определяется формулой

$$EDE3_{k1k2k3}(x) = DES_{k3}(DES^{-1}_{k2}(DES_{k1}(x))).$$

То есть, ключ для *EDE*3 имеет длину 56 × 3 = 168 бит, и шифрование 64-битового блока осуществляется шифрованием с одним подключом, расшифрованием с другим и затем шифрованием с третьим. (Причина, по которой вторым шагом является DES^{-1}_{k2}, а не DES_{k2}, является совместимость с *DES*: если выбрать $K = k,k,k$, то $EDE3_k = DES_k$. Причина использования *DES* три раза вместо двух заключается в существовании атаки "встреча в середине" на двойной *DES*).

Проблема с тройным *DES* состоит в том, что он гораздо медленнее, чем сам *DES*, – его скорость составляет ровно одну треть исходной. При использовании *EDE*3 в режиме сцепления блоков это замедление скажется как на аппаратном, так и на программном (даже если попытаться компенсировать его дополнительной аппаратной частью) уровнях. Во многих случаях такое падение производительности неприемлемо.

В 1984 г. Рон Ривест предложил расширение *DES*, называемое *DESX* (*DES eXtended*), свободное от недостатков тройного *DES*. *DESX* определяется как

$$DES_{k,k1,k2} = k_2 \otimes DES_k(k_1 \oplus x).$$

Ключ *DESX* $K = k,k_1,k_2$ состоит из 54 + 64 + 64 = 184 бит и включает три различных подключа: ключ "*DES*" k, предварительный "зашумляющий" ключ k_1 и завершающий "зашумляющий" ключ k_2.

Для шифрования блока сообщения сложим его поразрядно по модулю 2 с k_1, шифруем его алгоритмом DES с ключом k и вновь поразрядно складываем его по модулю 2 с k_2. Таким образом, затраты *DESX* на шифрование блока всего на две операции сложения по модулю 2 больше, чем затраты исходного алгоритма.

В отношении *DESX* замечательно то, что эти две операции "исключающее ИЛИ" делают шифр гораздо менее уязвимым по отношению к перебору ключей. *DESX* затрудняет получение даже одной пары $\langle x_i,\ DES_K(x_i)\rangle$ в том случае, когда злоумышленник организует атаку на шифр по выбранному исходному тексту, получая множество пар $\langle\ P_j, DES_K(P_j)\rangle$.

DESX предназначался для *увеличения* защищенности *DES* против перебора ключей и *сохранения* его стойкости против других возможных атак. Но *DESX* в действительности также увеличивает стойкость против дифференциального и линейного криптоанализа, увеличивая требуемое количество проб с выбранным исходным текстом до величины, превышающей 2^{60}. Дальнейшее увеличение стойкости против этих атак может быть достигнуто заменой в *DESX* операции "исключающее ИЛИ" на сложение, как это

было сделано в

$$DES - PEP_{k,k1,k2} = k_2 + DES_k(k_1 + x),$$

где сложение определяется следующим образом:

$$L.R + L'.R' = (L \Diamond L').(R \Diamond R'), \left|L\right| = \left|R\right| = \left|L'\right| = \left|R'\right| = 32,$$

а ◊ обозначает сложение по модулю 2^{32}.

Сказанное не означает, что невозможно построить машину, раскрывающую *DESX* за приемлемое время. Но оно подразумевает, что такая машина должна использовать какую-либо радикально новую идею. Это не может быть машина, реализующая перебор ключей в общепринятом смысле.

Таким образом, практически во всех отношениях *DESX* оказывается лучше *DES*. Этот алгоритм прост, совместим с *DES*, эффективно реализуется аппаратно, может использовать существующее аппаратное обеспечение *DES* и в его отношении было доказано, что он увеличивает стойкость к атакам, основанным на переборе ключей.

2.3.2. СТАНДАРТ *AES*. АЛГОРИТМ *RIJNDAEL*

В конце 1996 г. Национальным институтом стандартов США (*NIST*) был объявлен конкурс на создание нового общенационального стандарта шифрования, который должен прийти на замену *DES*. Разрабатываемому стандарту было присвоено рабочее наименование *AES* (*Advanced Encryption Standard*).

2 октября 2000 г. в качестве предлагаемого стандарта был выбран алгоритм *Rijndael* ("Рейндал"), который разработан Винсентом Райманом (*Vincent Rijman*) и Йоан Дамен (*Joan Daemen*) и представляет собой алгоритм, не использующий сети Фейстеля.

При описании алгоритма используется поле Галуа $GF(2^S)$, построенное как расширение поля $GF(2)$ по корням неприводимого многочлена

$$m(x) = x^8 + x^4 + x^3 + x + 1.$$

Данный многочлен выбран из соображений эффективности представления элементов поля. Элементарные операции, использующиеся в алгоритме, выполняются в указанном поле.

Алгоритм *Rijndael* представляет собой блочный шифр с переменной длиной блока и переменной длиной ключа. Длины блока и ключа могут быть выбраны независимо равными 128, 192 или 256 бит. Шифр является последовательностью итераций, выполняемых над некоторой промежуточной структурой, называемой *состоянием.*

Состояние может быть представлено в виде прямоугольного массива байтов. В массиве 4 строки, а число столбцов, обозначаемое как *Nb*, равно длине блока, деленной на 32. Ключ шифрования аналогичным образом представляется в виде прямоугольного байтового массива с 4 строками. Количество столбцов, обозначаемое *Nk*, равно длине ключа, деленной на 32. Входные и выходные значения алгоритма представляются в виде одномерных байтовых массивов соответствующей длины. Состояние и ключевой массив заполняются из этих массивов вначале по столбцам, а затем по строкам. Количество итераций обозначается *Nr* зависит от *Nb* и *Nk* в соответствии со следующей таблицей:

Nr	*Nb* = 4	*Nb* = 6	*Nb* = 8
Nk = 4	10	12	14
Nk = 6	12	12	14
Nk = 8	14	14	14

Блок *нелинейной обратимой байтовой замены* (*S*-бокс), состоящий из двух операций:

- каждый байт заменяется на мультипликативный обратный к нему в поле $GF(2^S)$.
- над каждым байтом выполняется аффинное преобразование в поле $GF(2)$, задаваемое следующим уравнением:

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

Это аффинное преобразование может быть описано в полиномиальном виде как $b(x) = (x^7 + x^6 + x^2 + x) + a(x)\ (x^7 + x^6 + x^5 + x^4 + 1)\ (\mathrm{mod}\ x^8 + 1)$. Полином, на который производится умножение, выбран взаимно простым с модулем, так что умножение является обратимым.

Обратным будет преобразование, состоящее из обратного аффинного преобразования и взятия мультипликативного обратного в $GF(2^S)$.

Следующее преобразование является *циклическим сдвигом влево строк* массива состояния на различную величину. Строка 0 не сдвигается, строка 1 сдвигается на C1 позиций, строка 2 – на C2 и строка 3 – на C3 позиций. Величины сдвига приведены в таблице:

Nb	C1	C2	C3
4	1	2	3
6	1	2	3
8	1	3	4

Обратным преобразованием будет циклический сдвиг строк массива вправо на то же количество *позиций.*

"*Перемешивание столбцов*". В этом преобразовании столбцы массива состояния рассматриваются как полиномы над полем $GF(2^8)$. Преобразование заключается в умножении столбца по модулю $(x^4 + 1)$ на фиксированный полином $c(x)$. Этот полином является взаимно простым $c\ (x^4 + 1)$ и поэтому умножение обратимо. В матричной форме данное преобразование можно представить как

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}.$$

Обратное преобразование представляет собой умножение на полином, мультипликативно обратный к $c\ (x)$ по модулю $(x^4 + 1) – d(x)$.

Преобразование *добавление ключа итерации* осуществляется простым побитовым сложением по модулю 2 каждого байта массива состояния с соответствующим байтом массива ключа. Это преобразование является обратным самому себе.

Ключи итерации получаются из ключа шифрования с помощью *алгоритма обработки ключа,* состоящего из двух компонентов – *расширения ключа* и *выбора ключа итерации*. Основные принципы его построения следующие:

- общее число бит ключей итерации равно длине блока, умноженной на количество итераций плюс один (например, для блока 128 бит и 10 итераций потребуется 1408 бит ключей итерации);
- ключ шифрования расширяется до *расширенного ключа.* Ключи итерации берутся из расширенного ключа следующим образом: первый ключ итерации состоит из первых *Nb* слов, второй – из следующих *Nb* слов и т.д.

Выбор ключа итерации. Ключ итерации с номером i задается словами из буфера расширенного ключа, начиная с $W[Nb\ i]$ и до $W[Nb\ (i + 1)]$ и состоит из трех этапов: начального добавления подключа; $Nr - 1$ итераций; конечной итерации.

2.3.3. АЛГОРИТМ *RC*6

В качестве одного из кандидатов фирмой *RSA Data Security*, *Inc*. был представлен алгоритм *RC*6, прошедший второй тур отбора. В нем предусматривается использование четырех рабочих регистров, а также введена операция целочисленного умножения, позволяющая существенно увеличить возмущения, вносимые каждым циклом шифрования, что приводит к увеличению стойкости и/или возможности сократить число циклов.

*RC*6 является полностью параметризованным алгоритмом шифрования. Конкретная версия *RC*6 обозначается как *RC*6-*w*/*r*/*b,* где w обозначает длину слова в битах, r – ненулевое количество итерационных циклов шифрования, а b – длину ключа в байтах. Во всех вариантах *RC*6-*w*/*r*/*b* работает с четырьмя w-битовыми словами, используя шесть базовых операций, обозначаемых следующим образом:

$a + b$ – целочисленное сложение по модулю 2^w;

$a - b$ – целочисленное вычитание по модулю 2^w;

$a \oplus b$ – побитовое "исключающее ИЛИ" w-битовых слов;

$a \times b$ – целочисленное умножение по модулю 2^w;

$a << b$ – циклический сдвиг w-битового слова влево на величину, заданную $\log_2 w$ младшими битами b;

$a >> b$ – циклический сдвиг w-битового слова вправо на величину, заданную $\log_2 w$ младшими битами b;

Шифрование при помощи *RC*6-*w*/*r*/*b* описывается следующим образом:

Вход: Исходный текст, записанный в 4 w-битовых входных регистрах A, B, C, D; число циклов шифрования r; ключевая таблица $S[0; ..., 2r + 3]$ w-битовых слов.

Выход: Шифрованный текст в регистрах A, B, C, D.

Процедура:

```
B=B + S[0]
D=D + S[1]
for i=1 to r do {
  t=(B×(2B+1))<<log2w
  u=(D×(2D+1))<<log2w
  A=((A⊕t)<<u)+S[2i]
  C=((C⊕u)<<t)+S[2i+1]
   (A; B; C; D) = (B; C; D; A)
}
A=A + S[2r + 2]
C=C + S[2r +3]
```

Расшифрование в этих обозначениях выглядит очень похоже.

Вход: Шифрованный текст, записанный в 4 w-битовых входных регистрах A, B, C, D; число циклов шифрования r; ключевая таблица $S[0; ..., 2r + 3]$ w-битовых слов.

Выход: Исходный текст в регистрах A, B, C, D.

Процедура:

```
C=C–S[2r +3]
A=A–S[2r + 2]
for i=r downto 1 do {
  (A; B; C; D) = (D;A; B; C)
  u=(D×(2D+1))<<log2w
  t=(B×(2B+1))<<log2w
```

```
  C=((C – S[2i+1])>>t)+ ⊕u
  A=((A – S[2i])>>u)+ ⊕t
}
D=D – S[1]
B=B – S[0]
```

Алгоритм вычисления ключей для *RC*6-*w*/*r*/*b* выглядит следующим образом:

Пользователь задает ключ длиной *b* байтов. Достаточное число ненулевых байтов дописываются в конец, чтобы получилось целое число слов. Затем эти байты записываются, начиная с младшего, в массив из *c* слов, т.е. первый байт ключа записывается в *L*[0] и т.д., а *L*[*c* – 1] при необходимости дополняется со стороны старших разрядов нулевыми байтами. В результате работы алгоритма генерации ключей будет вычислено 2*r* + 4 слов, которые будут записаны в массиве *S* [0; ..., 2*r* + 3].

Константы P_{32} = B7E15163h and Q_{32} = 9E3779B9h – это константы, получаемые из двоичного представления *e* – 2, где *e* – основание натуральных логарифмов, и ϕ – 1, где ϕ – золотое сечение, соответственно. Подобные же константы могут быть аналогичным образом получены и для *RC*6 с другим размером слова. Выбор констант является в некотором роде произвольным, и поэтому можно использовать и другие константы, получая при этом "частные" версии алгоритма.

Структура шифра *RC*6 является обобщением сети Фейстеля, но блок текста разбивается не на два, а на четыре подблока, и на каждой итерации изменяются два подблока из четырех. При этом в конце итерации шифрования производится циклический сдвиг подблоков влево (при расшифровании, соответственно, вправо). Однако, такое обобщение привело к тому, что было утеряно свойство инвариантности блоков шифрования и расшифрования, хотя это и не является определяющим в оценке данного алгоритма.

2.3.4. РОССИЙСКИЙ СТАНДАРТ ШИФРОВАНИЯ ГОСТ 28147–89

В Российской Федерации установлен единый стандарт преобразования данных для систем обработки информации в сетях ЭВМ, отдельных вычислительных комплексах и ЭВМ, который определяется ГОСТ 28147–89. Он носит обязательный характер для государственных органов, организаций, предприятий, банковских и иных учреждений, чья деятельность связана с обеспечением информационной безопасности государства. Для других организаций и частных лиц ГОСТ имеет рекомендательный характер. Этот алгоритм криптографического преобразования данных предназначен для аппаратной или программной реализации, удовлетворяет криптографическим требованиям и не накладывает ограничения на степень секретности защищаемой информации.

Данный стандарт формировался с учетом мирового опыта и, в частности, были приняты во внимание недостатки и нереализованные возможности алгоритма *DES*, поэтому использование стандарта ГОСТ предпочтительнее. Алгоритм шифрования построен с использованием сети Фейстеля.

2.3.4.1. Общая структурная схема

Структурная схема алгоритма криптографического преобразования (криптосхема) содержит элементы, согласно схеме рис. 2.5. Криптосхема содержит следующие элементы:

- ключевое запоминающее устройство (КЗУ) на 256 бит, состоящее из 8-ми 32-разрядных накопителей (X0, X1, X2, X3, X4, X5, X6, X7);
- четыре 32-разрядные накопителя (N1, N2, N3, N4);
- два 32-разрядных накопителя (N5, N6) с записанными в них постоянными заполнениями C2, C1;
- два 32-разрядных сумматора по модулю 232 (СМ1, СМ3);
- 32-разрядный сумматор поразрядного суммирования по модулю 2 (СМ2);

- сумматор по модулю 2 без ограничений на разрядность (СМ5);
- блок подстановки (К);
- регистр циклического сдвига на одиннадцать шагов в сторону старшего разряда (R).

Блок подстановки К состоит из восьми узлов замены К1-К8 с памятью по 64 бита каждый. Ключи, определяющие заполнение КЗУ и таблиц блока подстановки К являются секретными элементами и поставляются в установленном порядке.

Заполнение таблиц блока подстановки К является долговременным ключевым элементом, общим для сети ЭВМ. Организация различных видов связи достигается построением соответствующей ключевой системы. При этом может быть использована возможность выработки ключей (заполнений КЗУ) в режиме простой замены и зашифрования их в режиме простой замены с обеспечением имитозащиты для передачи по каналам связи или хранения в памяти ЭВМ.

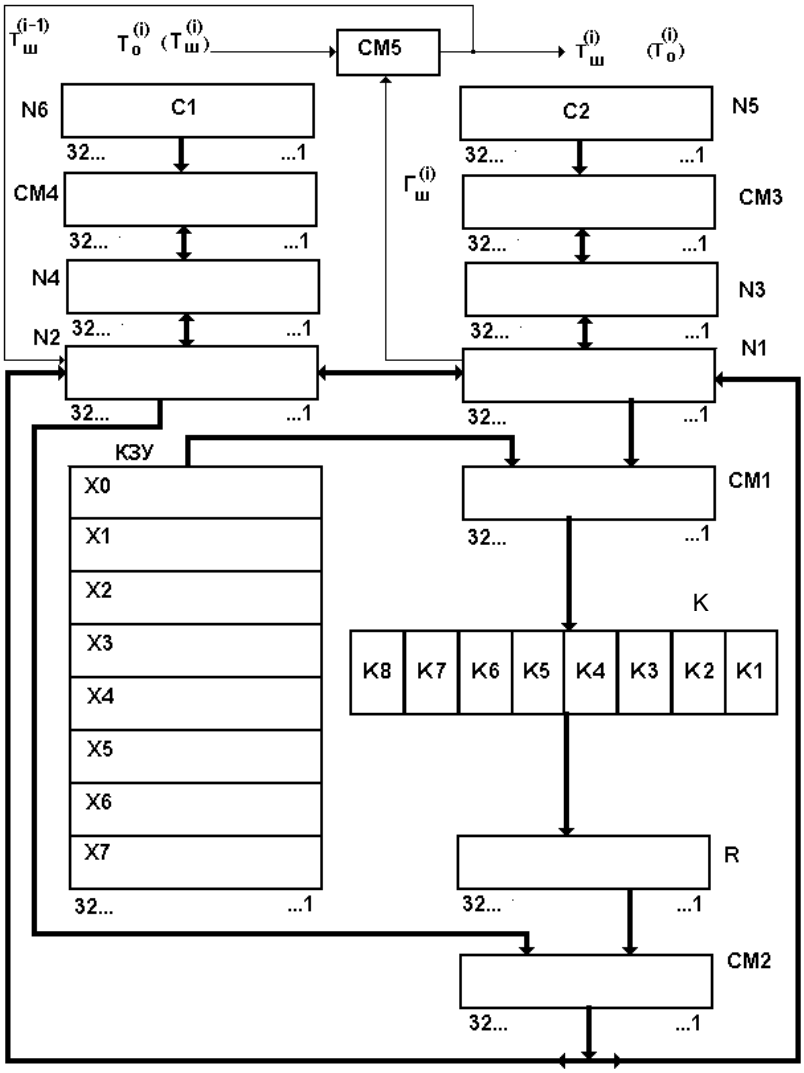


Рис. 2.5. Общая структурная схема алгоритма ГОСТ 28147–89

В криптосхеме предусмотрены четыре вида работы:

1) зашифрование (расшифрование) данных в режиме простой замены;

2) зашифрование (расшифрование) данных в режиме гаммирования;

3) зашифрование (расшифрование) данных в режиме гаммирования с обратной связью;

4) режим выработки имитовставки.

2.3.4.2. Режим простой замены

Зашифрование данных. Криптосхема, реализующая алгоритм зашифрования в режиме простой замены, должна иметь вид, показана на рис. 2.6.

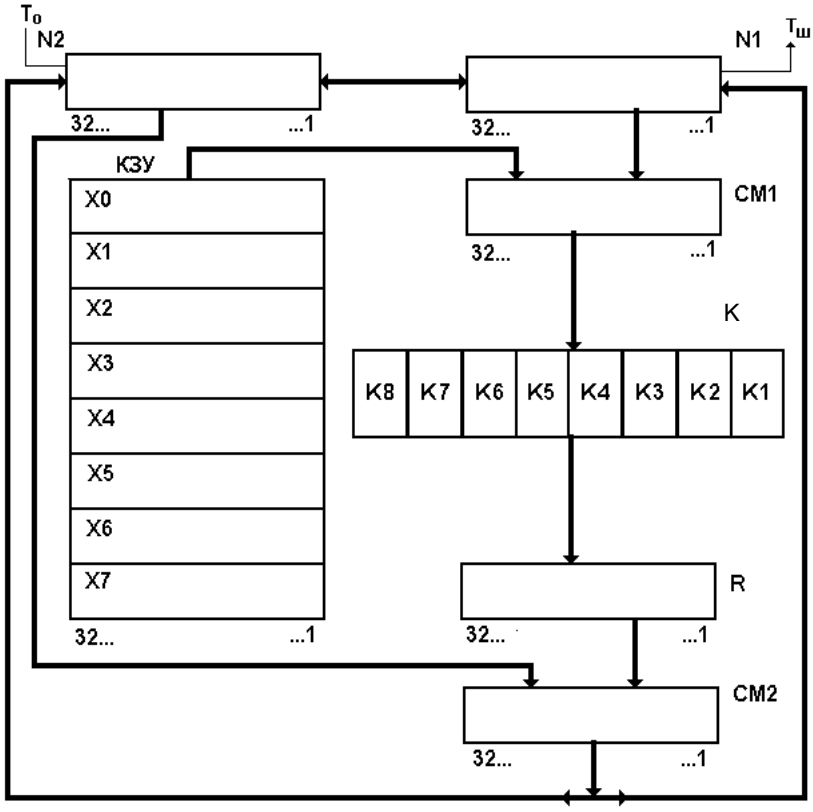


Рис. 2.6. Структурная схема зашифрования в режиме простой замены

Открытые данные, подлежащие зашифрованию, разбивают на блоки по 64 бита каждый. Ввод любого блока T_o = (a1(0), a2(0), ..., a31(0), a32(0), b1(0), b2(0), …, b32(0) двоичной информации в накопители N1 и N2 производятся так, что значение a1(0) вводится в 1-й разряд N1, значение a2(0) вводится во 2-й разряд накопителя и т.д., значение a32(0) вводится в 32-й разряд N1: значение b1(0) вводится в 1-й разряд N2, значение b2(0) вводится во 2-й разряд накопителя и т.д., значение b32(0) вводится в 32-й разряд N2. В результате получаем состояние (a32(0), a31(0), ..., a2(0), a1(0)) накопителя N1 и состояние (b32(0), b31(0),..., b2(0), b1(0)) накопителя N2. В КЗУ вводится 256 бит ключа. Содержимое восьми 32-разрядных накопителей X0, X1, ..., X7 имеет вид:

X0 = (W32, W31, ..., W2, W1)
X1 = (W64, W63, ..., W34, W33)
. . .
X7 = (W256, W255, ..., W226, W225)

Алгоритм зашифрования 64-разрядного блока открытых данных в режиме простой замены состоит из 32-х циклов.

В первом цикле начальное заполнение накопителя N1 суммируется по модулю 232 в сумматоре СМ1 с заполнением накопителя X0, при этом заполнение накопителя N1 сохраняется. Результат суммирования преобразуется в блоке подстановки K и полученный вектор поступает на вход регистра R, где циклически сдвигается на одиннадцать шагов в сторону старших разрядов. Результат сдвига суммируется поразрядно по модулю 2 в сумматоре СМ2 с 32-разрядным заполнением накопителя N2. Полученный в СМ2 результат записывается в N1, при этом старое заполнение N1 переписывается в N2. Первый цикл заканчивается.

Последующие циклы осуществляются аналогично, при этом во 2-м цикле из КЗУ считывается заполнение X1, в 3-ем цикле из КЗУ считывается заполнение X2 и т.д., в 8-м цикле из КЗУ считывается заполнение X7. В циклах с 9-го по 16-й, а также в циклах с 17-го по 24-й заполнения из КЗУ считываются в том же порядке. В последних восьми циклах с 25-й по 32-й порядок считывания заполнений КЗУ обратный. В 32-м цикле результат из сумматора СМ2 вводится в накопитель N2, а в накопителе N1 сохраняется

старое заполнение.

Полученные после 32-го цикла зашифрования заполнения накопителей N1 и N2 являются блоком зашифрованных данных, соответствующим блоку открытых данных.

64-разрядный блок зашифрованных данных $T_ш$ выводится из накопителей N1 и N2 в следующем порядке: из 1-го, 2-го, ..., 32-го разряда накопителя N1, затем из 1-го, 2-го,..., 32-го разрядов накопителя N2. Остальные блоки открытых данных в режиме простой замены зашифровываются аналогично.

Расшифрование данных. Криптосхема, реализующая алгоритм расшифрования в режиме простой замены имеет тот же вид, что и при зашифровании. В КЗУ вводятся 256 бит того же ключа, на котором осуществлялось зашифрование открытых данных.

Зашифрованные данные разбиваются на блоки по 64 бита каждый. Ввод любого блока

$$T_ш = (a1(32), a2(32), ..., a31(32), a32(32), b1(32), b2(32), ..., b32(32)$$

двоичной информации в накопители N1 и N2 производятся так, что значение a1(32) вводится в 1-й разряд N1, значение a2(32) вводится во 2-й разряд накопителя и т.д., значение a32(32) вводится в 32-й разряд N1: значение b1(32) вводится в 1-й разряд N2, значение b2(32) вводится во 2-й разряд накопителя и т.д., значение b32(32) вводится в 32-й разряд N2. Расшифрование осуществляется по тому же алгоритму, что и зашифрование открытых данных, с тем изменением, что заполнения накопителей X0, X1, ..., X7 считываются из КЗУ в циклах расшифрования в следующем порядке:

X0, X1, X2, X3, X4, X5, X6, X7, X7, X6, X5, X4, X3, X2, X1, X0
X7, X6, X5, X4, X3, X2, X1, X0, X7, X6, X5, X4, X3, X2, X1, X0

Полученные после 32-х циклов работы заполнения накопителей N1 и N2 составляют блок открытых данных. Аналогично расшифровываются остальные блоки зашифрованных данных. Алгоритм зашифрования в режиме простой замены 64-битового блока To обозначается через A, т.е.:

$$A (T_o) = A (a(0), b(0))) = (a(32), b(32)))= T_ш.$$

2.3.4.3. Режим гаммирования

Зашифрование данных. Криптосхема, реализующая алгоритм зашифрования данных в режиме гаммирования показана на схеме рис. 2.7. Открытые данные, разбитые на 64-разрядные блоки $T_o(1)$, $T_o(2)$, ..., $T_o(M – 1)$, $T_o(M)$, зашифровываются в режиме гаммирования путем поразрядного суммирования по модулю 2 в сумматоре СМ5 с гаммой шифра $Г_ш$, которая вырабатывается блоками по 64 бита:

$$Г_ш = (Г_ш(1), Г_ш(2), ..., Г_ш(M – 1), Г_ш(M)),$$

где M – определяется объемом шифруемых данных. В КЗУ вводятся 256 бит ключа. В накопителе N1, N2 вводится 64-разрядная двоичная последовательность (синхропосылка) S = (S1, S2, ..., S64), являющаяся исходным заполнением этих накопителей для последующей выработки M блоков гаммы шифра.

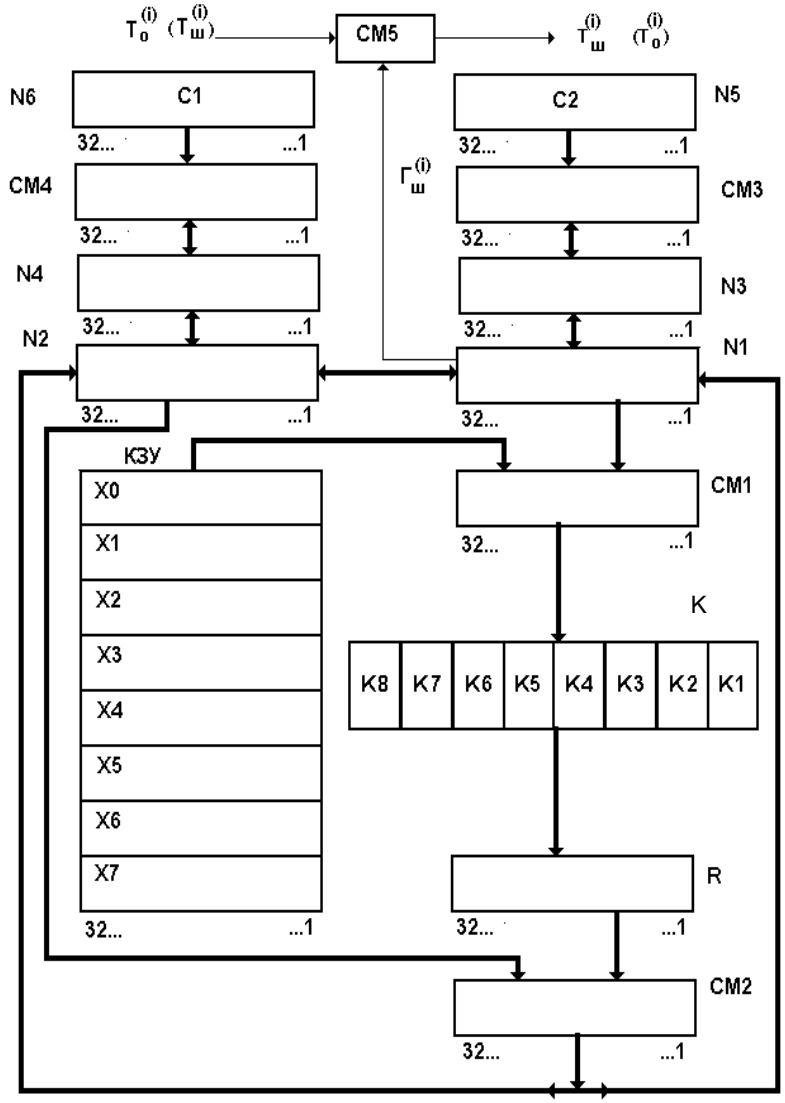


Рис. 2.7. Структурная схема зашифрования в режиме гаммирования

Исходное заполнение накопителей N1 и N2 (синхропосылка S) зашифровывается в режиме простой замены. Результат зашифрования A(S) = (Y0, Z0) переписывается в 32-разрядные накопители N3 и N4. Заполнение накопителя N4 суммируется по модулю (232-1) в сумматоре СМ4 с 32-разрядной константой С1 из накопителя N6, результат записывается в N4. Заполнение накопителя N3 суммируется по модулю 232 в сумматоре СМ3 с 32-разрядной константой С2 из накопителя N5, результат записывается в N3. Заполнение N3 переписывается в N1, а заполнение N4 переписывается в N2, при этом заполнение N3, N4 сохраняется. Заполнение N1 и N2 зашифровывается в режиме простой замены. Полученное в результате зашифрования заполнение N1, N2 образует первый 64-разрядный блок гаммы шифра $Г_ш(1)$, который суммируется поразрядно по модулю 2 в сумматоре СМ5 с первым 64-разрядным блоком открытых данных. В результате суммирования получается 64-разрядный блок зашифрованных данных. Аналогичным образом зашифровываются остальные блоки открытых данных. В канал связи или память ЭВМ передаются синхропосылка S и блоки зашифрованных данных.

Расшифрование данных. При расшифровании криптосхема имеет тот же вид, что и при зашифровании открытых данных в режиме гаммирования. В КЗУ вводятся 256 бит ключа, с помощью которого осуществлялось зашифрование данных. В накопители N1 и N2 вводится синхропосылка S. Процесс выработки M блоков гаммы шифра осуществляется совершенно аналогично описанному выше. Блоки зашифрованных данных суммируются поразрядно по модулю 2 в сумматоре СМ5 с блоками гаммы шифра, в результате получаются блоки открытых данных.

2.3.4.4. Режим гаммирования с обратной связью

Зашифрование данных. Криптосхема, реализующая алгоритм зашифрования в режиме гаммирования с обратной связью показана на рис. 2.8. Открытые данные, разбитые на 64-разрядные блоки зашифровываются в режиме гаммирования с обратной связью путем поразрядного суммирования по модулю 2 в сумматоре СМ5 с гаммой шифра, вырабатываемой блоками по 64 бита каждый.

В КЗУ вводится 256 бит ключа. Синхропосылка S из 64 бит вводится в N1 и N2. Исходное заполнение N1 и N2 зашифровывается в режиме простой замены. Полученное в результате заполнение N1 и N2 образует первый 64-разрядный блок гаммы шифра, который суммируется поразрядно по модулю 2 в сумматоре СМ5 с первым 64-разрядным блоком открытых данных. В результате получается 64-разрядный блок зашифрованных данных.

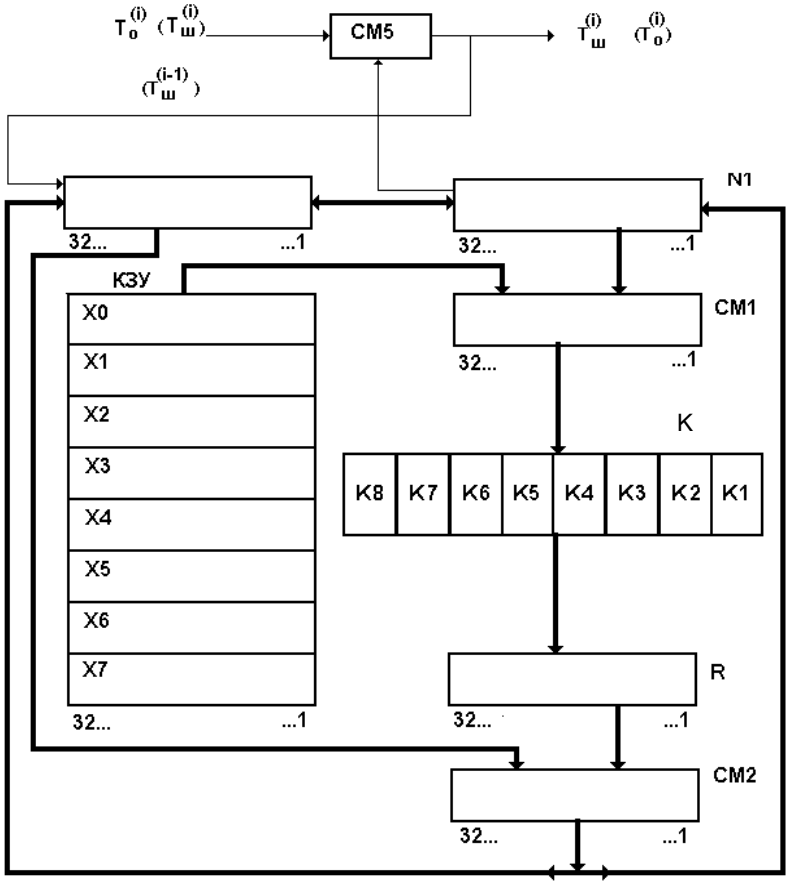


Рис. 2.8. Структурная схема зашифрования в режиме гаммирования с обратной связью

Блок зашифрованных данных одновременно является также исходным состоянием N1, N2 для выработки второго блока гаммы шифра и по обратной связи записывается в указанные накопители. Заполнение N1, N2 зашифровывается в режиме простой замены. Полученное в результате зашифрования заполнение N1, N2 образует второй 64-разрядный блок гаммы шифра, который суммируется поразрядно по модулю 2 в сумматоре СМ5 со вторым блоком открытых данных. Выработка последующих блоков гаммы шифра и зашифрование соответствующих блоков открытых данных производится аналогично. В канал связи или память ЭВМ передаются синхропосылка S и блоки зашифрованных данных.

Расшифрование данных. При расшифровании криптосхема имеет тот же вид, что и при зашифровании.

В КЗУ вводятся 256 бит того же ключа, с помощью которого производилось зашифрование данных. Синхропосылка S вводится в N1, N2. Исходное заполнение N1, N2 (синхропосылка) зашифровывается в режиме простой замены. Полученное в результате заполнение N1, N2 образует первый блок гаммы шифра, который суммируется с по модулю 2 в сумматоре СМ5 с блоком зашифрованных данных. В результате получается первый блок открытых данных. Блок зашифрованных данных является исходным

заполнением N1, N2 для выработки второго блока гаммы шифра. Полученное заполнение N1, N2 зашифровывается в режиме простой замены. Полученный в результате блок суммируется поразрядно по модулю 2 в сумматоре СМ5 со вторым блоком зашифрованных данных. В результате получается второй блок открытых данных. Аналогичным образом производится расшифрование последующих блоков зашифрованных данных.

2.3.4.5. Режим выработки имитовставки

Для обеспечения имитозащиты открытых данных, состоящих из М 64-разрядных блоков $T_o(1)$, $T_o(2)$, ..., $T_o(M)$ где $M > 2$, вырабатывается дополнительный блок из 1 бит (имитовставка И1).

Процесс выработки имитовставки единообразен для всех режимов шифрования. Первый блок открытых данных записывается в накопители N1, N2. Заполнение N1, N2 подвергается преобразованию, соответствующему первым 16 циклам алгоритма зашифрования в режиме простой замены. В КЗУ при этом находится тот же ключ, которым зашифровываются блоки открытых данных. Полученное после 16 циклов преобразование заполнение N1, N2 суммируется в сумматоре СМ5 по модулю 2 со вторым блоком открытых данных.

Результат суммирования заносится в N1, N2 и подвергается преобразованию, соответствующему первым 16 циклам алгоритма зашифрования в режиме простой замены. Полученное заполнение N1, N2 суммируется в СМ5 по модулю 2 с третьим блоком открытых данных и т.д. Итоговый результат суммирования заносится в N1, N2 и зашифровывается в режиме простой замены по первым 16 циклам работы алгоритма. Из полученного значения накопителей N1 и N2 выбирается отрезок И1 (имитовставка) длиной 1 бит. Имитовставка передается по каналу связи или в память ЭВМ в конце зашифрованных данных.

Поступившие зашифрованные данные расшифровываются, из полученных блоков открытых данных аналогично описанному выше вырабатывается имитовставка И1', которая затем сравнивается с имитовставкой И1 полученной вместе с зашифрованными данными. В случае несовпадения имитовставок полученные данные считаются ложными. Выработка имитовставки может производится или перед зашифрованием (после расшифрования) всего сообщения, или параллельно с зашифрованием (расшифрованием) по блокам. Первые блоки открытых данных, которые участвуют в выработке имитовставки, могут содержать служебную информацию (адресную часть, отметку времени, синхропосылку и т.д.) и не зашифровываются. Значение параметра l (длина двоичных разрядов в имитовставке) определяется требованием к криптографической защите, при этом учитывается, что вероятность навязывания ложных данных равна $2 - l$.

2.4. Режимы применения блочных шифров

Для шифрования исходного текста произвольной длины блочные шифры могут быть использованы в нескольких режимах. Далее будут рассмотрены четыре режима применения блочных шифров, наиболее часто встречающиеся в системах криптографической защиты информации, а именно режимы *электронной кодировочной книги* (*ECB – Electronic Code Book*), *сцепления блоков шифрованного текста* (*CBC – Cipher Block Chaining*), *обратной связи по шифрованному тексту* (*CFB – Cipher Feedback*) и *обратной связи по выходу* (*OFB – Output Feedback*).

В режиме *электронной кодировочной книги* каждый блок исходного текста шифруется блочным шифром независимо от других (рис. 2.9).

Стойкость режима *ECB* равна стойкости самого шифра. Однако, структура исходного текста при этом не скрывается. Каждый одинаковый блок исходного текста приводит к появлению одинакового блока шифрованного текста Исходным текстом можно легко манипулировать путем удаления, повторения или перестановки блоков. Скорость шифрования равна скорости блочного шифра.

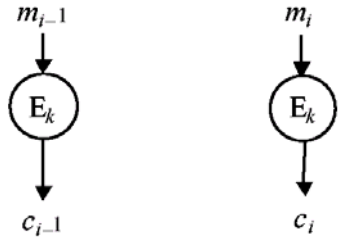


Рис. 2.9. Режим электронной кодировочной книги

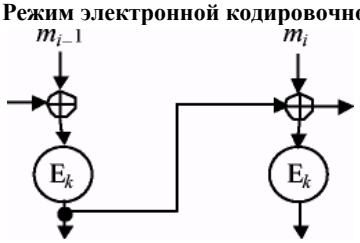


Рис. 2.10. Режим сцепления блоков шифрованного текста

Режим *ECB* допускает простое распараллеливание для увеличения скорости шифрования, но никакая обработка невозможна до поступления блока (за исключением генерации ключей). Режим *ECB* соответствует режиму простой замены ГОСТ.

В режиме *сцепления блоков шифрованного текста* (*CBC*) каждый блок исходного текста складывается поразрядно по модулю 2 с предыдущим блоком шифрованного текста, а затем шифруется (рис. 2.10). Для начала процесса шифрования используется *синхропосылка* (или начальный вектор), которая передается в канал связи в открытом виде.

Стойкость режима *CBC* равна стойкости блочного шифра, лежащего в его основе. Кроме того, структура исходного текста скрывается за счет сложения предыдущего блока шифрованного текста с очередным блоком открытого текста. Стойкость шифрованного текста увеличивается, поскольку становится невозможной прямая манипуляция исходным текстом, кроме как путем удаления блоков из начала или конца шифрованного текста.

Скорость шифрования равна скорости работы блочного шифра, но простого способа распараллеливания процесса шифрования не существует, хотя расшифрование может проводиться параллельно.

Одной из потенциальных проблем режима *CBC* является возможность внесения контролируемых изменений в последующий расшифрованный блок исходного текста. Например, если злоумышленник изменит один бит в блоке, то весь блок будет расшифрован неверно, но в следующем блоке появится ошибка в соответствующей позиции. Есть ситуации, когда такое нежелательно. Для борьбы с этой угрозой исходный текст должен содержать определенную избыточность.

Известны модификации режима *CBC*.

Режим *сцепления блоков шифрованного текста с распространением* (*PCBC – Propagating CBC*) отличается тем, что по модулю 2 складывается как предыдущий блок шифрованного, так и исходного текста:

$$c_i = E_k(m_i \oplus c_{i-1} \oplus m_{i-1}),$$
$$m_i = c_{i-1} \oplus m_{i-1} \oplus D_k(c$$

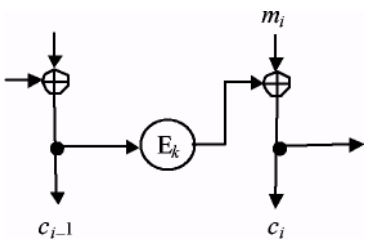


Рис. 2.11. Режим обратной связи по шифрованному тексту

Режим *сцепления блоков шифрованного текста с контрольной суммой*

(*CBCC – CBC with Checksum*) отличается тем, что к последнему блоку исходного текста перед шифрованием прибавляется сумма по модулю два всех предыдущих блоков исходного текста. Это дает возможность проконтролировать целостность передаваемого текста с небольшими дополнительными накладными расходами.

В режиме *обратной связи по шифрованному тексту* (*CFB*) предыдущий блок шифрованного текста шифруется еще раз, и для получения очередного блока шифрованного текста результат складывается поразрядно по модулю 2 с блоком исходного текста. Для начала процесса шифрования также используется начальный вектор (рис. 2.11).

Стойкость режима *CFB* равна стойкости блочного шифра, лежащего в его основе и структура исходного текста скрывается за счет использования операции сложения по модулю 2. Манипулирование исходным текстом путем удаления блоков из начала или конца шифрованного текста становится невозможным. В режиме *CFB*, если два блока шифрованного текста идентичны, то результаты их шифрования на следующем шаге также будут идентичны, что создает возможность утечки информации об исходном тексте.

Скорость шифрования равна скорости работы блочного шифра и простого способа распараллеливания процесса шифрования также не существует. Этот режим в точности соответствует режиму гаммирования с обратной связью алгоритма ГОСТ 28147–89.

Режим *обратной связи по выходу* (*OFB*) подобен режиму *CFB* за исключением того, что величины, складываемые по модулю 2 с блоками исходного текста, генерируются независимо от исходного или шифрованного текста (рис. 2.12). Для начала процесса шифрования также используется начальный вектор. Режим *OFB* обладает преимуществом перед режимом *CFB* – любые битовые ошибки, возникшие в процессе передачи, не влияют на расшифрование последующих блоков. Однако, возможна простая манипуляция исходным текстом путем изменения шифрованного текста. Существует модификация этого режима под названием режим *обратной связи по выходу с нелинейной функцией*.

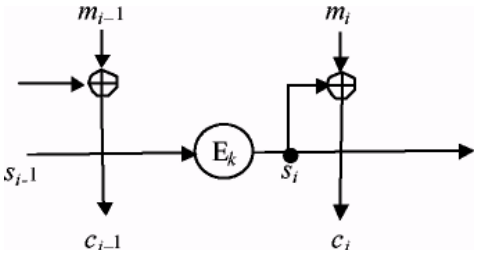


Рис. 2.12. Режим обратной связи по выходу

2.5. Потоковые шифры

2.5.1. ОБЩИЕ СВЕДЕНИЯ О ПОТОКОВЫХ ШИФРАХ

Потоковые шифры представляют собой разновидность гаммирования и преобразуют открытый текст в шифрованный последовательно по 1 биту. *Генератор ключевой последовательности,* иногда называемый *генератором бегущего ключа,* выдает последовательность бит k_1, k_2, ..., k_i, ... Эта ключевая последовательность складывается по модулю 2 с последовательностью бит исходного текста p_1, p_2, ..., p_i, ... для получения шифрованного текста:

$$c_i = p_i \oplus k_i .$$

На приемной стороне шифрованный текст складывается по модулю 2 с идентичной ключевой последовательностью для получения исходного текста:

$$c_i \oplus k_i = p_i \oplus k_i \oplus k_i = p_i .$$

Стойкость системы целиком зависит от внутренней структуры генератора ключевой последовательности. Если генератор выдает последовательность с небольшим периодом, то стойкость системы будет невелика. На-

против, если генератор будет выдавать бесконечную последовательность истинно случайных бит, то получится одноразовый блокнот с идеальной стойкостью.

Реальная стойкость потоковых шифров лежит где-то посредине между стойкостью простой моноалфавитной подстановки и одноразового блокнота. Генератор ключевой последовательности выдает поток битов, который выглядит случайным, но в действительности является детерминированным и может быть в точности воспроизведен на приемной стороне. Чем больше генерируемый поток похож на случайный, тем больше усилий потребуется от криптоаналитика для взлома шифра.

Если каждый раз при включении генератор будет выдавать одну и ту же последовательность, то взлом криптосистемы будет тривиальной задачей. Перехватив два шифрованных текста, злоумышленник может сложить их по модулю 2 и получить два исходных текста, сложенных также по модулю 2. Такую систему раскрыть очень просто. Если же в руках противника окажется пара исходный текст – шифрованный текст, задача вообще становится тривиальной.

По этой причине все потоковые шифры предусматривают использование ключа. Выход генератора ключевой последовательности зависит от этого ключа. В этом случае простой криптоанализ будет невозможен.

Потоковые шифры наиболее пригодны для шифрования непрерывных потоков данных, например, в сетях передачи данных.

Структуру генератора ключевой последовательности можно представить в виде конечного автомата с памятью, состоящего из трех блоков: блока памяти, хранящего информацию о состоянии генератора, выходной функции, генерирующей бит ключевой последовательности в зависимости от состояния, и функции переходов, задающей новое состояние, в которое перейдет генератор на следующем шаге.

2.5.2. МЕТОДЫ ЗАМЕНЫ

Шифрование методом замены (подстановки) основано на алгебраической операции, называемой подстановкой. *Подстановкой* называется взаимно-однозначное отображение некоторого конечного множества M на себя. Число N элементов этого множества называется степенью подстановки. Природа множества M роли не играет, поэтому можно считать, что $M = \{1, 2, ..., N\}$.

Если при данной подстановке S число j переходит в I_j, то подстановка обозначается символом S:

$$S = \begin{bmatrix} 1 & 2 & & n \\ I_1 & I_2 & & I_n \end{bmatrix}$$

В этой записи числа 1, 2, ..., n можно произвольным образом переставлять, соответственно переставляя числа I_1, I_2, ..., I_n. Результат последовательного выполнения двух подстановок S_1 и S_2 одной и той же степени также является подстановкой, которая называется произведением подстановок S_1 и S_2 и обозначается S_1S_2.

Пусть S – произвольная подстановка степени n. Если для некоторого j число I_j отлично от j, то говорят, что подстановка S действительно перемещает число j; в противном случае – подстановка S оставляет число j на месте.

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

Подстановка S называется *транспозицией*, если существует пара (j_1, j_2) различных элементов из M, удовлетворяющих условиям:

$$Ij_1 = j_2,\ Ij_2 = j_2,\ Ij = j \text{ для каждого } j \in \{M \backslash \{j_1, j_2\}\}.$$

Любая подстановка разлагается в произведение транспозиций.

В криптографии рассматриваются четыре типа подстановки (замены): моноалфавитная, гомофоническая, полиалфавитная и полиграммная. Далее всюду в примерах, где необходимо, будем использовать кодирование букв русского алфавита, приведенное в табл. 2.2. Знак "_" в табл. 2.2 и далее означает пробел.

2.2. Коды букв русского алфавита

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

При моноалфавитной замене каждой букве алфавита открытого текста ставится в соответствие одна буква шифртекста из этого же алфавита, например шрифт "отбаш".

Пример. Открытый текст: "ШИФРОВАНИЕ_ЗАМЕНОЙ". Подстановка задана в табл. 2.3.

2.3. Подстановка шифра

ИТ	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р
ШТ	_	Я	Ю	Э	Ь	Ы	Ъ	Щ	Ш	Ч	Ц	Х	Ф	У	Т	С	Р
ИТ	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я	_	
ШТ	П	О	Н	М	Л	К	Й	И	З	Ж	Е	Д	Г	В	Б	А	

ИТ – алфавит исходного текста ШТ – алфавит шифртекста. Шифртекст: "ИШМРТЮ_УШЫАЩ_ФЫУТЧ".

Основным недостатком рассмотренного метода является сохранение статистических свойств открытого текста (частота повторения букв) в шифртексте.

Общая формула моноалфавитной замены выглядит в виде:

$$Y_i = k_1 X_i + k_2 (\mathrm{mod}\ N),$$

где Y_i – i-й символ алфавита; k_1 и k_2 – константы; X_i – i-й символ открытого текста (номер буквы в алфавите); N – длина используемого алфавита.

Шифр, задаваемый формулой:

$$Y_i = X_i + k_i (\mathrm{mod}\ N),$$

где k_i – i-я буква ключа, в качестве которого используются слово или фраза, называется *шифром Вижинера*.

Пример. Открытый текст: "ЗАМЕНА". Ключ: "КЛЮЧ" (табл. 2.4).

2.4. Шифр Вижинера

Открытый текст	Ключ	Преобразование	Шифр
З	К	y1 = 8 + 11(mod 33) = 19	Т
А	Л	y2 = 1 + 12(mod 33) = 13	М
М	Ю	y3 = 13 + 31(mod 33) = 11	К
Е	Ч	y4 = 6 + 24(mod 33) = 30	Э
Н	К	y5 = 14 + 11(mod 33) = 25	Ш
А	Л	y6 = 1 + 12(mod 33) = 13	М

Шифртекст: "ТМКЭШМ".

Шифр Бофорта – многоалфавитная криптосистема, аналогичная криптосистеме Вижинера. Строками квадрата Бофорта являются строки квадрата Вижинера, записанные в обратном порядке. Криптосистема названа в честь адмирала Френсиса Бофорта. Шифр Бофорта использует

формулы

$$y_i = k_i - x_i (\mathrm{mod}\ n) \quad \text{и} \quad y_i = x_i - k_i (\mathrm{mod}\ n).$$

Гомофоническая замена одному символу открытого текста ставит в соответствие несколько символов шифртекста. Этот метод применяется для искажения статистических свойств шифртекста.

Пример. Открытый текст: "ЗАМЕНА". Подстановка задана табл. 2.5.

2.5. Гомофоническая подстановка

Алфавит открытого текста	А	Б	…	Е	Ж	З	…	М	Н	…
Алфавит шифртекста	17	23		97	47	76		32	55	
	31	44		51	67	19		28	84	
	48	63		15	33	59		61	34	

Шифртекст: "76 17 32 97 55 31".

Таким образом, при гомофонической замене каждая буква открытого текста заменяется по очереди цифрами соответствующего столбца.

Полиалфавитная подстановка использует несколько алфавитов шифртекста. Пусть используется k алфавитов. Тогда открытый текст

$$X = X_1X_2...X_kX_{k+1}...X_{2k}X_{2k+1}...$$

заменяется шифртекстом

$$Y = F_1(X_1)F_2(X_2)...F_k(X_k)F_{k+1}(X_{k+1})...F_{2k}(X_{2k})F_{2k+1}(X_{2k+1}),$$

где $F_i(X_j)$ – символ шифртекста алфавита i для символа открытого текста X_j.

Полиграммная замена формируется из одного алфавита с помощью специальных правил. В качестве примера рассмотрим *шифр Плэйфера*.

В этом шифре алфавит располагается в матрице. Открытый текст разбивается на пары символов X_iX_{i+1}. Каждая пара символов открытого текста заменяется на пару символов из матрицы следующим образом:

- если символы находятся в одной строке, то каждый из символов пары заменяется на стоящий правее его (за последним символом в строке следует первый);
- если символы находятся в одном столбце, то каждый символ пары заменяется на символ, расположенный ниже его в столбце (за последним нижним символом следует верхний);
- если символы пары находятся в разных строках и столбцах, то они считаются противоположными углами прямоугольника. Символ, находящийся в левом углу, заменяется на символ, стоящий в другом левом углу; замена символа, находящегося в правом углу, осуществляется аналогично;
- если в открытом тексте встречаются два одинаковых символа подряд, то перед шифрованием между ними вставляется специальный символ (например, тире).

Пример. Открытый текст: "ШИФР_ПЛЭЙФЕРА.". Матрица алфавита представлена в табл. 2.6.

2.6. Матрица алфавита

А	Х	Б	М	Ц	В
Ч	Г	Н	Ш	Д	О
Е	Щ	,	Ж	У	П
.	З	Ъ	Р	И	Й
С	Ь	К	Э	Т	Л
Ю	Я	_	Ы	Ф	–

Шифртекст: "РДИЫ,-СТ-И.ЖЧС".

При рассмотрении этих видов шифров становится очевидным, что чем больше длина ключа, тем лучше шифр. Существенного улучшения свойств шифртекста можно достигнуть при использовании шифров с автоключом.

Шифр, в котором сам открытый текст или получающаяся криптограмма используются в качестве "ключа", называется шифром с *автоключом*. Шифрование в этом случае начинается с ключа, называемого первичным, и продолжается с помощью открытого текста или криптограммы, смещенной на длину первичного ключа.

Пример. Открытый текст: "ШИФРОВАНИЕ_ЗАМЕНОЙ". Первичный

ключ: "КЛЮЧ". Шифрование с автоключом при использовании открытого текста представлено в табл. 2.7.

2.7. Шифрование с автоключом при использовании открытого текста

ИТ	Ш	И	Ф	Р	О	В	А	Н	И	Е	_	З	А	М	Е	Н	О	Й
Кл	К	Л	Ю	Ч	Ш	И	Ф	Р	О	В	А	Н	И	Е	_	З	А	М
Код	03	21	19	08	07	12	22	31	24	09	01	22	10	19	06	22	16	23
ШТ	В	Ф	Т	З	Ж	Л	Х	Ю	Ч	И	А	Х	Й	Т	Е	Х	П	Ц

ИТ – алфавит исходного текста; Кл – ключ; ШТ – алфавит шифртекста.

Шифрование с автоключом при использовании криптограммы представлено в табл. 2.8.

2.8. Шифрование с автоключом при использовании криптограммы

ИТ	Ш	И	Ф	Р	О	В	А	Н	И	Е	_	З	А	М	Е	Н	О	Й
Кл	К	Л	Ю	Ч	В	Ф	Т	З	С	Ч	У	Х	Ъ	Э	У	Э	Ы	Й
Код	03	21	19	08	18	24	20	22	27	30	20	30	28	10	26	11	10	20
ШТ	В	Ф	Т	З	С	Ч	У	Х	Ъ	Э	У	Э	Ы	Й	Щ	К	Й	У

ИТ – алфавит исходного текста; Кл – ключ; ШТ – алфавит шифртекста.

2.5.3. МЕТОДЫ ПЕРЕСТАНОВКИ

При использовании для шифрования информации методов перестановки символы открытого текста переставляются в соответствии с некоторыми правилами.

Пример. Открытый текст: "ШИФРОВАНИЕ_ПЕРЕСТАНОВКОЙ". Ключ (правило перестановки): группы из 8 букв с порядковыми номерами 1, 2, ..., 8 переставить в порядок 3-8-1-5-2-7-6-4.

Шифртекст: "ФНШОИАВР_СИЕЕЕРПННТВАОКО".

Можно использовать и усложненную перестановку. Для этого открытый текст записывается в матрицу по определенному ключу k_1. Шифртекст образуется при считывании из этой матрицы по ключу k_2.

Пример. Открытый текст: "ШИФРОВАНИЕ_ПЕРЕСТАНОВКОЙ". Матрица из четырех столбцов. Ключи: k_1 = {5-3-1-2-4-6}; k_2 = {4-2-3-1}. Запись по строкам производится в соответствии с ключом k_1. Чтение по столбцам в соответствии с ключом k_2 (табл. 2.9.).

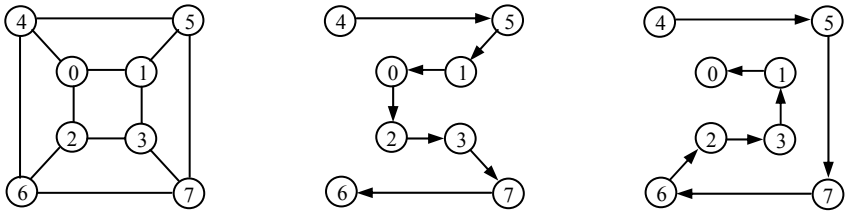
2.9. Шифрование перестановкой

1	И	Е	_	П
2	Е	Р	Е	С
3	О	В	А	Н
4	Т	А	Н	О
5	Ш	И	Ф	Р
6	В	К	О	Й
k_1/k_2	1	2	3	4

Шифртекст: "ПСНОРЙЕРВАИК_ЕАНФОИЕОТШВ".

Наиболее сложные перестановки осуществляются по гамильтоновым путям, которых в графе может быть несколько.

Пример. Открытый текст: "ШИФРОВАНИЕ_ПЕРЕСТАНОВКОЙ". Ключ: гамильтонов путь на графе (рис. 2.13).



Общий вид Маршрут 1 Маршрут 2

Рис. 2.13. Гамильтоновы пути на графе

Шифртекст: "ШАОНИРФВИЕЕСЕП_РТОВИАОНК".

Необходимо отметить, что для данного графа из восьми вершин можно предложить несколько маршрутов записи открытого текста и несколько гамильтоновых путей для чтения криптограмм.

В 1991 г. В.М. Кузьмич предложил схему перестановки, основанной на кубике Рубика. Согласно этой схеме открытый текст записывается в ячейки граней куба по строкам. После осуществления заданного числа заданных поворотов слоев куба считывание шифртекста осуществляется по столбикам. Сложность расшифрования в этом случае определяется количеством ячеек на гранях куба и сложностью выполненных поворотов слоев. Перестановка, основанная на кубике Рубика, получила название объемной (многомерной) перестановки.

В 1992–1994 гг. идея применения объемной перестановки для шифрования открытого текста получила дальнейшее развитие. Усовершенствованная схема перестановок по принципу кубика Рубика, в которой наряду с открытым текстом перестановке подвергаются и функциональные элементы самого алгоритма шифрования, легла в основу секретной системы "Рубикон". В качестве прообразов пространственных многомерных структур, на основании объемных преобразований которых осуществляются перестановки, в системе "Рубикон" используются трехмерные куб и тетраэдр. Другой особенностью системы "Рубикон" является генерация уникальной версии алгоритма и ключа криптографических преобразований на основании некоторого секретного параметра (пароля). Это обеспечивает как дополнительные трудности для криптоанализа перехваченных сообщений нарушителем (неопределенность примененного алгоритма), так и возможность априорного задания требуемой криптостойкости. Криптостойкость данной системы определяется длиной ключа, криптостойкостью отдельных функциональных элементов алгоритма криптографических преобразований, а также количеством таких преобразований.

Использование уникальных алгоритма и ключа шифрования для каждого пользователя системы соответствует положению теории К. Шеннона о том, что абсолютно стойкий шифр может быть получен только при использовании "ленты однократного применения", т.е. уникальных параметров при каждом осуществлении шифрования.

2.5.4. МЕТОДЫ АНАЛИТИЧЕСКИХ ПРЕОБРАЗОВАНИЙ

Шифрование методами аналитических преобразований основано на понятии односторонней функции. Будем говорить, что функция $y = f(x)$ является односторонней, если она за сравнительно небольшое число операций преобразует элемент открытого текста X в элемент шифртекста Y для всех значений X из области определения, а обратная операция (вычисление $X = F^{-1}(Y)$ при известном шифртексте) является вычислительно трудоемкой.

В качестве односторонней функции можно использовать следующие преобразования: умножение матриц; решение задачи об укладке ранца; вычисление значения полинома по модулю; экспоненциальные преобразования и др.

Метод умножения матриц использует преобразование вида

$$Y = CX, \text{ где } Y = \|y_1, y_2, ..., y_n\|^T; C = \|C_{ij}\|; X = \|x_1, x_2, ..., x_n\|.$$

Пример. Открытый текст: "ПРИКАЗ" ("16 17 09 11 01 08" согласно табл. 2.2).

Матрица C: $\begin{vmatrix}1&3&2\\2&1&5\\3&2&1\end{vmatrix}\begin{vmatrix}1&3&2\\2&1&5\\3&2&1\end{vmatrix}*\begin{vmatrix}16\\17\\09\end{vmatrix}=|85\ 94\ 91|\qquad\begin{vmatrix}1&3&2\\2&1&5\\3&2&1\end{vmatrix}*\begin{vmatrix}11\\01\\08\end{vmatrix}=|30\ 63\ 43|$

Шифртекст: "85 94 91 30 63 43".

Задача об укладке ранца формулируется следующим образом.

Задан вектор $C = |c_1, c_2, ..., c_n|$, который используется для шифрования сообщения, каждый символ s_i которого представлен последовательностью из n бит $s_i = |x_1, x_2, ..., x_n|^T$, $X_k \subset \{0, 1\}$. Шифртекст получается как скалярное произведение Cs_i.

Пример. Открытый текст: "ПРИКАЗ" ("16 17 09 11 01 08" согласно табл. 2.2). Вектор $C = \{1,3,5,7,11\}$. Запишем символы открытого текста пятиразрядным двоичным кодом.

П	Р	И	К	А	З
10000	10001	01001	01011	00001	01000

Произведем соответствующие операции:

$y_1 = 1*1 = 1.$
$y_2 = 1*1 + 1*11 = 12.$
$y_3 = 1*3 + 1*11 = 14.$
$y_4 = 1*3 + 1*7 + 1*11 = 21.$
$y_5 = 1*11 = 11.$
$y_6 = 1*3 = 3.$

Шифртекст: "01 12 14 21 11 03".

Метод полиномов основан на преобразовании

$$y_i = x_i^n + a_1*x_i^{(n-1)} + ... + a_n*x_i (\mathrm{mod}\ p),$$

где n, a_1, a_2, ..., a_n – целые неотрицательные числа, не превосходящие p, $1 <= x_i, y_i <= p$; p – большое простое число.

Пример. Открытый текст: "ПРИКАЗ". ("16 17 09 11 01 08" согласно табл. 2.2)

Полином:

$y_i = x_i^3 + 2x_i^2 + 3xi + 4 (\mathrm{mod}\ 991).$
$y_1 = 16^3 + 2*16^2 + 3*16 + 4 (\mathrm{mod}\ 991) = 696.$
$y_2 = 17^3 + 2*17^2 + 3*16 + 4 (\mathrm{mod}\ 991) = 591.$
$y_3 = 9^3 + 2*9^2 + 3*9 + 4 (\mathrm{mod}\ 991) = 922.$
$y_4 = 11^3 + 2*11^2 + 3*11 + 4 (\mathrm{mod}\ 991) = 619.$
$y_5 = 1^3 + 2*1^2 + 3*1 + 4 (\mathrm{mod}\ 991) = 10.$
$y_6 = 8^3 + 2*8^2 + 3*8 + 4 (\mathrm{mod}\ 991) = 668.$

Шифртекст: "96 591 922 619 010 668".

Экспоненциальный шифр использует преобразование вида

$$y_i = a^{(xi)} (\mathrm{mod}\ p),$$

где x_i – целое, $1 <= x_i <= p - 1$; p – большое простое число; a – целое, $1 <= a <= p$.

Пример. Открытый текст: "ПРИКАЗ" ("16 17 09 11 01 08" согласно табл. 2.2); $a = 3$; $p = 991$.

$y_1 = 3^{16} (\mathrm{mod}\ 991) = 43046721 (\mathrm{mod}\ 991) = 654.$
$y_2 = 3^{17} (\mathrm{mod}\ 991) = 129140163 (\mathrm{mod}\ 991) = 971.$
$y_3 = 3^9 (\mathrm{mod}\ 991) = 19683 (\mathrm{mod}\ 991) = 854.$
$y_4 = 3^{11} (\mathrm{mod}\ 991) = 177147 (\mathrm{mod}\ 991) = 749.$
$y_5 = 3^1 (\mathrm{mod}\ 991) = 3.$
$y_6 = 3^8 (\mathrm{mod}\ 991) = 6561 (\mathrm{mod}\ 991) = 615.$

Шифртекст: "654 971 854 749 003 615".

2.5.5. ГАММИРОВАНИЕ (ШИФРОВАНИЕ С ПОМОЩЬЮ ДАТЧИКАПСЕВДОСЛУЧАЙНЫХ ЧИСЕЛ)

Различают два случая: метод конечной гаммы и метод бесконечной гаммы. В качестве конечной гаммы может использоваться фраза, а в качестве бесконечной – последовательность, вырабатываемая датчиком псевдослучайных чисел.

Принцип зашифрования заключается в генерации гаммы шифра с помощью датчика псевдослучайных чисел (ПСЧ) и наложении полученной гаммы на открытые данные обратимым образом (например, при использовании логической операции "исключающее ИЛИ").

Процесс расшифрования данных сводится к повторной генерации гаммы шифра при известном ключе и наложению такой гаммы на зашифрованные данные. Полученный зашифрованный текст является достаточно трудным для раскрытия в том случае, когда гамма шифра не содержит повторяющихся битовых последовательностей. По сути дела гамма шифра должна изменяться случайным образом для каждого шифруемого слова. Фактически если период гаммы превышает длину всего зашифрованного текста и неизвестна никакая часть исходного текста, то шифр можно раскрыть только прямым перебором (подбором ключа). В этом случае криптостойкость определяется размером ключа.

Чтобы получить линейные последовательности элементов гаммы, длина которых превышает размер шифруемых данных, используются датчики ПСЧ. На основе теории групп было разработано несколько типов таких датчиков. В настоящее время наиболее доступными и эффективными являются *конгруэнтные генераторы ПСЧ*. Они вырабатывают последовательности псевдослучайных чисел $T(i)$, описываемые соотношением

$$T(i + 1) = (A*T(i) + C) \bmod M,$$

где A и C – константы; $T(0)$ – исходная величина, выбранная в качестве порождающего числа.

Для шифрования данных с помощью датчика ПСЧ может быть выбран ключ любого размера. Например, пусть ключ состоит из набора чисел $X(j)$ размерностью b, где $j = 1, 2, ..., N$. Тогда создаваемую гамму шифра G можно представить как объединение непересекающихся множеств $H(j)$:

$$G = H(1) \cup H(2) \cup ... \cup H(N),$$

где $H(j)$ – множество соответствующих j-му сегменту данных и полученных на основе порождающего числа $Y(j)$, определенного как функция от $X(j)$ (например, ПСЧ, полученное на основе $X(j)$).

Разумеется, возможны и другие, более изощренные варианты выбора порождающих чисел для гаммы шифра. Более того, гамму шифра необязательно рассматривать как объединение непересекающихся множеств. Например, гамма шифра может быть представлена в виде

$$G = H(1) (+) H(2) (+) ... (+) H(N),$$

где символ (+) обозначает операцию "исключающее ИЛИ".

Шифрование с помощью датчика ПСЧ является довольно распространенным криптографическим методом, а качество шифра определяется не только и не столько характеристиками датчика, сколько алгоритмом получения гаммы. Хорошие результаты дает метод гаммирования с обратной связью, который заключается в том, что для получения сегмента гаммы используется контрольная сумма определенного участка шифруемых данных.

2.6. Примеры потоковых шифров

2.6.1. АЛГОРИТМ *RC*4

Алгоритм *RC*4 представляет собой потоковый шифр с переменной длиной ключа, разработанный в 1987 г. Роном Ривестом для компании *RSA Data Security*, *Inc.* В течение семи лет этот шифр лицензировался компанией только на условиях неразглашения. Однако, в 1994 г. он был анонимно опубликован в Интернете и с тех пор стал доступен для независимого анализа.

Описывается шифр очень просто. Алгоритм работает в режиме *OFB*. Ключевая последовательность не зависит от исходного текста. Структура алгоритма включает блок замены размерностью 8×8: $S_0, ..., S_{255}$. Блок замены представляет собой зависимую от ключа переменной длины перестановку чисел 0, ..., 255. Имеется два счетчика i и j, первоначально равные 0. Для генерирования псевдослучайного байта выполняются следующие действия:

$$i = (i + 1) \bmod 256,$$
$$j = (j + S_i) \bmod 256.$$

Далее переставить S_i и S_j:

$$t = (S_i + S_j) \bmod 256,$$
$$k = S_i.$$

Затем байт k складывается по модулю 2 с байтом исходного текста для получения шифрованного.

Инициализация блока замены также проста. Вначале он заполняется линейно: $S_0 = 0$, ..., $S_{255} = 255$. Затем заполняется еще один 256-байтный массив ключом, при этом ключ может повторяться необходимое число раз для заполнения всего массива: k_0, ..., k_{255}. Счетчик j устанавливается в 0. После чего производятся следующие действия:

for i = 0 *to* 255
$j = (j + k_i + S_i) \bmod 256$

переставить S_i и S_j.

Шифрование по этому алгоритму, примерно, в 10 раз быстрее, чем шифрование DES при программной реализации.

Возможно обобщение алгоритма на большие длину слова и размер блока замены. Например, можно построить шифр с блоком замены размерностью 16 × 16 и длиной слова 16 бит. Этап инициализации будет значительно медленнее, необходим цикл до 65535, но получившийся в результате алгоритм будет более быстрым.

2.6.2. АЛГОРИТМ *SEAL*

Алгоритм *SEAL* (*Software Encryption Algorithm*) представляет собой приспособленный для программной реализации потоковый шифр, разработанный Филом Рогэвэем и Доном Копперсмитом из компании *IBM*. Алгоритм оптимизирован для 32-разрядных процессоров. Для эффективной работы ему требуются восьми 32-разрядных регистров и кэш объемом несколько килобайт.

Одним из замечательных свойств этого шифра является то, что он не является потоковым шифром в традиционном смысле, а представляет собой семейство псевдослучайных функций. 160-битовый ключ k и 32-битовое значение n (индекс) шифр преобразует в L-битовую строку $k(n)$. L может принимать любое значение, меньшее 64 килобайт. Такой шифр обозначают $SEAL(k, n, L)$. Предполагается, что если k выбирается случайно, то $k(n)$ будет вычислительно неотличима от случайной L-битовой функции от n.

Большинство шифров генерирует битовые последовательности в одном направлении. Зная ключ k и позицию i, определить значение i-го бита ключевой последовательности можно, только вычислив все биты до i-го один за другим. В случае семейства псевдослучайных функций можно получить простой доступ к любому элементу ключевой последовательности.

Предположим, что необходимо зашифровать содержимое жесткого диска компьютера по 512-байтным секторам. Тогда сектор с номером n будет шифроваться с помощью ключевой последовательности $k(n)$. При этом легко может быть обеспечен доступ к произвольному сектору диска.

Данный шифр также облегчает проблему синхронизации, свойственную традиционным потоковым шифрам. Сообщение с номером n будет шифроваться с ключевой последовательностью $k(n)$, и n будет передаваться вместе с сообщением. Получателю не нужно будет хранить состояние генератора ключевой последовательности и беспокоиться о потерянных сообщениях и их влиянии на процесс расшифрования.

Алгоритм *SEAL* предусматривает использование трех зависящих от ключа таблиц: R, S и T заполняются на предварительном этапе при помощи алгоритма, основанного на *SHA* (описан далее), и зависят только от ключа.

Заполнение таблиц можно описать с помощью функции $G_a(i)$, которая представляет собой функцию сжатия из алгоритма *SHA*: a – 160-битное значение, i – 32-битный индекс. Значение a используется для инициализации внутренних регистров A, B, C, D и E в алгоритме *SHA*, а 512-битный

блок для обработки представляет собой строку $i||0^{480}$. Выходное значение функции G также имеет длину 512 бит.

Основные идеи, лежащие в основе этого алгоритма, можно сформулировать следующим образом:

1. использование большого секретного, зависящего от ключа блока замены T;
2. перемежение операций, не коммутирующих друг с другом (сложение и исключающее ИЛИ);
3. использование внутреннего состояния, явно не проявляющегося в потоке данных (значения n_i, используемые в конце итерации для модификации регистров A и C);
4. изменение шаговой функции в зависимости от номера шага и изменение итерационной функции в зависимости от номера итерации;
5. использование известных и отработанных алгоритмов для заполнения таблиц.

Шифр *SEAL* требует пять элементарных машинных операций в пересчете на один байт текста при шифровании и расшифровании. Таким образом, он является одним из самых быстрых программно реализуемых алгоритмов.

2.6.3. АЛГОРИТМ *WAKE*

Алгоритм *WAKE* (*Word Auto Key Encryption* – шифрование слов с автоключом) был предложен Дэвидом Уилером. На выходе получается последовательность 32-битовых слов, которые могут служить в качестве гаммы шифра. *WAKE* работает в режиме *CFB* – предыдущее слово шифрованного текста используется для генерации следующего слова ключевой последовательности.

В алгоритме используется специальный блок замены S из 256 32-битных слов. При этом старшие байты этих слов являются перестановкой чисел от 0 до 255, а остальные три младших байта выбираются случайными.

Вначале инициализируется блок замены на основе ключа. Затем инициализируются четыре регистра A, B, C и D начальными значениями, также зависящими от ключа (возможно другого): a_0, b_0, c_0, d_0. Очередное слово ключевой последовательности получается по формуле $k_i = d_i$.

После этого изменяется значение регистров:

$$a_{i+1} = M(a_i, d_i);$$
$$b_{i+1} = M(b_i, a_{i+1});$$
$$c_{i+1} = M(c_i, b_{i+1});$$
$$d_{i+1} = M(d_i, c_{i+1}).$$

где $M(x, y) = (x + y) \gg 8 \oplus S_{(x+y)\&255}$ $ здесь 8 младших бит суммы $x + y$ используются для входа в таблицу замены.

Данный шифр является достаточно быстрым, хотя и нестойким к атакам по выбранному исходному тексту.

2.7. Комбинированные методы

Шифрование комбинированными методами основывается на результатах, полученных К. Шенноном. Наиболее часто применяются такие комбинации, как подстановка и гамма, перестановка и гамма, подстановка и перестановка, гамма и гамма. При составлении комбинированных шифров необходимо проявлять осторожность, так как неправильный выбор составлявших шифров может привести к исходному открытому тексту.

В качестве примера можно привести шифр, предложенный Д. Френдбергом, который комбинирует многоалфавитную подстановку с генератором ПСЧ. Особенность данного алгоритма состоит в том, что при большом объеме шифртекста частотные характеристики символов шифртекста близки к равномерному распределению независимо от содержания открытого текста.

Комбинация методов подстановки и перестановки была применена в 1974 г. фирмой IBM при разработке системы ЛЮЦИФЕР.

Система ЛЮЦИФЕР строится на базе блоков подстановки (*S*-блоков) и блоков перестановки (*P*-блоков). Блок подстановки включает линейные и

нелинейные преобразования.

Первый преобразователь *S*-блока осуществляет развертку двоичного числа из *n* разрядов в число по основанию 2^n. Второй преобразователь осуществляет свертку этого числа.

Блок перестановки осуществляет преобразование *n* разрядного входного числа в *n* разрядное число.

Входные данные (открытый текст) последовательно проходят через чередующиеся слои 32-разрядных *P*-блоков и 8-разрядных *S*-блоков.

Реализация шифрования данных в системе ЛЮЦИФЕР программными средствами показала низкую производительность, поэтому *P*- и *S*-блоки были реализованы аппаратно, это позволило достичь скорости шифрования до 100 Кбайт/с. Опыт, полученный при разработке и эксплуатации системы, позволил создать стандарт шифрования данных *DES* (*Data Encryption Standard*).

2.7.1. САМОСИНХРОНИЗИРУЮЩИЕСЯ ШИФРЫ

В 1946 г. в США была запатентована базовая идея так называемых *самосинхронизирующихся потоковых шифров* (или *шифрования с автоключом – CipherText Auto Key* (*CTAK*)). Она заключается в том, что внутреннее состояние генератора является функцией фиксированного числа предшествующих битов шифрованного текста. Поскольку внутреннее состояние зависит только от *n* бит шифрованного текста, генератор на приемной стороне войдет в синхронизм с передающей стороной после получения *n* бит.

Реализация этого подхода выглядит следующим образом. Каждое сообщение предваряется случайным заголовком длиной *n* бит. Этот заголовок шифруется и передается в линию. На приемной стороне заголовок расшифровывается. Результат расшифрования будет неверным, но после обработки *n* бит заголовка оба генератора будут синхронизированы.

Недостатком системы является распространение ошибок. При искажении одного бита генератор на приемной стороне выдаст *n* неверных бит ключевой последовательности, пока ошибочный бит не будет вытолкнут из памяти, что приведет к ошибочному расшифрованию *n* бит исходного текста.

Кроме того, самосинхронизирующиеся шифры уязвимы для атак типа "воспроизведение". Злоумышленник записывает некоторое количество бит шифрованного текста. Затем, позднее, он подменяет биты трафика записанными – "воспроизводит" их. После некоторого количества "мусора", пока приемная сторона не синхронизируется, старый шифрованный текст будет расшифровываться нормально. У приемной стороны нет никаких средств определения того, что принимаемые данные не являются актуальными.

Самосинхронизирующиеся потоковые шифры могут быть реализованы в виде блочных шифров, используемых в режиме обратной связи по шифрованному тексту. При этом за один раз может шифроваться произвольное число бит, меньшее либо равное длине блока. Проиллюстрируем это на примере шифрования по одному байту за цикл.

Блочный шифр работает над очередью размером, равным длине блока. Первоначально очередь заполняется синхропосылкой. Затем очередь шифруется и левые 8 бит складываются с первыми 8 битами исходного текста. Полученные 8 бит шифрованного текста передаются в линию, очередь сдвигается влево на 8 бит, левые биты отбрасываются, а правые заполняются 8 битами шифрованного текста, переданными в линию. Далее процедура повторяется. Число 8 взято только для примера. За один цикл работы блочного алгоритма может шифроваться и один бит, хотя это будет не слишком эффективно с точки зрения скорости работы схемы. В режиме *CPB* синхропосылка должна быть уникальна для каждого сообщения в течение срока действия ключа. Если это будет не так, злоумышленник сможет восстановить исходный текст.

В случае возникновения ошибки в шифрованном тексте на приемной стороне в общем случае возникнут ошибки при расшифровании текущего и последующих $[m/n]$ блоков, где m – размер блока; n – число бит, шифруемых за один цикл, т.е. пока ошибочный бит шифрованного текста не будет вытеснен из памяти.

2.7.2. СИНХРОННЫЕ ШИФРЫ

Потоковые шифры носят название *синхронных*, если выходные значения генератора не зависят от исходного или шифрованного текстов.

Основная сложность в данном подходе заключается в необходимости синхронизации генераторов ключа на передающей и приемной сторонах. Если в процессе передачи произошло выпадение или вставка хотя бы одного бита, то вся последовательность битов шифрованного текста после ошибочного бита не сможет быть расшифрована. Если такое произойдет, стороны должны провести повторную синхронизацию. При этом синхронизация должна быть проведена так, чтобы никакой отрезок ключевой последовательности не повторился, так что очевидное решение возвратиться к некоторому предыдущему состоянию генератора не подходит.

Положительным свойством синхронных потоковых шифров является отсутствие эффекта распространения ошибок. Один искаженный бит при передаче приведет к искажению только одного бита текста при расшифровании.

Синхронные шифры также защищают от вставок и выбрасываний отрезков шифрованного текста из потока. Такие операции приведут к нарушению синхронизации, что будет сразу же обнаружено на приемной стороне.

Однако, такие шифры уязвимы к изменению отдельных бит. Если злоумышленник знает исходный текст, то он сможет изменять биты в потоке шифрованного текста таким образом, что он будет расшифровываться так, как необходимо злоумышленнику.

Синхронный потоковый шифр может быть реализован в виде блочного шифра, работающего в режиме обратной связи по выходу (режим *OFB*) с любым размером обратной связи, меньшим размера блока. Однако, данный способ не рекомендуется, так как при этом снижается период генератора до, примерно, $2^{m/2}$. При длине блока 64 это будет около 2^{32}, чего явно недостаточно.

В этом случае выходная функция очень часто выбирается простой, например, суммой по модулю 2 нескольких бит состояния или вообще одним битом состояния. Криптографическая стойкость обеспечивается функцией перехода к следующему состоянию, которая зависит от ключа. Иногда данный режим называют режимом *с внутренней обратной связью,* поскольку обратная связь является внутренней по отношению к алгоритму генерации ключевой последовательности.

Вариантом этого режима является схема, когда ключ задает начальное состояние генератора, после чего последний работает без дальнейшего вмешательства.

Еще одним способом построения потокового шифра является использование счетчика в качестве входного значения для блочного шифра. После каждого цикла шифрования блока значение счетчика увеличивается, чаще всего на единицу. Свойства данного режима в отношении распространения ошибок и синхронизации будут такими же, как и для режима *OFB*. В качестве счетчика может быть использован любой генератор псевдослучайных чисел, вне зависимости от его криптографической стойкости.

При использовании потокового шифра в режиме счетчика выбирается простая функция перехода и сложная, зависящая от ключа функция выхода. Функция перехода может быть простым счетчиком, увеличивающимся на единицу на каждом такте.

2.7.4. МЕТОДЫ КОДИРОВАНИЯ

Как уже отмечалось, под кодированием понимается замена элементов открытого текста (букв, слов, фраз и т.п.) кодами. Различают символьное и смысловое кодирование.

При *символьном кодировании* каждый знак алфавита открытого текста заменяется соответствующим символом. Примером символьного кодирования служит азбука Морзе, а также методы шифрования заменой и перестановкой. Рассмотрим метод символьного кодирования, который использует предыдущие символы открытого текста. Этот метод, называемый ме-

тодом стопки книг, был предложен Б.Я. Рябко.

Предположим, что нужно передать сообщение X из алфавита A, в котором буквы алфавита отождествлены с числами 1, 2, …, L, где L – число элементов алфавита A. Каждой букве алфавита соответствует код k_i, 1 = 1, ..., L. При появлении в сообщении X очередной буквы x_j ее код представляется кодом номера позиции j, занимаемой в данный момент буквой x_j в списке. Это дает возможность на приемном конце по коду номера позиции j определить букву x_j. После кодирования буквы x_j одновременно на приемном и передающих концах перемещают букву x_j в начало списка, увеличивая тем самым на единицу номера букв, стоявших на позициях от 1 до j – 1. Номера букв, стоявших на позициях от j + 1 до L, остаются без изменений. В результате кодирования открытого текста в начале списка будут находиться буквы, которые наиболее часто встречались в открытом тексте.

Интересный метод кодирования в 1992 г. предложил С.П. Савчук. В отличие от метода стопки книг перемещению подвергается список кодов. Пусть алфавит $A = \{a_1, a_2, ..., a_n\}$. Данному порядку расположения букв соответствует начальный список кодов $K_0 = \{k_1, k_2, ..., k_n\}$. При появлении в кодируемом сообщении буквы a_i в качестве кода выбирается соответствующий ее местоположению код k_i. После этого осуществляется сдвиг списка кодов:

$$\{k_1, k_2, ..., k_i, ..., k_n\} \rightarrow \{k_2, k_3, ..., k_n, k_1\}.$$

Таким образом, список кодов образует замкнутое кольцо.

Смысловое кодирование – это кодирование, в котором в качестве исходного алфавита используются не только отдельные символы (буквы), но и слова и даже наиболее часто встречающиеся фразы.

Рассмотрим пример одноалфавитного и многоалфавитного смыслового кодирования.

Пример. Открытый текст: "19.9.1992 ГОДА" (табл. 2.10).

2.10. Таблица кодирования

Элементы открытого текста	Коды
1	089 146 214 417
2	187 226 045 361
9	289 023 194 635
ГОД	031 155 217 473
.	786 432 319 157

Закодированное сообщение при одноалфавитном кодировании:

"089 289 786 289 786 089 289 289 187 031".

Закодированное сообщение при многоалфавитном кодировании:

"089 289 786 023 432 146 194 635 187 031" (при многоалфавитном кодировании одинаковые символы заменяются кодами из следующего столбца).

Среди различных кодов, применяемых для кодирования естественных языков, особый интерес вызывает код Хаффмена, который позволяет сжимать открытый текст. Суть его состоит в присваивании наиболее часто встречающимся буквам наиболее коротких кодов.

Строка двоичных символов кодов Хаффмена единственным образом разлагается на коды символов (такие коды называются префиксными).

Пример. Закодированное кодом Хаффмена сообщение имеет вид:

"01101000100000010101111000100000".

Пользуясь деревом для английского языка, получаем 0110=S.

Далее снова начинаем движение из вершины: 100=E; 01000=C; 00010=U; 1011=R; 1010=I; 001=T; 00000=Y.

Открытый текст: "SECURITY".

2.7.5. ДРУГИЕ МЕТОДЫ

Широкое применение персональная ЭВМ (ПЭВМ) сделало актуальной задачу защиты хранящихся данных (файлов). Для защиты файлов могут быть применены рассмотренные методы шифрования и кодирования.

Специфика применения ПЭВМ позволяет реализовать дополнитель-

ные методы кодирования для надежного закрытия содержимого файлов. Примером такого кодирования является метод рассечения-разнесения, в соответствии с которым содержимое одного файла разбивается на блоки, которые разносятся по нескольким файлам. Каждый такой файл не несет никакой информации, а сбор данных в единое целое осуществляется простой программой.

Пример. Блок (файл открытого текста) начинается словами: "МЕТОД_РАССЕЧЕНИЯ-РАЗНЕСЕНИЯ".

Для рассечения блока открытого текста на 8 частей запишем открытый текст в следующем виде (табл. 2.11).

2.11. Рассечения открытого текста

	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	М	Е	Т	О	С	С	Е	Ч	–	Р	А	З	Н	И	Я	_
2	Д	_	Р	А	Е	Н	И	Я	Н	Е	С	Е				

Для рассечения текста на 8 частей выбраны 2 строки и 4 столбца. Пусть столбцы s_j выбираются в последовательности {4, 1, 3, 2}, а строки r_i – в последовательности {2, 1}. Тогда номер k блока Φ_k, куда записывается очередной символ открытого текста, определяется по формуле

$$k = (r_i - 1)n + s_j,$$

где n – число столбцов.

Первый символ M запишется в блок с номером ($r_i = 2$, $s_j = 4$) : $k = (2 - 1)*4 + 4 = 8$; второй символ E – в блок с номером ($r_i = 2$, $s_j = 1$) : $k = (2 - 1)*4 + 1 = 5$, и т.д.

Тогда блоки Φ_k, записанные в порядке номеров, будут содержать следующие символы: Φ_1 = (_НЕ...), Φ_2 = (АЯЕ...), Φ_3 = (РИС..,), Φ_4 = {ДЕН...), Φ_5 = {ЕСРИ...}, Φ_6 = {ОЧЗ...), Φ_7 = {ТЕАЯ...), Φ_8 = {МСН...}. Таким образом, один блок открытого текста заменяется восемью блоками, которые в сумме дают длину исходного блока.

Одной из важных проблем при использовании ПЭВМ является проблема хранения больших массивов данных. Для этой цели применяют различные методы сжатия данных (сжатие рассматривается как метод кодирования).

Методы сжатия данных осуществляют такое преобразование повторяющихся символов и строк символов, которое позволяет использовать для хранения данных меньший объем памяти. Методы сжатия можно разделить на два класса: *статические и динамические* (адаптивные).

Методы статического сжатия данных эффективны, когда частоты появления символов изменяются незначительно. Методы динамического сжатия адаптивно отслеживают неравномерности частот появления символов с сохранением последовательности изменений вероятностей появления символов.

Адаптивные методы сжатия могут динамично реагировать на изменения в открытом тексте, происходящие по мере кодирования. Первые такие методы являлись модификацией кодов Хаффмена и использовали счетчики для хранения текущих частот появления каждого символа. При таких методах наиболее часто встречающиеся символы сдвигаются ближе к корню дерева и, следовательно, получают более короткие кодовые слова.

Кодирование Лемпеля-Зива использует синтаксический метод для динамического источника. Этот метод осуществляет синтаксический анализ символьных потоков, которые не превышают заданной длины, и строит таблицу отображения этих потоков в кодированные слова фиксированной длины. Длина кодового слова зависит от размера таблицы, используемой для хранения кодового отображения поток-слово. Например, размер таблицы в 4096 слов требует 12-битового кодового слова. Кодовое слово является просто табличным адресом соответствующих слов в таблице.

При кодировании по методу Лемпеля-Зива-Уэлча таблица инициализируется символьным множеством и содержит вместо потоков заданной длины пары (кодовое слово, символ) фиксированной длины. Таблица строится на основе синтаксического анализа самого длинного опознанного в таблице потока и использовании последующего символа для формирования нового входа в таблицу. Это позволяет уменьшить размеры таблицы.

В последнее время широкое распространение получили методы сжатия на основе расширяющихся деревьев. Префиксный код переменной длины в этих методах строится на основе положения символов в дереве. Для получения оптимальных кодов дерево балансируется.

Несомненно, криптография должна стать обязательным компонентом защиты всех сколько-нибудь развитых систем. К сожалению, этому мешает огромное количество самых разных барьеров.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Дайте определение шифра и сформулируйте основные требования к нему.
2. **Назовите и охарактеризуйте методы шифрования.**
3. Дешифруйте сообщение, зашифрованное с помощью шифра простой замены: " 11 03 18 21 11 04 15 02 13 29 12 10 01 28 02 18 21 11 01 28 28 12 30 12 21 32 04 00 27 32 01 21 11 12 27 32 01 32 12 14 28 12 21 28 01 23 32 02 21 22 04 27 32 12 21 28 01 21 00 12 32 12 29 12 22 21 28 01 06 12 11 02 32 27 18 21 27 03 12 10 12 21 11 04 15 02 13 29 12 10 01 28 02 04".
4. Поясните, что вы понимаете под совершенным шифром. Приведите пример совершенного шифра.
5. Дешифруйте следующее сообщение: "15 03 23 21 22 32 14 06 25 14 04 07 24 16 31" методом перебора всех возможных ключей. Исходное сообщение состоит из цифр от 0 до 32, которыми закодированы буквы в соответствии с табл. 4. Шифрование произведено сложением по модулю 32 с псевдослучайной гаммой: K, (K^2) mod 32, …, (K^n) mod 32, где K – неизвестный ключ из множества чисел от 0 до 32.
6. Перечислите основные режимы работы, предусмотренные в российском стандарте на шифрование данных.
7. Проведите сравнительный анализ параметров алгоритмов шифрования *DES*, *AES* и Российского стандарта (в режиме простой замены).
8. Сравните наиболее распространенные стандарты шифрования.

3. АСИММЕТРИЧНЫЕ КРИПТОСИСТЕМЫ

3.1. Общие положения

Еще одним обширным классом криптографических систем являются так называемые асимметричные или двухключевые системы. Эти системы характеризуются тем, что для шифрования и для расшифрования используются разные ключи, связанные между собой некоторой зависимостью. Применение таких шифров стало возможным благодаря К. Шеннону предложившему строить шифр таким способом, чтобы его раскрытие было эквивалентно решению математической задачи, требующей выполнения объемов вычислений, превосходящих возможности современных ЭВМ (например, операции с большими простыми числами и их произведениями).

Один из ключей (например, ключ шифрования) может быть сделан общедоступным, и в этом случае проблема получения общего секретного ключа для связи отпадает. Если сделать общедоступным ключ расшифрования, то на базе полученной системы можно построить си-стему аутентификации передаваемых сообщений. Поскольку в большинстве случаев один ключ из пары делается общедоступным, такие системы получили также название криптосистем с открытым ключом.

Первый ключ не является секретным и может быть опубликован для использования всеми пользователями системы, которые зашифровывают данные. Расшифрование данных с помощью известного ключа невозможно. Для расшифрования данных получатель зашифрованной информации использует второй ключ, который является секретным. Разумеется, ключ расшифрования не может быть определен из ключа зашифрования.

Использование асимметричного шифрования иллюстрирует рис. 3.1.

Криптосистема с открытым ключом определяется тремя алгоритмами: генерации ключей, шифрования и расшифрования. Алгоритм генерации

ключей открыт, всякий может подать ему на вход случайную строку r надлежащей длины и получить пару ключей (k_1, k_2). Один из ключей (например, k_1) публикуется, он называется открытым, а второй, называемый секретным, хранится в тайне. Алгоритмы шифрования E_k и расшифрования D_k таковы, что для любого открытого текста m

$$D_{k_2}(E_{k_1}(m)) = m.$$

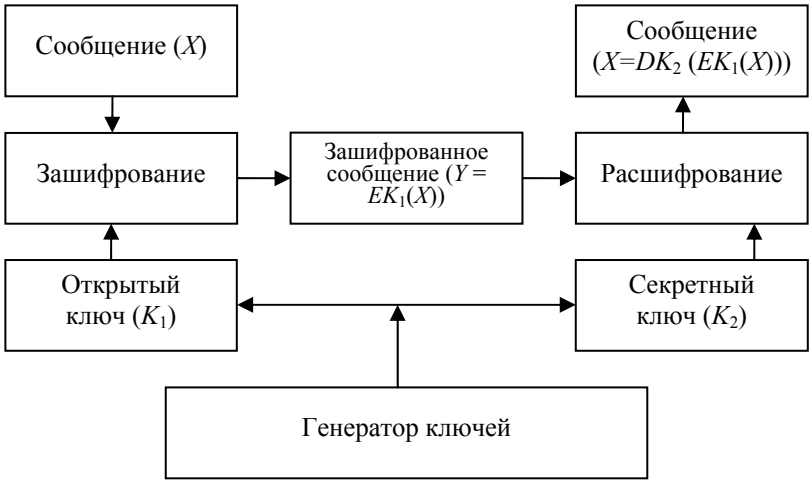


Рис. 3.1. Использование асимметричного метода шифрования

Рассмотрим теперь гипотетическую атаку злоумышленника на эту систему. Противнику известен открытый ключ k_1, но неизвестен соответствующий секретный ключ k_2. Противник перехватил криптограмму d и пытается найти сообщение m, где $d = E_k(m)$. Поскольку алгоритм шифрования открыт, противник может просто последовательно перебрать все возможные сообщения длины n, вычислить для каждого такого сообщения m_i криптограмму $d_i = E_k(m_i)$ и сравнить d_i с d. Тогда сообщение, для которого $d_i = d$, и будет искомым открытым текстом. Если повезет, то открытый текст будет найден достаточно быстро. В худшем же случае перебор будет выполнен за время порядка $2^n T(n)$, где $T(n)$ – время, требуемое для шифрования сообщения длины n. Если сообщения имеют длину порядка 1000 битов, то такой перебор неосуществим на практике ни на каких самых мощных компьютерах.

Такой способ атаки на криптосистему и простейший алгоритм поиска открытого текста называется алгоритмом полного перебора. Используется также и другое название: "метод грубой силы". Другой простейший алгоритм поиска открытого текста – угадывание. Этот очевидный алгоритм требует небольших вычислений, но срабатывает с пренебрежимо малой вероятностью (при больших длинах текстов). На самом деле противник может пытаться атаковать криптосистему различными способами и использовать различные, более изощренные алгоритмы поиска открытого текста.

Кроме того, злоумышленник может попытаться восстановить секретный ключ, используя знания (в общем случае несекретные) о математической зависимости между открытым и секретным ключами. Естественно считать криптосистему стойкой, если любой такой алгоритм требует практически неосуществимого объема вычислений или срабатывает с пренебрежимо малой вероятностью. Это и есть теоретико-сложностный подход к определению стойкости. Для его реализации в отношении того или иного типа криптографических систем необходимо выполнить следующее: дать формальное определение системы данного типа; дать формальное определение стойкости системы; доказать стойкость конкретной конструкции системы данного типа.

Здесь сразу же возникает ряд проблем.

Во-первых, для применения теоретико-сложностного подхода необходимо построить математическую модель криптографической си-стемы, зависящую от некоторого параметра, называемого параметром безопасности, который может принимать сколь угодно большие значения (обычно предполагается, что параметр безопасности может пробегать весь натуральный ряд).

Во-вторых, определение стойкости криптографической системы зави-

сит от той задачи, которая стоит перед противником, и от того, какая информация о схеме ему доступна. Поэтому стойкость систем приходится определять и исследовать отдельно для каждого предположения о противнике.

В-третьих, необходимо уточнить, какой объем вычислений можно считать "практически неосуществимым". Из сказанного следует, что эта величина не может быть просто константой, она должна быть представлена функцией от растущего параметра безопасности. В соответствии с тезисом Эдмондса алгоритм считается эффективным, если время его выполнения ограничено некоторым полиномом от длины входного слова (от параметра безопасности). В противном случае говорят, что вычисления по данному алгоритму практически неосуществимы. При этом сами криптографические системы должны быть эффективными, т.е. все вычисления, предписанные той или иной схемой, должны выполняться за полиномиальное время.

В-четвертых, необходимо определить, какую вероятность можно считать пренебрежимо малой. В криптографии принято считать таковой любую вероятность, которая для любого полинома p и для всех достаточно больших n не превосходит $1/p(n)$, где n – параметр безопасности.

Итак, при наличии всех указанных выше определений, проблема обоснования стойкости криптографической системы сводится к доказательству отсутствия полиномиального алгоритма, который решает задачу, стоящую перед противником. Но здесь возникает еще одно и весьма серьезное препятствие: современное состояние теории сложности вычислений не позволяет доказывать сверхполиномиальные нижние оценки сложности для конкретных задач рассматриваемого класса. Из этого следует, что на данный момент стойкость криптографических систем может быть установлена лишь с привлечением каких-либо недоказанных предположений. Поэтому основное направление исследований состоит в поиске наиболее слабых достаточных условий (необходимых и достаточных) для существования стойких систем каждого из типов. В основном, рассматриваются предположения двух типов – общие (или теоретико-сложностные) и теоретико-числовые, т.е. предположения о сложности конкретных теоретико-числовых задач. Все эти предположения в литературе обычно называются криптографическими.

3.2. Односторонние функции и функции-ловушки

Центральным понятием в теории асимметричных криптографических систем является понятие односторонней функции.

Под *односторонней функцией* понимается эффективно вычислимая функция, для обращения которой (т.е. для поиска хотя бы одного значения аргумента по заданному значению функции) не существует эффективных алгоритмов. Заметим, что обратная функция может и не существовать.

Под *функцией* будем понимать семейство отображений $\{f_n\}$, где

$$f_n : \sum^n \to \sum^m,\ m = m(n).$$

Для простоты предположим, что n пробегает натуральный ряд, а отображения f_n, определены всюду. Функция f называется *честной*, если $\exists q(x), \forall n\ \ q(m(n)) \geq n$.

Функцией-ловушкой называется односторонняя функция, для которой обратную функцию вычислить просто, если имеется некоторая дополнительная информация, и сложно, если такая информация отсутствует.

В качестве задач, приводящих к односторонним функциям, можно привести следующие:

1. Разложение числа на простые сомножители. Вычислить произведение двух простых чисел очень просто. Однако, для решения обратной задачи – разложения заданного числа на простые сомножители – эффективного алгоритма в настоящее время не существует.

2. Дискретное логарифмирование в конечном простом поле (проблема Диффи-Хеллмана). Допустим, задано большое простое число p и пусть g – примитивный элемент поля $GF(p)$. Тогда для любого a вычислить $g^a \pmod p$ просто, а вычислить a по заданным $k = g^a \pmod p$ и p оказывается затруднительным.

Криптосистемы с открытым ключом основываются на односторонних

функциях-ловушках. При этом открытый ключ определяет конкретную реализацию функции, а секретный ключ дает информацию о ловушке. Любой, знающий ловушку, может легко вычислять функцию в обоих направлениях, но тот, у кого такая информация отсутствует, может производить вычисления только в одном направлении. Прямое направление используется для шифрования и для верификации цифровых подписей, а обратное – для расшифрования и выработки цифровой подписи.

Во всех криптосистемах с открытым ключом, чем больше длина ключа, тем выше различие между усилиями, необходимыми для вычисления функции в прямом и обратном направлениях (для того, кто не обладает информацией о ловушке).

Все практические криптосистемы с открытым ключом основываются на функциях, считающихся односторонними, но это свойство не было доказано в отношении ни одной из них. Это означает, что теоретически возможно создание алгоритма, позволяющего легко вычислять обратную функцию без знания информации о ловушке. В этом случае, криптосистема, основанная на этой функции, станет бесполезной. С другой стороны, теоретические исследования могут привести к доказательству существования конкретной нижней границы сложности обращения некоторой функции, и это доказательство будет существенным событием, которое окажет значительное позитивное влияние на развитие криптографии.

3.3. Асимметричные системы шифрования

3.3.1. КРИПТОСИСТЕМА ЭЛЬ-ГАМАЛЯ

Система Эль-Гамаля – это криптосистема с открытым ключом, основанная на проблеме логарифма, Система включает как алгоритм шифрования, так и алгоритм цифровой подписи.

Множество параметров системы включает простое число p и целое число g, степени которого по модулю p порождают большое число элементов Z_p. У пользователя A есть секретный ключ a и открытый ключ y, где $y = g^a \pmod p$. Предположим, что пользователь B желает послать сообщение m пользователю A. Сначала B выбирает случайное число k, меньшее p, и вычисляет

$$y_1 = g^k \pmod p, \quad y_2 = m \oplus (y^k \pmod p),$$

где $\oplus$ обозначает побитовое "исключающее ИЛИ". B посылает A пару (y_1, y_2).

После получения шифрованного текста пользователь A вычисляет

$$m = (y_1^a \bmod p) \oplus y_2 .$$

Известен вариант этой схемы, когда операция $\oplus$ заменяется на умножение по модулю p. Это удобнее в том смысле, что в первом случае текст (или значение хэш-функции) необходимо разбивать на блоки той же длины, что и число $y^k \pmod p$. Во втором случае этого не требуется и можно обрабатывать блоки текста заранее заданной фиксированной длины (меньшей, чем длина числа p). Уравнение расшифрования в этом случае будет таким:

$$m = y_2 / y_1^k \bmod p .$$

Схема Эль-Гамаля не лишена определенных недостатков, среди них можно указать следующие: отсутствие семантической стойкости; делимость шифра. Для защиты от недостатков Шнорром и Якобссоном было предложено объединить схему шифрования Эль-Гамаля с цифровой подписью Шнорра, что позволяет не только шифровать сообщение, но и аутентифицировать его.

3.3.2. КРИПТОСИСТЕМА, ОСНОВАННАЯ НА ПРОБЛЕМЕ ДИФФИ-ХЕЛЛМАНА

Система шифрования была представлена Мишелем Абдаллой, Михиром Беллэром и Филлипом Рогэвэем в рамках европейского проекта *NESSIE* (*New European Schemes for Signatures*, *Integrity and Encryption*). Она столь же эффективна, что и система Эль-Гамаля, но обладает дополнительными свойствами безопасности. Более того, стойкость системы может быть

доказана в предположении о стойкости лежащих в ее основе криптографических примитивов.

Данная криптосистема реализуема на основе любой циклической группы, для которой может быть сформулирована проблема Диффи-Хеллмана, например в $\{Z_p^*\}$ или в группе точек на эллиптической кривой.

Система строится из криптографических примитивов низкого уровня: групповой операции, симметричного шифра, функции хэширования и алгоритма вычисления кода аутентификации сообщения-имитовставки (*MAC*). Стойкость доказывается на основе предположения о сложности решения соответствующей проблемы Диффи-Хеллмана и предположения о стойкости входящих в схему симметричных примитивов.

Опишем криптографические примитивы, входящие в схему.

Циклическая группа $G = \{g\}$. Далее будем использовать мультипликативную запись групповой операции. Алгоритмы, реализующие эту операцию, будут работать с *представлениями* элементов группы в виде битовых строк фиксированной длины $gLen \in N$. Способ кодирования $G \to \{0,1\}^{gLen}$ не фиксируется и может быть выбран из соображений эффективности.

Код аутентификации сообщения позволяет пользователям, обладающим общим секретным ключом, выработать битовую строку для аутентификации и проверки целостности данных Пусть $Msg = \{0,1\}^*$ – пространство сообщений, $mKey = \{0,1\}^{mLen}$ – пространство ключей для вычисления *MAC* для некоторого $mLen \in N$, $Tag = \{0,1\}^{tLen}$ – включающее множество всех возможных значений *MAC* для некоторого $tLen \in N$. В этих обозначениях код аутентификации сообщений представляет собой пару алгоритмов $MAC = \{MAC.gen, MAC.ver\}$. Алгоритм генерации *MAC* определяется как отображение $MAC.gen(k, x) : mKey \times Msg \to Tag$ и может быть вероятностным.

Алгоритм верификации *MAC* является отображением со свойством $MAC.ver(k, x, MAC.gen(k, x)) = 1$.

В качестве *MAC* можно использовать, например, блочный шифр с достаточной длиной блока и ключа в режиме сцепления блоков шифрованного текста.

Симметричный шифр позволяет пользователям, обладающим общим секретным ключом, обеспечить секретность. Пусть *Msg*, как и ранее, пространство сообщений, $eKey = \{0,1\}^{eLen}$ – пространство ключей для некоторого $eLen \in N$, $Ctext = \{0,1\}^*$ – включающее множество всех возможных значений шифрованного текста и $Coins = \{0,1\}^\infty$ – множество строк бесконечной длины. В этих обозначениях шифр представляет собой пару алгоритмов $SYM = \{SYM.enc, SYM.dec\}$. Алгоритм зашифрования определяется как отображение

$$SYM.enc(k, x, r) : eKey \times Msg \times Coins \to Ctext,$$

алгоритм расшифрования является отображением

$$SYM.dec(k, y) : eKey \times Ctext \to M \cup \{BAD\},$$

где значение *BAD* выдается, если шифртекст *y* не является результатом зашифрования никакого открытого текста

Асимметричный шифр. Пусть *Msg*, *Ctext*, *Coins* определены как и ранее, $PK \subseteq \{0,1\}^*$, $SK \subseteq \{0,1\}^*$ – множества открытых и секретных ключей. Асимметричный шифр определяется как тройка алгоритмов $ASYM = \{ASYM.enc, ASYM.dec, ASYM.Key\}$. Алгоритм зашифрования является отображением

$$ASYM.enc(pk, x, r) : PK \times Msg \times Coins \to Ctext,$$

а расшифрования:

$$ASYM.enc(sk, y) : SK \times Ctext \to M \cup \{BAD\}.$$

Алгоритм выработки ключа в качестве аргумента берет строку $r \in Coins$ и выдает пару ключей $pk, sk \in PK \times SK$. При этом должно выполняться следующее свойство:

$$\forall (pk, sk) : \exists r' \in Coins : (pk, sk) = ASYM.key(r'), \forall r \in Coi;$$
$$\forall x \in Msg\ ASYM.dec(sk, ASYM.enc(pk, x, r)) =.$$

Функция хэширования является отображением следующего вида:

$$H : \{0,1)^{2gLen} \rightarrow \{0,1)^{mLen+eLen}.$$

Теперь можно описать криптографические примитивы, непосредственно составляющие рассматриваемую криптографическую систему. Графически процесс зашифрования представлен на рис. 3.2.

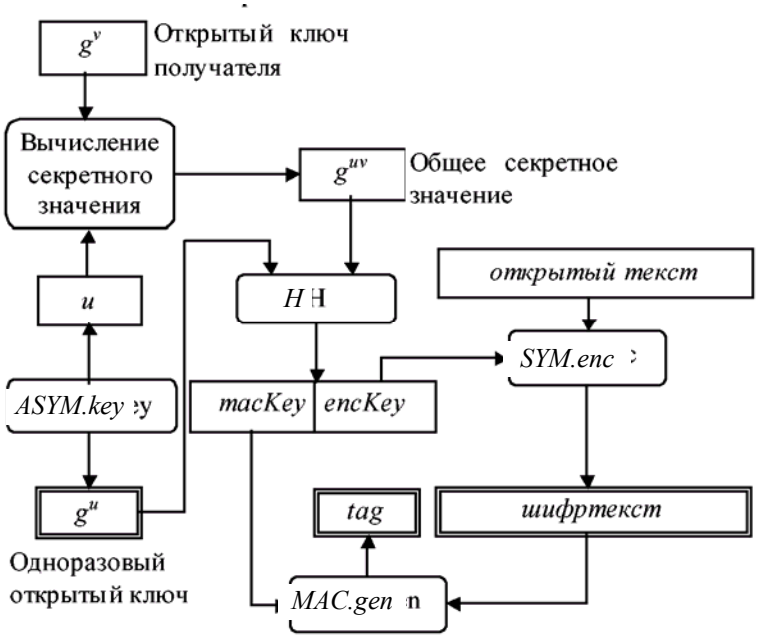


Рис. 3.2. Процесс зашифрования

Все ключевые пары в данном алгоритме выбираются так же, как и в криптосистеме Эль-Гамаля, т.е. пара (*pk*, *sK*) = (g^v, *v*) для некоторого случайного *v*. При отсылке сообщения выбирается некоторое случайное значение *u* и получателю отсылается g^u, что обеспечивает неявный обмен ключами по схеме Диффи-Хеллмана. Таким образом, зашифрованное сообщение состоит из одноразового открытого ключа, текста, зашифрованного симметричным шифром, и кода аутентификации сообщения, выработанного с помощью алгоритма *MAC.gen*.

Процесс расшифрования и аутентификации графически представлен на рис. 3.3. Элементы принятого сообщения также выделены двойной рамкой.

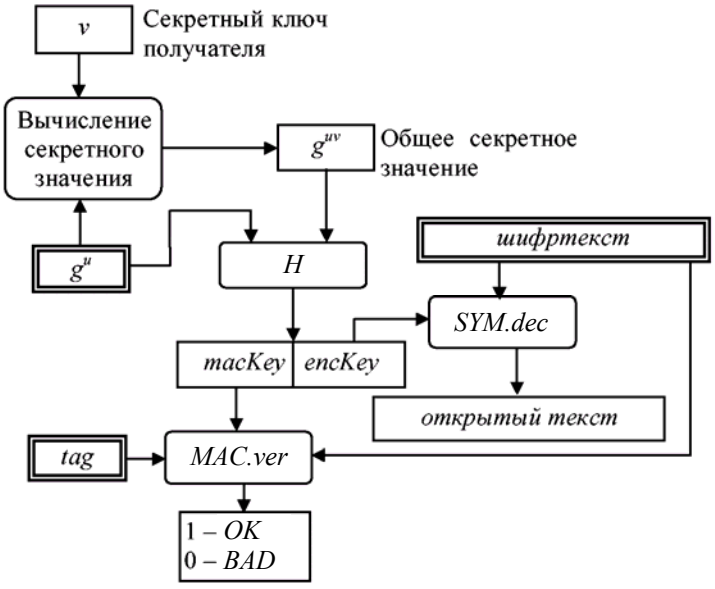


Рис. 3.3. Процесс расшифрования и аутентификации

Рассмотренная криптосистема является семантически стойкой и неделимой. В частности, неделимость обеспечивается тем, что значение g^u подается на вход функции хэширования. Если этого не сделать, то возможна атака, подобная атаке на шифр Эль-Гамаля.

Эффективность предложенной схемы по существу та же, что и у шифра Эль-Гамаля, т.е. для зашифрования требуются две операции возведения в степень, а для расшифрования – одна. Тем самым для больших сообщений скорость шифрования будет определяться скоростью работы симметричного шифра и алгоритма вычисления кода аутентификации сообщения.

3.3.3. КРИПТОСИСТЕМА РИВЕСТА–ШАМИРА–АДЛЕМАНА

В настоящее время наиболее развитым методом криптографической защиты информации с известным ключом является *RSA*, названный так по начальным буквам фамилий ее изобретателей (*Rivest*, *Shamir* и *Adleman*) и представляющую собой криптосистему, стойкость которой основана на сложности решения задачи разложения числа на простые сомножители.. Перед тем как приступить к изложению концепции метода *RSA*, необходимо определить некоторые термины.

Под *простым числом* будем понимать такое число, которое делится только на единицу и на само себя. Взаимно простыми числами будем называть такие числа, которые не имеют ни одного общего делителя, кроме единицы.

Под *результатом операции* $i \bmod j$ будем считать остаток от целочисленного деления i на j. Чтобы использовать алгоритм *RSA*, надо сначала сгенерировать открытый и секретный ключи, выполнив следующие шаги:

- выберем два очень больших простых числа p и q;
- определим n как результат умножения p на q ($n = p \times q$);
- выберем большое случайное число, которое назовем d (оно должно быть взаимно простым с m результатом умножения $(p-1) \times (q-1)$);
- определим такое число e, для которого является истинным следующее соотношение: $(e \times d) \bmod (m) = 1$ или $e = (1 \bmod (m))/d$.

Открытым ключом будут числа e и n, а секретным ключом – числа d и n.

Теперь, чтобы зашифровать данные по известному ключу $\{e, n\}$, необходимо сделать следующее:

- разбить шифруемый текст на блоки, каждый из которых может быть представлен в виде числа $M(i) = 0, 1, \ldots, n-1$;
- зашифровать текст, рассматриваемый как последовательность чисел $M(i)$ по формуле $C(i) = (M(i)^e) \bmod n$.

Чтобы расшифровать данные, используя секретный ключ $\{d, n\}$, необходимо выполнить следующие вычисления: $M(i) = (C(i)^d) \bmod n$. В результате получится множество чисел $M(i)$, которые представляют собой исходный текст.

Пример. Применим метод *RSA* для шифрования сообщения "ГАЗ". Для простоты будем использовать очень маленькие числа (на практике используются намного большие числа).

Выберем $p = 3$ и $q = 11$.

Определим $n = 3 \times 11 = 33$.

Найдем $(p-1) \times (q-1) = 20$. Следовательно, в качестве d выберем любое число, которое является взаимно простым с 20, например $d = 3$.

Выберем число e. В качестве такого числа может быть взято любое число, для которого удовлетворяется соотношение $(e \times 3) \bmod 20 = 1$, например 7.

Представим шифруемое сообщение как последовательность целых чисел в диапазоне 0…32. Пусть буква А изображается числом 1, буква Г – числом 4, а буква З – числом 9. Тогда сообщение можно представить в виде последовательности чисел 4 1 9. Зашифруем сообщение, используя ключ {7, 33}:

$C_1 = (4^7) \bmod 33 = 16384 \bmod 33 = 16,$
$C_2 = (1^7) \bmod 33 = 1 \bmod 33 = 1,$
$C_3 = (9^7) \bmod 33 = 4782969 \bmod 33 = 15.$

Шифртекст: "16 1 15"

Попытаемся расшифровать сообщение {16, 1, 15}, полученное в результате зашифрования по известному ключу, на основе секретного ключа {3, 33}:

$M_1 = (16^3) \bmod 33 = 4096 \bmod 33 = 4$,
$M_2 = (1^3) \bmod 33 = 1 \bmod 33 = 1$,
$M_3 = (15^3) \bmod 33 = 3375 \bmod 33 = 9$.

Таким образом, в результате расшифрования сообщения получено исходное сообщение "ГАЗ".

Криптостойкость алгоритма *RSA* основывается на предположении, что исключительно трудно определить секретный ключ по известному, поскольку для этого необходимо решить задачу о существовании делителей целого числа. Данная задача является *NP* – полной и, как следствие этого факта, не допускает в настоящее время эффективного (полиномиального) решения. Более того, сам вопрос существования эффективных алгоритмов решения *NP* – полных задач является до настоящего времени открытым. В связи с этим для чисел, состоящих из 200 цифр (а именно такие числа рекомендуется использовать), традиционные методы требуют выполнения огромного числа операций (около 10^{23}). Время выполнения наилучших из известных алгоритмов разложения при $n > 10^{145}$ на сегодняшний день выходит за пределы современных технологических возможностей.

Существует вариант криптосистемы *RSA*, в которой вместо функции Эйлера используется функция Кармайкла λ, где $\lambda(n)$ – наименьшее целое t, такое что для любого целого x, взаимно простого с n, выполняется $x^t = 1 \bmod n$. Если n выбирается так, как описано выше, то $\lambda(n) = НОК(p - 1, q - 1)$.

3.3.4. КРИПТОСИСТЕМЫ МЕРКЛЯ-ХЕЛЛМАНА И ХОРА-РИВЕСТА

Криптосистемы Меркля-Хеллмана и Хора-Ривеста основаны на использовании односторонней функции, известной под названием "задача укладки рюкзака".

Пусть имеется n объектов, так что можно составить n-компонентный вектор f, в котором i-й компонент f представляет собой место, занимаемое i-м объектом. Имеется рюкзак общим объемом K.

Тогда задачу укладки рюкзака можно сформулировать следующим образом: даны f и K, и требуется найти битовый вектор x, такой что $fx = K$. Доказано, что не существует эффективного алгоритма вычисления x по f и K в общем случае. Таким образом, можно использовать вектор f для шифрования n-битового сообщения x путем вычисления произведения $K = f\ x$.

Важно отметить, что выбор f является критическим. Предположим, что f выбирается в виде супервозрастающей последовательности, тогда для любого i

$$f_i > \sum_{j=1}^{i-1} f_j \ .$$

В этом случае при данных f и K вычислить x очень просто. Проверим, является ли K большим, чем последний элемент f, и если да, то делаем последний элемент x равным 1, вычитаем это значение из K и рекурсивно решаем меньшую проблему. Этот метод работает, поскольку когда K больше последнего элемента f, даже если выбрать $x = (1\ 1\ 1\ ...\ 1\ 0)$, то произведение fx все равно будет слишком маленьким, благодаря тому, что последовательность супервозрастающая. Таким образом, необходимо выбирать единицу в последней позиции x.

Ясно, что выбор f очень важен: можно получить, а можно и не получить одностороннюю функцию. Однако, именно существование этого простого случая позволяет создать функцию-ловушку, которую можно использовать для построения криптосистемы с открытым ключом.

Пользователь A получает свой открытый ключ следующим образом:

1. Выбирает супервозрастающую последовательность f', примерно, из 100 элементов.
2. Выбирает случайное целое m, большее суммы элементов f'.
3. Выбирает другое случайное целое w, взаимно простое с m.
4. Теперь вычисляется f' умножением каждого компонента f на w по модулю m; $f' = f\ w \pmod m$.
5. Проводится случайная перестановка P элементов f' для получения открытого ключа f.

Теперь A раскрывает ключ f и держит в секрете f', m, w и P.

Когда пользователь B хочет послать A сообщение (битовый вектор) x,

он вычисляет $S = fx$ и посылает это вычисленное S. Если данная система является стойкой, тогда для внешнего наблюдателя C вычисление x по S и публичному ключу f будет эквивалентно решению задачи рюкзака в общем случае. Допустим, что предположение о стойкости верно. В этом случае, хотя C не может расшифровать сообщение, A может это сделать, применяя секретные значения, которые он использовал при вычислении f.

Пользователь A может вычислить $S' = f'x$, так что она сможет решить задачу рюкзака в случае супервозрастающей последовательности. Вычисление S' производится следующим образом:

$$S' = f'x = \sum_i f_i' x_i \bmod m = w^{-1} \sum_i w f_i' x_i \bmod m =$$
$$w^{-1} \sum_i w f_i x_i \bmod m = w^{-1} S \bmod m.$$

Таким образом, A просто умножает S на мультипликативное обратное w по модулю m, а затем решает задачу рюкзака в случае супервозрастающей последовательности f', и теперь сможет прочитать сообщение.

В 1982 г. Э. Шамир открыл атаку на криптосистему, использующую одну итерацию, что привело к отказу от систем, основанных на "задаче рюкзака".

В 1986 г. Бен-Цион Хор предложил криптосистему, на сегодняшний день единственную, не использующую модульное умножение для скрытия простой задачи укладки рюкзака. Это также единственная си-стема, основанная на задаче укладки рюкзака, которая не раскрыта.

Во-первых, что любая супервозрастающая последовательность должна расти экспоненциально, поскольку минимальная супервозра-стающая последовательность – это степени двойки. Во-вторых, отметим, что причина, по которой используются супервозрастающие последовательности, заключается в том, что любая h-элементная сумма из нее уникальна. Другими словами, если представить последовательность в виде вектора f, функция скалярного произведения f на битовый вектор x будет однозначна и поэтому может быть обращена. Но оказывается возможным построить последовательность, растущую только полиномиально, но сохраняющую свойство единственности h-элементных сумм. Конструкция такой последовательности была опубликована в 1962 г.

Пусть $GF(p)$ – поле целых чисел по модулю простого числа p, и $GF(p^h)$ – расширение степени h основного поля. Также пусть 1 – вектор, все элементы которого равны единице.

Построим последовательность длины p такую, что для любого i от 0 до $p - 1$

$$1 \le a_i \le p^h - 1,$$

и для каждых различных x, y, таких, что $x \times 1 = y \times 1 = h$, $x \times a$ и $y \times a$ также должны быть различны. Можно представить векторы x и y как битовые.

Далее построение проводится довольно просто. Во-первых, выберем t – алгебраический элемент степени h над $GF(p)$, т.е. минимальный многочлен с коэффициентами из $GF(p)$, корнем которого является t, имеет степень h. Далее выберем g – мультипликативный генератор (примитивный элемент) поля $GF(p^h)$, т.е. для каждого элемента x из $GF(p^h)$ (кроме нуля) существует некоторое i, такое, что g в степени i будет равно x.

Теперь рассмотрим аддитивный сдвиг $GF(p)$, т.е. множество

$$t + GF(p) = \{t + i \,|\, 0 \le i \le p - 1\} \subset GF(p^h).$$

Пусть каждый элемент вектора a будет логарифмом по основанию g соответствующего элемента из $t + GF(p)$:

$$a_i = \log_g (t + 1).$$

При этом каждый элемент a будет лежать в заданном диапазоне, поскольку g порождает $GF(p, h)$. Теперь пусть есть различные x и y, такие что $x \times 1 = y \times 1 = h$, но $x \times a = y \times a$. Тогда, возводя g в степень $x \times a$ и $y \times a$, получим:

$$g^{\sum_{i=0}^{p-1} x_i a_i} = g^{\sum_{i=0}^{p-1} y_i a_i} .$$

Это можно записать:

$$\prod_{i=0}^{p-1} (g^{a_i})^{x_i} = \prod_{i=0}^{p-1} (g^{a_i})^{y_i} \text{ и } \prod_{i=0}^{p-1} (t+i)^{x_i} = \prod_{i=0}^{p-1} (t+i)^{y_i} .$$

Теперь заметим, что произведение в обеих частях неравенства представляет собой приведенный многочлен от t степени h. Иными словами, если вычислить оба этих произведения и заменить значение t формальным параметром z, тогда старшим членом на каждой стороне был бы x в степени h с коэффициентом 1. Зная, что если подставить значение t вместо z, то значения этих двух полиномов будут равны. Тогда вычтя один из другого, старшие члены сократятся, и если подставить i, то получим 0. При этом получим полином степени $h - 1$, корнем которого является t. Но это противоречит тому, что выбранное t является алгебраическим элементом степени h. Таким образом, доказательство закончено и построение корректно.

Хор Б.-Ц. разработал метод использования данного построения в качестве основы криптосистемы. Выбирают p и h достаточно маленькими, чтобы можно было вычислять дискретные логарифмы в $GF(p^h)$ (рекомендуется p около 200, а h около 25), далее выбирают t и g. Для каждого из них будет много вариантов, и можно просто произвести случайный выбор. (В действительности, будет так много пар $<t, g>$, что очень большое количество пользователей могут использовать одинаковые p и h, и вероятность того, что два пользователя выберут одинаковые ключи, будет пренебрежимо мала.). Затем, следуя конструкции Боуза–Чоула, вычисляют логарифмы по основанию g от $t + i$ для каждого i, что даст a. Наконец, выбирают случайную перестановку a, которая и будет ключом. При этом о результатах перестановки a вместе с p и h извещают, а величины t, g и использованная перестановка остаются в секрете.

Чтобы послать сообщение A, B просто берет свое сообщение и вычисляет $S = x \times a$. В действительности, это не так уж и просто, поскольку сообщение должно быть длиной p бит и должно быть $x \times 1 = h$, но Хор Б.-Ц. представил довольно прямолинейный метод преобразования неограниченной битовой строки в несколько блоков, каждый из которых имеет требуемую форму. A получает S. Он возводит g в степень S и выражает результат в виде полинома от t степени h с коэффициентами из $GF(p)$. Далее он вычисляет h корней этого полинома, затем применяет обратную подстановку и получает индексы элементов в x, содержащих единицы.

Интересно отметить, что если кто-либо откроет эффективный метод вычисления дискретных логарифмов, то такой алгоритм не только не поможет вскрыть эту систему, но и облегчит генерацию ключей, так как при этом необходимо вычислять дискретные логарифмы.

До настоящего времени не было опубликовано ни одного эффективного метода вскрытия этой системы при знании только открытого ключа.

3.3.5. КРИПТОСИСТЕМЫ, ОСНОВАННЫЕ НА ЭЛЛИПТИЧЕСКИХ КРИВЫХ

Рассмотренная выше криптосистема Эль-Гамаля основана на том, что проблема логарифмирования в конечном простом поле является сложной с вычислительной точки зрения. Однако, конечные поля являются не единственными алгебраическими структурами, в которых может быть поставлена задача вычисления дискретного логарифма. В 1985 г. Коблиц и Миллер независимо друг от друга предложили использовать для построения криптосистем алгебраические структуры, определенные на множестве точек на эллиптических кривых. Рассмотрим случаи определения эллиптических кривых над простыми полями Галуа произвольной характеристики и над полями Галуа характеристики 2.

Пусть $p > 3$ – простое число. Пусть $a, b \in GF(p)$ такие, что $4a^2 + 27b^2 \neq 0$. Эллиптической кривой E над полем $GF(p)$ называется множество решений (x, y) уравнения

$$y^2 = x^3 + ax + b$$

над полем $GF(p)$ вместе с дополнительной точкой ∞ называемой *точкой в бесконечности* или *нулевой точкой O* (поскольку эта точка выполняет роль нейтрального элемента в группе точек). Такое представление эллиптической кривой носит название *эллиптической кривой в форме Вейерштрасса.*

Если обозначить количество точек на эллиптической кривой E через $\#E$, то по теореме Хассе $\#E = p + 1 - t$, где $|t| \le 2\sqrt{q}$.

$\#E$ называется *порядком* кривой E, а t *– следом* кривой E.

Множество точек эллиптической кривой E с заданной бинарной операцией образует абелеву группу.

Если $\#E = p + 1$, то кривая E называется *суперсингулярной.*

Эллиптическая не являющаяся суперсингулярной кривая E над полем $GF(2^m)$ характеристики 2 задается следующим образом.

Пусть $m > 3$ – целое число. Пусть $a, b \in GF(2^m)$, $b \neq 0$. Эллиптической кривой E над полем $GF(2^m)$ называется множество решений (x, y) уравнения

$$y^2 + xy = x^3 + ax + b$$

над полем $GF(2^m)$ вместе с дополнительной точкой ∞, называемой *точкой в бесконечности.*

Количество точек на кривой E также определяется теоремой Хассе:

$$q + 1 - 2\sqrt{q} \le \#E \le q + 1 + 2\sqrt{q}\,, \quad \text{где} \quad q = 2^m, \quad \text{а} \quad \#E - \text{четно}.$$

В этом случае множество точек эллиптической кривой E с заданной бинарной операцией также образует абелеву группу.

Группа точек на кривой имеет простую структуру, а именно она является абелевой группой ранга 1 или 2, т.е. изоморфна прямому произведению двух циклических групп $Z_{n1} \times Z_{n2}$, где n_1 и n_2 – целые числа, n_2 делит n_1 и n_2 делит $q - 1$, где q – порядок поля коэффициентов, при этом n_1 может быть равно 1. Индекс подгруппы Z_n в группе точек называют *кофактором* эллиптической кривой.

Пользуясь операцией сложения точек на кривой, можно естественным образом определить операцию умножения точки $P \in E$ на произвольное целое число n: $nP = P + P + ... + P$, где операция сложения выполняется n раз.

Теперь построим одностороннюю функцию, на основе которой можно будет создать криптографическую систему.

Пусть E – эллиптическая кривая, $P \in E$ – точка на этой кривой. Выберем целое число $n < \#E$. Тогда в качестве прямой функции выберем произведение nP. Для его вычисления по оптимальному алгоритму потребуется не более $2\log_2 n$ операций сложения. Обратную задачу сформулируем следующим образом: по заданным эллиптической кривой E, точке $P \in E$ и произведению nP найти n. В настоящее время все известные алгоритмы решения этой задачи требуют экспоненциального времени.

Теперь можно описать криптографический протокол, аналогичный известному протоколу Диффи-Хеллмана. Для установления защищенной связи два пользователя A и B совместно выбирают эллиптическую кривую E и точку P на ней. Затем каждый из пользователей выбирает свое секретное целое число, соответственно a и b. Пользователь A вычисляет произведение aP, а пользователь B – bP. Далее они обмениваются вычисленными значениями. При этом параметры самой кривой, координаты точки на ней и значения произведений являются открытыми и могут передаваться по незащищенным каналам связи. Затем пользователь A умножает полученное значение на a, а пользователь B умножает полученное им значение на b. В силу свойств операции умножения на число $abP = baP$. Таким образом, оба пользователя получат общее секретное значение (координаты точки abP), которое они могут использовать для получения ключа шифрования. Отметим, что злоумышленнику для восстановления ключа потребуется решить сложную с вычислительной точки зрения задачу определения a и b по известным E, P, aP и bP.

3.4. Применение асимметричных алгоритмов

Существенным недостатком асимметричных методов является их низкое быстродействие, поэтому их приходится сочетать с симметричными (асимметричные методы на 3–4 порядка медленнее симметричных). Так, для решения задачи рассылки ключей сначала сообщение симметрично шифруют случайным ключом, затем этот ключ шифруют открытым асимметричным ключом получателя, после чего сообщение и ключ отправляются по сети.

Рис. 3.4 иллюстрирует эффективное шифрование, реализованное путем сочетания симметричного и асимметричного методов, а рис. 3.5 – расшифрование эффективно зашифрованного сообщения.

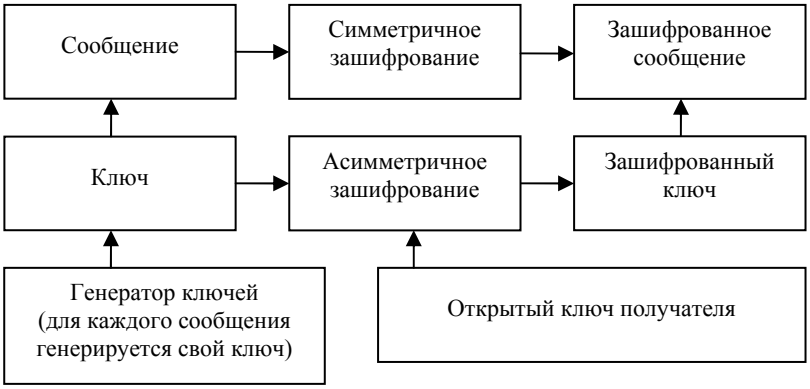


Рис. 3.4. Эффективное шифрование сообщения

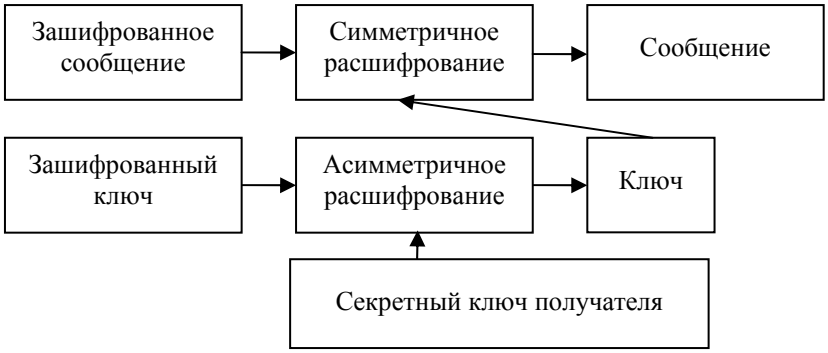


Рис. 3.5. Расшифрование эффективно зашифрованного сообщения

Применение асимметричных методов позволило решить важную задачу совместной выработки секретных ключей, обслуживающих сеанс взаимодействия, при изначальном отсутствии общих секретов. Для этого используется алгоритм Диффи-Хелмана.

Определенное распространение получила разновидность симметричного шифрования, основанная на использовании составных ключей. Идея состоит в том, что секретный ключ делится на две части, хранящиеся отдельно. Каждая часть сама по себе не позволяет выполнить расшифрование. Если у правоохранительных органов появляются подозрения относительно лица, использующего некоторый ключ, они могут в установленном порядке получить половинки ключа и дальше действовать обычным для симметричной расшифровки образом.

Составные ключи – отличный пример следования принципу разделения обязанностей. Они позволяют сочетать право граждан на тайну с возможностью эффективно следить за нарушителями закона, хотя, здесь очень много тонкостей и технического, и юридического плана.

Криптографические методы позволяют надежно контролировать целостность информации, определять ее подлинность, гарантировать невозможность отказаться от совершенных действий. В отличие от традиционных методов контрольного суммирования, способных противостоять только случайным ошибкам, криптографическая контрольная сумма (имитовставка), вычисленная с применением секретного ключа, практически исключает все возможности незаметным образом изменить данные.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Дайте определение асимметричного шифрования (с открытым ключом) и сформулируйте основные требования к нему.

2. Изложите схему организации секретной связи с использованием системы шифрования с открытым ключом.

3. Назовите и охарактеризуйте методы шифрования с открытым ключом.

4. Расскажите, каким образом можно организовать передачу цифровых сообщений, с помощью криптосистемы *RSA*. Приведите примеры.

5. Расскажите, каким образом с помощью криптосистемы *RSA* можно организовать передачу сообщений, подлинность которых подтверждена цифровой подписью. Приведите примеры.

6. Пусть $p = 383$, $q = 563$, $n = 215629$, $E = 49$. Докажите, что в этом случае открытый текст полностью восстанавливается уже через 10 шагов повторного шифрования (подсказка: достаточно показать, что $49^{10} \equiv 1(\mathrm{mod}(p-1)q-1))$.

7. Дешифруйте сообщение "17 12 26 26 24 29 25 32 19 26 29 10 05 11 07 10 05 32 29 09 05 29 05 01 05 08 32 04 26 30 26 19 17 25 32 31 00 19 19 05 03 32 13 00 31 00 23 17", зашифрованное на открытом ключе: $E = 3$, $n = 33$. Исходное сообщение состоит из цифр от 0 до 32, которыми закодированы буквы в соответствии с табл. 4.

8. Разложите на простые сомножители число 391, которое является произведением двух близких по значению простых чисел.

9. Сравните наиболее распространенные стандарты шифрования.

4. ЭЛЕКТРОННЫЕ ЦИФРОВЫЕ ПОДПИСИ

4.1. Постановка задачи

Передача сообщения отправителем (пользователь А) получателю (пользователь В) предполагает передачу данных, побуждающую пользователей к определенным действиям. Передача данных может представлять собой передачу фондов между банками, продажу акций или облигаций на автоматизированным рынке, а также передачу приказов (сигналов) по каналам электросвязи. Участники нуждаются в защите от множества злонамеренных действий, к которым относятся:

- отказ (ренегатство) – отправитель впоследствии отказывается от переданного сообщения;
- фальсификация – получатель подделывает сообщение;
- изменение – получатель вносит изменения в сообщение;
- маскировка – нарушитель маскируется под другого пользователя.

Для верификации (подтверждения) сообщения M (пользователь A – пользователю B) необходимо следующее:

1. Отправитель (пользователь A) должен внести в M подпись, содержащую дополнительную информацию, зависящую от M и, в общем случае, от получателя сообщения и известной только отправителю закрытой информации kA.

2. Необходимо, чтобы правильную подпись M: $SIG\{kA, M,$ идентификатор $B\}$ в сообщении для пользователя B нельзя было составить без kA.

3. Для предупреждения повторного использования устаревших сообщений процедура составления подписи должны зависеть от времени.

4. Пользователь B должен иметь возможность удостовериться, что $SIG\{kA, M,$ идентификатор $B\}$ – есть правильная подпись M пользователем A.

Рассмотри эти пункты подробнее.

1. *Подпись сообщения* – определенный способ шифрования M путем криптографического преобразования. Закрываемым элементом k_A в преобразовании <Идентификатор B, $M > \longrightarrow SIG\{k_A, M,$ идентификатор $B\}$ является ключ криптопреобразования.

Цифровая сигнатура – это строка символов, зависящая как от идентификатора отправителя, так и содержания сообщения.

Во всех практических криптографических системах k_A принадлежит конечному множеству ключей K. Исчерпывающая проверка всех ключей, задаваемых соответствующими парами $< M$, идентификатор $B > \leftrightarrow SIG\{k_A,$

M, идентификатор $B\}$ в общем должна привести к определению ключа k_A злоумышленником. Если множество K достаточно велико и ключ k определен методом случайного выбора, то полная проверка ключей невозможна. Говоря, что составить правильную подпись без ключа невозможно, имеется в виду, что определение $SIG\{k_A, M$, идентификатор $B\}$ без k_A с вычислительной точки зрения эквивалентно поиску ключа.

2. Доступ к аппаратуре, программам и файлам системы обработки информации обычно контролируется паролями. Подпись – это вид пароля, зависящий от отправителя, получателя информации и содержания передаваемого сообщения.

3. Подпись должна меняться от сообщения к сообщению для предупреждения ее повторного использования с целью проверки нового сообщения. Цифровая подпись отличается от рукописной, которая обычно не зависит от времени составления и данных. Цифровая и рукописная подписи идентичны в том смысле, что они характерны только для данного владельца

4. Хотя получатель информации не может составить правильную подпись, он должен уметь удостоверять ее подлинность. При коммерческих сделках, например, продажа недвижимой собственности, эту функцию зачастую выполняет третье, независимое доверенное лицо (нотариус).

Установление подлинности подписи – это процесс, посредством которого каждая сторона устанавливает подлинность другой. Обязательным условием этого процесса является сохранение тайны. Во многих случаях приходится удостоверять свою личность, например, подписью при получении денег по чеку либо фотографией в паспорте при пересечении границы. Для того чтобы в системе обработки данных получатель мог установить подлинность отправителя, необходимо выполнение следующих условий:

1. Отправитель (пользователь A) должен обеспечить получателя (пользователя B) удостоверяющей информацией $AUTH\{k_A, M$, идентификатор $B\}$, зависящей от секретной информации k_A, известной только пользователю A.

2. Необходимо, чтобы удостоверяющую информацию $AUTH\{k_A$, идентификатор $B\}$ от пользователя A пользователю B можно было дать только при наличии ключа k_A.

3. Пользователь B должен располагать процедурой проверки того, что $AUTH\{k_A$, идентификатор $B\}$ действительно подтверждает личность пользователя A.

4. Для предупреждения использования предыдущей проверенной на достоверность информации процесс установления подлинности должен иметь некоторую зависимость от времени.

В основе криптографического контроля целостности лежат два понятия: *хэш-функция*, *электронная цифровая подпись* (ЭЦП).

Хэш-функция – это труднообратимое преобразование данных (одностороняя функция), реализуемое, как правило, средствами симметричного шифрования со связыванием блоков. Результат шифрования последнего блока (зависящий от всех предыдущих) и служит результатом хэш-функции.

Пусть имеются данные, целостность которых должна быть проверена, хэш-функция и ранее вычисленный результат ее применения к исходным данным (дайджест). Хэш-функцию обозначим через h, исходные данные – через T, проверяемые данные – через T'. Контроль целостности данных сводится к проверке равенства $h(T) = h(T')$. Если оно выполняется, считается, что $T = T'$. Совпадение дайджестов для различных данных называется коллизией. В принципе коллизии возможны (так как мощность множества дайджестов меньше множества хэшируемых данных), однако, исходя из определения хэш-функции, специально организовать коллизию за приемлемое время невозможно.

Асимметричные методы позволяют реализовать так называемую электронную цифровую подпись, или электронное заверение сообщения. Идея состоит в том, что отправитель посылает два экземпляра сообщения – открытое и дешифрованное его секретным ключом (естественно, дешифровка незашифрованного сообщения на самом деле есть форма шифрования). Получатель может зашифровать с помощью открытого ключа отправителя дешифрованный экземпляр и сравнить с открытым. Если они совпадут, личность и подпись отправителя можно считать установленными.

Пусть $E(T)$ обозначает результат шифрования текста T с помощью открытого ключа, а $D(T)$ – результат дешифровки текста T с помощью секретного ключа. Чтобы асимметричный метод мог применяться для реализации электронной подписи, необходимо выполнение тождества

$$E(D(T)) = D(E(T)) = T.$$

Проиллюстрируем (рис. 4.1) процедуру эффективной генерации электронной подписи, состоящую в шифровании преобразованием D дайджеста $h(T)$, а проверка эффективно сгенерированной электронной подписи может быть реализована способом, изображенным на рис. 4.2.

Из равенства $E(S') = h(T)$ следует $S' = D(h(T'))$, т.о. ЭЦП защищает целостность сообщения, удостоверяет личность отправителя и служит основой неотказуемости.

Два российских стандарта − "Процедуры выработки и проверки электронной цифровой подписи на базе асимметричного криптографического алгоритма" и "Функция хэширования", объединенные общим заголовком "Информационная технология. Криптографическая защита информации", регламентируют вычисление дайджеста и реализацию электронной подписи.

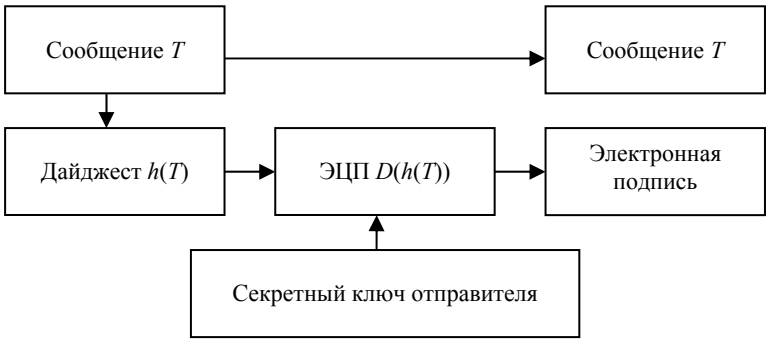


Рис. 4.1. Выработка электронной цифровой подписи

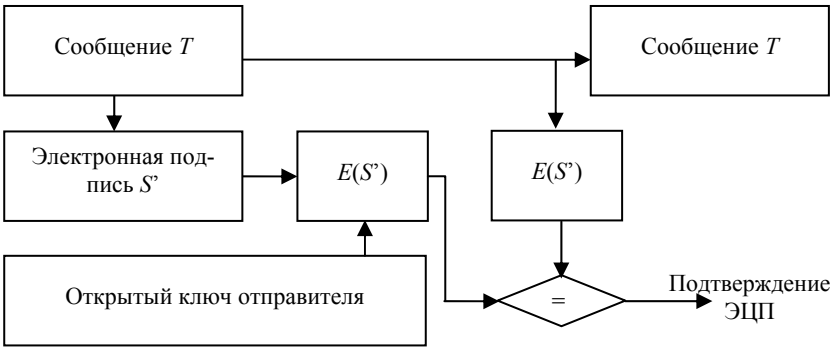


Рис. 4.2. Проверка электронной цифровой подписи

В сентябре 2001 г. утвержден, а с 1 июля 2002 г. вступил в силу новый стандарт ЭЦП – ГОСТ Р 34.10–2001.

Для контроля целостности последовательности сообщений (т.е. защиты от кражи, дублирования и переупорядочения сообщений) применяют временные штампы и нумерацию элементов последовательности, при этом штампы и номера включают в подписываемый текст.

Обратим внимание на то, что при использовании асимметричных методов шифрования (в частности, ЭЦП) необходимо иметь гарантию подлинности пары (имя, открытый ключ) адресата. Для решения этой задачи в спецификациях X.509 вводятся понятия цифрового сертификата и сертификационного центра. Сертификационный центр – это компонент глобальной службы каталогов, отвечающий за управление криптографическими ключами пользователей, заверяющий подлинность пары "имя–открытый ключ адресата" своей подписью.

Цифровые сертификаты в формате X.509 стали не только формальным, но и фактическим стандартом, поддерживаемым многочисленными сертификационными центрами.

Отметим, что услуги, характерные для асимметричного шифрования,

можно реализовать и с помощью симметричных методов, если имеется надежная третья сторона, знающая секретные ключи своих клиентов. Эта идея положена, например, в основу сервера аутентификации Kerberos.

4.2. Алгоритмы электронной цифровой подписи

4.2.1. ЦИФРОВЫЕ ПОДПИСИ, ОСНОВАННЫЕ НА АСИММЕТРИЧНЫХ КРИПТОСИСТЕМАХ

Для формирования системы ЭЦП можно использовать криптографическую систему Ривеста-Шамира-Адлемана.

Пользователь *A* вырабатывает цифровую подпись предназначенного для пользователя *B* сообщения *M* с помощью следующего преобразования:

$$SIG(M) = E_{e_B,n_B}(E_{d_A,n_A}(M)).$$

При этом он использует: свое секретное преобразование E_{d_A,n_A}; открытое преобразование E_{e_B,n_B} пользователя *B*.

Затем он передает пользователю *B* пару $\langle M, SIG(M)\rangle$.

Пользователь *B* может верифицировать это подписанное сообщение сначала при помощи своего секретного преобразования E_{d_B,n_B} с целью получения

$$E_{d_A,n_A}(M) = E_{d_B,n_B}(SIG(M)) = E_{d_B,n_B}(E_{e_B,n_B}(E_{d_A,n_A}(M)))$$

и затем открытого E_{e_A,n_A} пользователя *A* для получения сообщения *M*:

$$M = E_{e_A,n_A}(E_{d_A,n_A}(M)).$$

Затем пользователь *B* производит сравнение полученного сообщения *M* с тем, которое он получил в результате проверки цифровой подписи, и принимает решение о подлинности/подложности полученного сообщения.

В рассмотренном примере проверить подлинность ЭЦП может только пользователь *B*. Если же требуется обеспечение возможности верификации ЭЦП произвольным пользователем (например, при циркулярной рассылке документа), то алгоритм выработки ЭЦП упрощается, и подпись вырабатывается по формуле

$$SIG(M) = E_{d_A,n_A}(M),$$

а пользователи осуществляют верификацию с использованием открытого преобразования отправителя (пользователя *A*):

$$M = E_{e_A,n_A}(SIG(M)) = E_{e_A,n_A}(E_{d_A,n_A}(M)).$$

Вместо криптосистемы *RSA* для подписи сообщений можно использовать и любую другую асимметричную криптосистему.

Недостатком подобного подхода является то, что производительность асимметричной криптосистемы может оказаться недостаточной для удовлетворения предъявляемым требованиям.

Возможным решением является применение специальной эффективно вычисляемой функции, называемой хэш-функцией или функцией хэширования. Входом этой функции является сообщение, а выходом – слово фиксированной длины, много меньшей, чем длина исходного сообщения. ЭЦП вырабатывается по той же схеме, но при этом используется не само сообщение, а значение хэш-функции от него.

Очень часто бывает желательно, чтобы электронная цифровая подпись была разной, даже если дважды подписывается одно и то же сообщение. Для этого в процесс выработки ЭЦП необходимо внести элемент "случайности". Конкретный способ был предложен Эль-Гамалем аналогично тому, как это делается в системе шифрования, носящей его имя.

Выбирается большое простое число p и целое число g, являющееся примитивным элементом в Z_p. Эти числа публикуются. Затем выбирается секретное число x и вычисляется открытый ключ для проверки подписи $y = g^x \pmod p$.

Далее для подписи сообщения *M* вычисляется его хэш-функция $m = h(M)$. Выбирается случайное целое k: $1 < k < (p - 1)$, взаимно простое с p –

1, и вычисляется $r = g^k \pmod p$. После этого с помощью расширенного алгоритма Евклида решается относительно s уравнение $m = xr + ks \pmod{(p-1)}$. Подпись образует пара чисел (r, s). После выработки подписи значение k уничтожается.

Получатель подписанного сообщения вычисляет хэш-функцию сообщения $m = h(M)$ и проверяет выполнение равенства $y^r r^s = g^{xr} g^{ks} = g^{xr+ks} = g^m \pmod p$. Корректность этого уравнения очевидна.

Еще одна подобная схема была предложена Шнорром. Как обычно, p – большое простое число; q – простой делитель $(p - 1)$; g – элемент порядка q в Z_p; k – случайное число, x и $y = g^x \pmod p$ – секретный и открытый ключи соответственно. Уравнения выработки подписи выглядят следующим образом:

$$r = g^k \pmod p;\ e = h(m, r); s = k + xe \pmod q.$$

Подписью является пара (r, s). На приемной стороне вычисляется значение хэш-функции $e = h(m, r)$ и проверяется выполнение равенства $r = g^s y^{-e} \pmod p$, при этом действия с показателями степени производятся по модулю q.

Другой вариант подписи Шнорра выглядит так. Для подписи сообщения m автор выбирает случайное $k \in Z_a$, вычисляет $g^k \pmod p$, $e = h(g^k, m)$ и $z = k + xe \pmod q$. Подписью является тройка (m, e, z). Проверка подписи заключается в проверке равенства $h(g^z y^{-e}, m) = e$.

4.2.2. СТАНДАРТ ЦИФРОВОЙ ПОДПИСИ *DSS*

Новая редакция стандарта на выработку и верификацию цифровой подписи *DSS* (*Digital Signature Standard*) принята в США 7 января 2000 г. (*FIPS PUB* 186-2). Согласно этому стандарту, электронная цифровая подпись может вырабатываться по одному из трех алгоритмов: *DSA* (*Digital Signature Algorithm*), основанному на проблеме логарифма в конечном поле, *ANSI X*9.31 (*RSA DSA*) или *ANSI X*9.63 (*EC DSA* – алгоритм выработки подписи, основанной на проблеме логарифма в группе точек эллиптической кривой над конечным полем).

Опишем алгоритм *DSA*.

1. Предварительный этап – выбор параметров.

Выбираются числа p, q и g, такие, что p – простое число, $2^{l-1} < p < 2^l$, где l кратно 64 и $512 < l < 1024$; q – простой делитель числа $p - 1$ длиной 160 бит ($2^{159} < q < 2^{160}$); g – элемент порядка q в Z_p. g выбирается в виде $g = h^{(p-1)/q}$, где $1 < h < p - 1$ и $h^{(p-1)/q} > 1$. Эти три числа являются открытыми данными и могут быть общими для группы пользователей.

Выбирается секретный ключ x, $0 < x < q$, и вычисляется открытый ключ для проверки подписи $y = g^x \pmod p$.

2. Выработка электронной цифровой подписи.

Вычисляется значение хэш-функции от сообщения $h(/w)$. При этом используется алгоритм безопасного хэширования *SHA*-1 (*Secure Hashing Algorithm*), на который ссылается стандарт (*FIPS PUB* 180-1). Значение хэш-функции $h(/w)$ имеет длину 160 бит.

Далее подписывающий выбирает случайное или псевдослучайное значение k, $0 < k < q$, вычисляет $k^{-1} \bmod q$, и вырабатывает пару значений:

$$r = g^k \pmod p \pmod q;\ s = k^{-1}(h(m) + xr) \pmod q.$$

Эта пара значений (r, s) и является электронной подписью под сообщением M. После выработки цифровой подписи значение k уничтожается.

3. Верификация электронной цифровой подписи.

Пусть было принято сообщение m_1. Тогда уравнение проверки выглядит следующим образом:

$$r \equiv g^{h(m_1)s^{-1}} y^{rs^{-1}} \pmod p \pmod q.$$

Алгоритм выработки ЭЦП, основанный на эллиптических кривых, может быть описан следующим образом:

1. *Выбор параметров*. Стандарт определяет поля, над которыми задаются эллиптические кривые. Это простые поля Галуа и поля Галуа харак-

теристики 2. Выбор полей в стандарте сделан, исходя из требования повышения вычислительной эффективности машинных операций умножения в поле. Для этого в качестве простых модулей выбраны так называемые обобщенные числа Мерсенна (табл. 4.1).

4.1. Обобщенные числа Марсенна для эллиптических кривых

Кривая	*P*
P-192	$2^{192} - 2^{64} - 1$
P-224	$2^{224} - 2^{96} + 1$
P-256	$2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$
P-384	$2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$
P-521	$2^{521} - 1$

Стандарт фиксирует кривые, которые должны использоваться в алгоритмах, и примерные базовые точки на этих кривых. Пользователь может либо воспользоваться приведенными в стандарте базовыми точками, либо сгенерировать свои, если, например, ему понадобится обеспечить криптографическое разделение сетей ЭВМ. В частности, кривые P-192, ..., P-521 представляют собой эллиптические кривые простого порядка r вида $y^2 = x^3 - 3x + b$ над полем $GF(p)$.

Для задания кривых над полями характеристики 2 выбраны порождающие многочлены. При этом возможно использовать представление полей как в полиномиальном, так и в нормальном базисах (табл. 4.2).

4.2. Порождающие многочлены

Кривая	Порождающий многочлен *p(t)*	Тип нормального базиса
K-163, B-163	$t^{163} + t^7 + t^6 + t^3 + 1$	4
K-233, B-233	$t^{233} + t^{74} + 1$	2
K-283, B-283	$t^{283} + t^{12} + t^7 + t^5 + 1$	6
K-409, B-409	$t^{409} + t^{87} + 1$	4
K-571, B-571	$t^{571} + t^{10} + t^5 + t^2 + 1$	10

Коэффициенты b заданы для каждого размера поля. Например, кривая P–521 задается коэффициентом b = 051 953eb961 8elc9alf 929a21a0 b68540ee a2da725b 99b315f3 b8b48991 8ef109e1 56193951 ec7e937b 1652c0bd 3bblbf07 3573df88 3d2c34fl ef451fd4 6b503f00 (в шестнадцатеричном виде) и имеет порядок r = 68647976601306097149819007990813932172694353001433054093944634591855431833976553942450577463332171975329639963713633211138647686124403803403728088927070054 49 (в десятичной записи).

Поскольку определенные в стандарте кривые имеют простой порядок, группы точек на них являются циклическими и порядок базовой точки n в точности равен r.

Для каждого из полей $GF(2^m)$ в стандарте указаны по две кривых: псевдослучайная вида $y^2 + xy = x^3 + x^2 + b$, и специальная кривая Коблица, или аномальная двоичная кривая вида $y^2 + xy = x^3 + ax^2 + 1$, где $a = 0$ или 1.

При генерации ключевой пары пользователь выбирает в качестве секретного ключа целое число d, $0 < d < n$, где n – порядок базовой точки G на эллиптической кривой. Далее он вычисляет $Q = dG$ и публикует Q в качестве открытого ключа.

2. *Выработка ЭЦП.* Для того, чтобы подписать сообщение m, пользователь:

1) Выбирает случайное число k, $0 < k < n - 1$.

2) Вычисляет $kG = (x_1, y_1)$, $r = x_1 (\mathrm{mod}\ n)$. Если $r = 0$, то перейти к шагу 1.

3) Вычисляет $k^{-1} \pmod n$.

4) Вычисляет $e = SHA(m)$. Выбор варианта функции хэширования осуществляется в зависимости от используемого поля.

5) Вычисляет $s = k^{-1}(e + dr) \bmod n$. Если $s = 0$, то перейти к шагу 1.

6) Подписью сообщения m является пара (r, s).

3. *Верификация ЭЦП.*

Для проверки подписи получатель сообщения выполняет следующие действия:

1) Проверяет, что r и s лежат на интервале $(0, n)$.

2) Вычисляет $e = SHA(m)$.

3) Вычисляет $w = s^{-1} \bmod n$.

4) Вычисляет $u_1 = ew \bmod n$ и $w_2 = rw \bmod n$.

5) Вычисляет $X = u_1G + u_2Q$. Если $X = \infty$, то подпись отвергается, иначе, вычислить $v = x_1 \bmod n$, где $X = (x_1, y_1)$.

6) Принять подпись, если и только если v = г.

4.2.3. СТАНДАРТ ЦИФРОВОЙ ПОДПИСИ ГОСТ Р 34.10–94

Российский стандарт ЭЦП разрабатывался позже первоначального варианта американского, поэтому параметры этого алгоритма выбраны с учетом возросших возможностей потенциального противника по вскрытию криптосистем. В частности, увеличена длина значения хэш-функции, что снижает вероятность столкновений, и, соответственно, порядок элемента-генератора, что делает более сложным решение задачи дискретного логарифма для восстановления секретного ключа. При описании алгоритма будут использоваться следующие обозначения:

B^* – множество всех конечных слов в алфавите $B = \{0,1\}$;

$|A|$ – длина слова A;

$V_k(2)$ – множество всех двоичных слов длины k;

$A||B$ – конкатенация слов A и B, также обозначается как AB;

A^k – конкатенация k экземпляров слова A;

$<N>k$ – слово длины k, содержащее запись $N \pmod{2^k}$, где N – неотрицательное целое;

$\oplus$ – побитовое сложение слов по модулю 2;

[+] – сложение по правилу A [+] $B = <A + B>_k$ $(k = |A| = |B|)$;

m – передаваемое сообщение;

m_1 – полученное сообщение;

h – хэш-функция, отображающая последовательность m в слово $h(m) \in V_{256}(2)$;

p – простое число, $2^{509} < p < 2^{512}$, либо $2^{1020} < p < 2^{1024}$;

q – простое число, $2^{254} < q < 2^{256}$ и q является делителем для $(p – 1)$;

a – целое число, $1 < a < p – 1$, при этом $a^q \pmod p = 1$;

k – целое число, $0 < k < q$;

x – секретный ключ пользователя для формирования подписи, $0 < x < q$;

y – открытый ключ для проверки подписи $y = a^x \pmod p$.

Система ЭЦП включает в себя процедуры выработки и проверки подписи под данным сообщением.

Цифровая подпись, состоящая из двух целых чисел, вычисляется с помощью определенного набора правил, изложенных в стандарте.

Числа p, q и a, являющиеся параметрами системы, не являются секретными. Конкретный набор их значений может быть общим для группы пользователей. Целое число k, которое генерируется в процедуре подписи сообщения, должно быть секретным и должно быть уничтожено сразу после выработки подписи. Число k снимается с физического датчика случайных чисел или вырабатывается псевдослучайным методом с использованием секретных параметров.

Процедура выработки подписи включает в себя следующие шаги:

1. Вычислить $h(m)$ – значение хэш-функции h от сообщения m. Если $h(m) \pmod q = 0$, то присвоить $h(m)$ значение $0^{255}1$.

2. Выработать целое число k, $0 < k < q$.

3. Вычислить два значения: $r' = d^k \pmod p$ и $r = r' \pmod q$. Если $r = 0$, то перейти к шагу 2 и выработать другое значение числа k.

4. С использованием секретного ключа x пользователя вычислить

значение $s = (xr + kh(m))(\mathrm{mod}\, q)$. Если $s = 0$, то перейти к шагу 2, в противном случае закончить работу алгоритма.

Сообщение, дающее нулевое значение хэш-функции, не подписывается. В противном случае уравнение подписи упростилось бы до $s = xr \pmod q$ и злоумышленник легко мог бы вычислить секретный ключ x.

Проверка цифровой подписи возможна при наличии у получателя открытого ключа отправителя, пославшего сообщение.

Уравнение проверки будет следующим:

$$r \equiv (a^{sh(m_1)^{-1}} y^{-rh(m_1)^{-1}} (\mathrm{mod}\, p))(\mathrm{mod}\, q).$$

Вычисления по этому уравнению реализуются следующим образом:

1. Проверить условия: $0 < s < q$ и $0 < r < q$. Если хотя бы одно из этих условий не выполнено, то подпись считается недействительной.
2. Вычислить $h(m_1)$ – значение хэш-функции h от полученного сообщения m_1. Если $h(m_1)(\mathrm{mod}\, q) = 0$, присвоить $h(m_1)$ значение $0^{255}1$.
3. Вычислить значение $v = (h(m_1)))^{q-2} (\mathrm{mod}\, q)$, что является ни чем иным, как мультипликативным обратным к $h(m_1)(\mathrm{mod}\, q)$. Вообще говоря, алгоритм проверки можно несколько ускорить, если вычислять $h(m_1)^{-1} (\mathrm{mod}\, q)$ с помощью расширенного алгоритма Евклида, а ни путем возведения в степень.
4. Вычислить значения: $z_1 = sv(\mathrm{mod}\, q)$ и $z_2 = (q - r)v(\mathrm{mod}\, q)$.
5. Вычислить значение $u = (a^{z_1} y^{z_2} (\mathrm{mod}\, p))(\mathrm{mod}\, q)$.
6. Проверить условие $r = u$.

При совпадении значений r и u получатель принимает решение о том, что полученное сообщение подписано данным отправителем и в процессе передачи не нарушена целостность сообщения, т.е. $m_1 = m$. В противном случае подпись считается недействительной.

4.2.4. СТАНДАРТ ЦИФРОВОЙ ПОДПИСИ ГОСТ Р 34.10–2001

С повышением производительности вычислительных средств и усовершенствованием алгоритмов вычисления логарифмов в конечном поле появилась потребность в повышении стойкости ЭЦП к различного рода атакам. В связи с этим был разработан стандарт ГОСТ Р 34.10–2001 "Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи", введенный с 1 июля 2002 г. вместо ранее действовавшего стандарта ГОСТ Р 34.10–94.

В данном стандарте используются операции группы точек эллиптической кривой над конечным полем. Применяется эллиптическая кривая E в форме Вейерштрасса над простым полем, задаваемая коэффициентами a и b или величиной $J(E)$, называемой инвариантом эллиптической кривой:

$$J(E) \equiv 1728 \frac{4a^3}{4a^3 + 27b^2} (\mathrm{mod}\, p).$$

Коэффициенты a и b кривой E определяются по известному инварианту следующим образом:

$$\begin{cases} a \equiv 3k (\mathrm{mod}\, p), \\ b \equiv 2k (\mathrm{mod}\, p \text{ где } k \equiv \dfrac{J(E)}{1728 - J(E)} (\mathrm{mod}\, p),\ J(E) \neq 0 \text{ или } 1728. \end{cases}$$

Точку Q будем называть точкой кратности k, $k \in Z$, если для некоторой точки P выполнено равенство $Q = kP$.

Параметрами схемы ЭЦП являются следующие значения: p – модуль эллиптической кривой, простое число; $p > 2^{255}$; эллиптическая кривая, задаваемая инвариантом $J(E)$ или коэффициентами a и b; целое число m – порядок группы точек эллиптической кривой E; простое число q – порядок циклической подгруппы группы точек эллиптической кривой E, для которого выполнены следующие условия:

$$m = nq, n \in Z, n \geq 1, 2^{254} \leq q \leq 2^{256}.$$

Базовая точка $P \neq 0$ на кривой, имеющая порядок q, т.е. удовлетворяющая равенству $qP = 0$. Координаты этой точки обозначим через (x_p, y_p);

хэш-функция, отображающая сообщения произвольной длины в множество двоичных векторов длины 256. Хэш-функция определена стандартом ГОСТ Р 34.11–94.

Каждый пользователь схемы ЭЦП должен обладать личной ключевой парой:

- секретный ключ пользователя – целое число d, $0 < d < q$;
- открытый ключ пользователя – точка Q с координатами (x_q, y_q), удовлетворяющая равенству $dP = Q$.

Параметры ЭЦП должны удовлетворять следующим условиям:

$p^t \neq 1 (\mathrm{mod}\, q)$, для всех целых $t = 1, 2, ..., B$, где B удовлетворяет неравенству $B \geq 31$; $m \neq p$; $J(E) \neq 0$ или 1728.

Двоичному вектору $\bar{h} = (\alpha_{255}, ..., \alpha_0)$ ставится в соответствие число $\alpha = \sum_{i=0}^{255} \alpha_i 2^t$.

ЭЦП под сообщением M вырабатывается по следующему алгоритму:

1. Вычислить хэш-функцию сообщения M: $\bar{h} = h(M)$.

2. Вычислить целое число α, двоичным представлением которого является вектор h, и определить $e \equiv \alpha (\mathrm{mod}\, q)$. Если $e = 0$, то задать $e = 1$.

3. Сгенерировать (псевдо)случайное целое число k, удовлетворяющее неравенствам $0 < k < q$;

4. Вычислить точку эллиптической кривой $C = kP$ и положить $r \equiv x_c (\mathrm{mod}\, q)$, где x_c – x – координата точки C. Если $r = 0$, то вернуться на шаг 3.

5. Вычислить значение $s \equiv (rd + ke) (\mathrm{mod}\, q)$. Если $s = 0$, то вернуться на шаг 3.

6. Вычислить двоичные векторы, соответствующие числам r и s. Определить цифровую подпись $\zeta = (\bar{r} \| \bar{s})$ как конкатенацию двух двоичных векторов.

Для проверки подписи ζ под полученным сообщением M необходимо выполнить следующие действия, используя ключ проверки подписи Q.

1. По полученной подписи ζ вычислить целые числа r и s. Если выполнены неравенства $0 < r < q$, $0 < s < q$, то перейти к следующему шагу. В противном случае подпись неверна.

2. Вычислить хэш-функцию полученного сообщения M:

$$\bar{h} = h(M).$$

3. Вычислить целое число α, двоичным представлением которого является вектор h и определить $e \equiv \alpha (\mathrm{mod}\, q)$. Если $e = 0$, то задать $e = 1$.

4. Вычислить значение $v \equiv e^{-1} (\mathrm{mod}\, q)$.

5. Вычислить значения $z_1 \equiv sv (\mathrm{mod}\, q)$, $z_2 \equiv -rv (\mathrm{mod}\, q)$.

6. Вычислить точку эллиптической кривой $C = z_1P + z_2Q$ и определить $R \equiv x_c (\mathrm{mod}\, q)$, где x_c – x-координата точки C.

7. Если выполнено равенство $R = r$, то подпись принимается, в противном случае подпись неверна.

4.2.5. ЦИФРОВЫЕ ПОДПИСИ, ОСНОВАННЫЕ НА СИММЕТРИЧНЫХ КРИПТОСИСТЕМАХ

Общеизвестно, что так называемая "современная", она же двухключевая криптография возникла и стала быстро развиваться в последние десятилетия именно потому, что ряд новых криптографических протоколов типа протокола цифровой подписи не удалось эффективно реализовать на базе традиционных криптографических алгоритмов, широко известных и хорошо изученных к тому времени. Тем не менее, это возможно. И первыми, кто обратил на это внимание, были родоначальники криптографии с открытым ключом У. Диффи и М. Хеллман, опубликовавшие описание подхода, позволяющего выполнять процедуру цифровой подписи одного бита с помощью блочного шифра. Прежде чем изложить эту идею примем во внимание несколько замечаний о сути и реализациях цифровой подписи.

Стойкость какой-либо схемы подписи доказывается обычно установлением равносильности соответствующей задачи вскрытия схемы какой-

либо другой, о которой известно, что она вычислительно неразрешима. Практически все современные алгоритмы ЭЦП основаны на так называемых "сложных математических задачах" типа факторизации больших чисел или логарифмирования в дискретных полях. Однако, доказательство невозможности эффективного вычислительного решения этих задач отсутствует, и нет никаких гарантий, что они не будут решены в ближайшем будущем, а соответствующие схемы взломаны – как это произошло с "ранцевой" схемой цифровой подписи. Более того, с бурным прогрессом средств вычислительной техники "границы надежности" методов отодвигаются в область все больших размеров блока. Всего пару десятилетий назад, на заре криптографии, с открытым ключом считалось, что для реализации схемы подписи *RSA* достаточно даже 128-битовых чисел.

Сейчас эта граница отодвинута до 1024-битовых чисел и это далеко еще не предел. Все это приводит к необходимости переписывать реализующие схему программы, и зачастую перепроектировать аппаратуру. Ничего подобного не наблюдается в области классических блочных шифров, если не считать изначально ущербного и непонятного решения комитета по стандартам США ограничить размер ключа алгоритма *DES* 56-ю битами, тогда как еще во время обсуждения алгоритма предлагалось использовать ключ большего размера. Схемы подписи, основанные на классических блочных шифрах, свободны от указанных недостатков:

- во-первых, их стойкость к попыткам взлома вытекает из стойкости использованного блочного шифра, поскольку классические методы шифрования изучены гораздо больше, а их надежность обоснована намного лучше, чем надежность асимметричных криптографических систем;
- во-вторых, даже если стойкость использованного в схеме подписи шифра окажется недостаточной в свете прогресса вычислительной техники, его легко можно будет заменить на другой, более устойчивый, с тем же размером блока данных и ключа, без необходимости менять основные характеристики всей схемы – это потребует только минимальной модификации программного обеспечения.

Итак, вернемся к схеме Диффи и Хеллмана подписи одного бита сообщения с помощью алгоритма, базирующегося на любом классическом блочном шифре. Предположим, что есть алгоритм зашифрования E_k, оперирующий блоками данных X размера n и использующий ключ размером $n_к$: $|X| = n, |K| = n_k$. Структура ключевой информации в схеме следующая: секретный ключ подписи k_s выбирается как произвольная (случайная) пара ключей k_0, k_1 используемого блочного шифра: $k_s = (k_0, k_1)$. Таким образом, размер ключа подписи равен удвоенному размеру ключа использованного блочного шифра: $|k_s| = 2|K| = 2n_K$.

Ключ проверки представляет собой результат шифрования двух блоков текста X_0 и X_1 с ключами k_0 и k_1 соответственно:

$$k_V = (C_0, C_1) = (E_{k_0}(X_0), E_{k_1}(X_1)),$$

где являющиеся параметром схемы блоки данных несекретны и известны проверяющей подпись стороне. Таким образом, размер ключа проверки подписи равен удвоенному размеру блока использованного блочного шифра: $|k_s| = 2|K| = 2n_K$.

Алгоритм *Sig* выработки цифровой подписи для бита t ($t \in \{0, 1\}$) заключается просто в выборе соответствующей половины из пары, составляющей секретный ключ подписи: $s = S(t) = k_t$.

Алгоритм *Ver* проверки подписи состоит в проверке уравнения $E_{kt}(Xf) = Q$, которое, очевидно, должно выполняться для нашего t. Получателю известны все используемые при этом величины.

Таким образом, функция проверки подписи будет следующей:

$$Ver(t, s, k_V) = \begin{cases} 1, & E_s(X_t) = C_t; \\ 0, & E_s(X_t) \neq C_t. \end{cases}$$

Предложенная Диффи и Хеллманом схема цифровой подписи на основе классического блочного шифра обладает такой же стойкостью, что и

лежащий в ее основе блочный шифр, и при этом весьма проста. Однако, у нее есть два существенных недостатка.

Первый недостаток заключается в том, что данная схема позволяет подписать лишь один бит информации. В блоке большего размера придется отдельно подписывать каждый бит, поэтому даже с учетом хэширования сообщения все компоненты подписи – секретный ключ, проверочная комбинация и собственно подпись получаются довольно большими по размеру и более чем на два порядка превосходят размер подписываемого блока.

Второй недостаток данной схемы менее заметен, но столь же серьезен. Дело в том, что пара ключей выработки подписи и проверки подписи могут быть использованы только один раз, так как выполнение процедуры подписи бита сообщения приводит к раскрытию половины секретного ключа, после чего он уже не является полностью секретным и не может быть использован повторно. Поэтому для каждого подписываемого сообщения необходим свой комплект ключей подписи и проверки. Это практически исключает возможность использования рассмотренной схемы Диффи-Хеллмана в первоначально предложенном варианте в реальных системах ЭЦП.

Однако, несколько лет назад Березин и Дорошкевич предложили модификацию схемы Диффи-Хеллмана, фактически устраняющую ее недостатки.

Центральным в этом подходе является алгоритм "односторонней криптографической прокрутки", который в некотором роде может служить аналогом операции возведения в степень. Предположим, что имеется криптографический алгоритм E_k с размером блока данных и ключа соответственно n и n_k бит, причем $n < n_k$. Пусть также имеется некоторая функция отображения n – битовых блоков данных в n_k – битовые $Y = P_{n \to n_k}(X), |X| = n, |Y| = n_k.$ Определим рекурсивную функцию R_k "односторонней прокрутки" блока данных T размером n бит k раз ($k > 0$) при помощи формулы

$$R_k(T) = \begin{cases} T, \ k =; \\ E_{P_{n \to n_k}(R_{k-1}(T))}(X), k >, \end{cases}$$

где X – произвольный несекретный n-битовый блок данных, являющийся параметром процедуры прокрутки. По своей идее функция односторонней прокрутки чрезвычайно проста, надо всего лишь нужное количество раз (K) выполнить следующие действия: расширить n-битовый блок данных T до размера ключа использованного алгоритма шифрования (n_k), на полученном расширенном блоке как на ключе зашифровать блок данных X, результат зашифрования занести на место исходного блока данных (T). По определению операция $R_k(T)$ обладает двумя важными свойствами.

1. Аддитивность и коммутативность по числу прокручиваний:

$$R_{k+k'}(T) = R_{k'}(R_k(T)) = R_k(R_{k'}(T)).$$

2. Односторонность или необратимость прокрутки: если известно только некоторое значение функции $R_k(T)$, то вычислительно невозможно найти значение $R_{k'}(T)$, для любого $k' < k$ – если бы это было возможно, то был бы способ определить ключ шифрования по известному входному и выходному блоку алгоритма E_k, что противоречит предположению о стойкости шифра.

При реализации данного алгоритма размер ключа проверки подписи равен удвоенному размеру блока данных использованного блочного шифра: $|k_S = 2n|$.

Вычисление и проверка ЭЦП будут выглядеть следующим образом:

Алгоритм Sig_{n_T} выработки цифровой подписи для n_T-битового блока T заключается в выполнении "односторонней прокрутки" обеих половин ключа подписи T и $2^{n_T} - 1 - T$ раз соответственно:

$$S = Sig_{n_T}(T) = (s_0, \ s_1) = R_T(k_0), \ P_{2^{n_T} - 1 - T}(k_1).$$

Алгоритм Ver_{n_T} проверки подписи состоит в проверке истинности соотношений $P_{2^{n_T}-1-T}(s_0) = C_0,\ R_T(S_1) = C_1$, которые должны выполняться для подлинного блока данных T:

$$P_{2^{n_T}-1-T}(s_0) = P_{2^{n_T}-1-T}((R_T(k_0)) = P_{2^{n_T}-1-T+T}(k_0) = P_{2^{n_T}-1}(k_0) = C_0;$$
$$P_T(s_1) = R_T(P_{2^{n_T}-1-T}(k_1)) = R_{T+2^{n_T}-1-T}(k_1) = P_{2^{n_T}-1}(k_1) = C_1.$$

Таким образом, функция проверки подписи будет следующей:

$$Ver(T,\ s,\ k_C) = \begin{cases} 1, R_{2^{n_T}-1-T}(s_0) = C_0 \& \quad R_T(s_1) = C_1; \\ 0, R_{2^{n_T}-1-T}(s_0) \neq C \& \quad R_T(s_1) \neq C_1. \end{cases}$$

Таким образом, рассмотренная модификация схемы Диффи-Хеллмана делает возможным подпись не одного бита, а целой битовой группы. Это позволяет в несколько раз уменьшить размер подписи и ключей подписи/проверки данной схемы. Однако надо понимать, что увеличение размера подписываемых битовых групп приводит к экспоненциальному росту объема необходимых вычислений и, начиная с некоторого значения, делает работу схемы также неэффективной. Граница "разумного размера" подписываемой группы находится где-то около десяти бит, и блоки большего размера все равно необходимо подписывать "по частям".

Размер ключа подписи и проверки подписи можно дополнительно уменьшить следующими приемами:

1. Нет необходимости хранить ключи подписи отдельных битовых групп, их можно динамически вырабатывать в нужный момент времени с помощью генератора криптостойкой гаммы. Ключом подписи в этом случае будет являться обычный ключ использованного в схеме подписи блочного шифра. Например, если схема подписи будет построена на алгоритме ГОСТ 28147–89, то размер ключа подписи будет равен 256 битам.
2. Аналогично, нет необходимости хранить массив ключей проверки подписи отдельных битовых групп блока, достаточно хранить его значение хэш-функции этого массива. При этом алгоритм выработки ключа подписи и алгоритм проверки подписи будут дополнены еще одним шагом – вычислением хэш-функции массива проверочных комбинаций отдельных битовых групп.

4.3. Функции хэширования

Функция хэширования (хэш-функция) H представляет собой отображение, на вход которого подается сообщение переменной длины M, а выходом является строка фиксированной длины $H(M)$. В общем случае $H(M)$ будет гораздо меньшим, чем M, например, $H(M)$ может быть 128 или 256 бит, тогда как M может быть размером в мегабайт или более.

Функция хэширования может служить для обнаружения модификации сообщения, т.е. она может служить в качестве криптографической контрольной суммы (также называемой кодом обнаружения изменений (*MDC – Manipulation Detection Code*) или проверкой целостности сообщения (*MIC – Message Integrity Check*)).

Теоретически возможно, что два различных сообщения могут быть сжаты в одну и ту же свертку (так называемая ситуация "столкновения"). Поэтому для обеспечения стойкости функции хэширования необходимо предусмотреть способ избегать столкновений. Полностью столкновений избежать нельзя, поскольку в общем случае количество возможных сообщений превышает количество возможных выходных значений функции хэширования. Однако вероятность столкновения должна быть низкой.

Для того чтобы функция хэширования могла должным образом быть использована в процессе аутентификации, функция хэширования H должна обладать следующими свойствами:

1. H может быть применена к аргументу любого размера.
2. Выходное значение H имеет фиксированный размер.
3. $H(x)$ достаточно просто вычислить для любого x. Скорость вычисления хэш-функции должна быть такой, чтобы скорость выработки и проверки ЭЦП при использовании хэш-функции

была значительно больше, чем при использовании самого сообщения.

4. Для любого y с вычислительной точки зрения невозможно найти x такое, что $H(x) = y$.
5. Для любого фиксированного x с вычислительной точки зрения невозможно найти $x' \neq x$ такое, что $H(x') = H(x)$.

Последнее свойство гарантирует отсутствие другого сообщения, дающего ту же свертку, что предотвращает подделку и позволяет использовать H в качестве криптографической контрольной суммы для проверки целостности.

Четвертое свойство эквивалентно тому, что H является односторонней функцией. Стойкость систем с открытыми ключами зависит от того, что открытое криптопреобразование является односторонней *функцией-ловушкой*. Напротив, функции хэширования являются односторонними функциями, не имеющими ловушек. Классическая хэш-функция является открытым преобразованием. В случае, когда она зависит от ключа, результат ее вычисления носит название *кода аутентификации сообщения* (*MAC – Message Authentication Code*).

4.3.1. ФУНКЦИЯ ХЭШИРОВАНИЯ *SHA*

Алгоритм безопасного хэширования *SHA* (*Secure Hash Algorithm*) принят в качестве стандарта США в 1992 г. и предназначен для использования совместно с алгоритмом цифровой подписи, определенным в стандарте *DSS*. При вводе сообщения M алгоритм вырабатывает 160-битовое выходное сообщение, называемое сверткой (*Message Digest*), которая и используется при выработке ЭЦП.

Рассмотрим работу алгоритма подробнее. Пусть исходное сообщение дополняется так, чтобы его длина стала кратной 512 битам. При этом сообщение дополняется даже тогда, когда его длина уже кратна указанной. Процесс происходит следующим образом: добавляется единица, затем столько нулей, сколько необходимо для получения сообщения, длина которого на 64 бита меньше, чем кратная 512, и затем добавляется 64-битовое представление длины исходного сообщения.

Далее инициализируются пять 32-битовых переменных следующими шестнадцатеричными константами:

A = 67452301 B = EFCDAB89 C = 98BADCFE D = 10325476 E = C3D2E1F0

Эти пять переменных копируются в новые переменные a, b, c, d и e соответственно.

Главный цикл может быть описан на псевдокоде следующим образом:

```
for (t=0; t<80; t++){
   temp = (a <<< 5)+f_t(b, c, d) + e + W_t + K_t;
   e = d;  d = c;  c = b <<<30; b = a; a = temp;
}
```

где <<< – операция циклического сдвига влево; K_t – 16-ричные константы, определяемые как:

$$K_t = \begin{cases} 5A827999, & t = 0...19; \\ 6ED9EDA1, & t = 20...39; \\ 8F1BBCDC, & t = 40...59; \\ CA62C1D6, & t = 60...79, \end{cases}$$

функции $f_t(x, y, z)$ задаются следующими выражениями:

$$f(x,\ y,\ z)_t = \begin{cases} X \wedge Y \vee \neg X \wedge Z, & t = 0...19; \\ X \oplus Y \oplus Z, & t = 20...39,\ 60...79; \\ X \wedge Y \vee X \wedge Z \vee Y \wedge Z, & t = 40...59, \end{cases}$$

значения W_t получаются из 32-битовых подблоков 512-битового блока расширенного сообщения по следующему правилу:

$$W_t = \begin{cases} M_t, \ t = 0...19; \\ (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) <<< 1, \ t = 16...79. \end{cases}$$

После окончания главного цикла значения *a*, *b*, *c*, *d* и *e* складываются с содержимым *A*, *B*, *C*, *D* и *E* соответственно и осуществляется переход к обработке следующего 512-битового блока расширенного сообщения. Выходное значение хэш-функции является конкатенацией значений *A*, *B*, *C*, *D* и *E*.

4.3.2. ФУНКЦИИ ХЭШИРОВАНИЯ *SHA*-256, *SHA*-512 И *SHA*-384

Стойкость функции хэширования к поиску столкновений, примерно, равна $2^{n/2}$, где *n* – длина выходного значения функции. В связи с разработкой в США нового стандарта шифрования с длиной ключа 128, 192 и 256 бит потребовалось создать "сопровождающие" алгоритмы, обеспечивающие такой же уровень стойкости. В качестве нового стандарта США предполагается принять алгоритмы вычисления функций хэширования с длиной выходного значения 256, 384 и 512 бит.

Работа алгоритма *SHA*-256 может быть разбита на две части – описание функции сжатия и алгоритма обработки сообщения. Функция сжатия представляет собой по сути алгоритм блочного шифрования с размером блока 256 бит промежуточного значения функции хэширования с использованием очередного текстового блока в качестве ключа. Размер слова равен 32 битам. Сложение производится по модулю 2^{32}. Стартовый вектор хэширования H^0 представляет собой набор из 8 32–разрядных слов, получаемых взятием дробной части квадратных корней первых 8 простых чисел. Далее вычисление происходит по схеме подобной *SHA*, более подробно алгоритм изложен [7]. В *SHA*-256 используются шесть логических функций, аргументы и значения которых – 32-битовые слова. Слова-константы K_0, ..., K_{63} берутся как первые 32 бита дробных частей кубических корней первых 64 простых чисел в 16-ричном виде.

Функция *SHA*-512 подобна по своей структуре *SHA*-256, но работает с размером слова 64 бита. Вначале текст дополняется так, чтобы его длина была кратна 1024. Процедура дополнения аналогична: добавляется 1, затем столько нулей, что длина текста станет на 128 меньше, нежели кратная 1024, а затем 128-битовое представление длины исходного текста. Стартовый вектор хэширования задается следующим образом: берутся первые 64 бита дробных частей квадратных корней первых 8 простых чисел. Далее исходный текст разбивается на блоки по 1024 бита. Слова-константы K_0, ..., K_{79} берутся как первые 64 бита дробных частей кубических корней первых 80 простых чисел в 16-ричном виде.

Функция *SHA*-384 определяется точно так же, как и функция *SHA*-512 с тем исключением, что в качестве стартового вектора хэширования берутся первые 64 бита квадратных корней простых чисел с девятого по шестнадцатое. Далее выход функции обрезается до 384 левых бит и эти биты берутся в качестве значения функции хэширования *SHA*-384.

4.3.3. ФУНКЦИЯ ХЭШИРОВАНИЯ ГОСТ Р 34.11–94

При описании функции хэширования будут использоваться те же обозначения, что использовались при описании алгоритма выработки цифровой подписи согласно ГОСТ Р 34.10. Кроме того, пусть *M* – последовательность двоичных символов, подлежащих хэшированию; *h* – хэш-функция, отображающая последовательность *M* в слово $h(M) \in V_{256}(2)$; $E_K(A)$ – результат шифрования слова *A* на ключе *K* с использованием алгоритма шифрования по ГОСТ 28147–89 в режиме простой замены; *H* – стартовый вектор хэширования.

Под хэш-функцией *h* понимается отображение $h : B^* \to V_{256}(2)$.

Для определения хэш-функции необходимы: алгоритм вычисления шаговой функции хэширования κ, где $\kappa : V_{256}(2) \times V_{256}(2) \to V_{256}(2)$; описание итеративной процедуры вычисления значения хэш-функции *h*.

Алгоритм вычисления шаговой функции хэширования состоит из трех частей: генерации четырех 256-битных ключей; шифрующего преобразования – шифрования 64-битных подслов слова H на ключах K_i (i = 1, 2, 3, 4) с использованием алгоритма ГОСТ 28147–89 в режиме простой замены; перемешивающего преобразования результата шифрования.

Генерация ключей. Рассмотрим $X = (b_{256},\ \ b_{255},\ ...,\ b_1) \in V_{256}(2)$.

Пусть $X = x_4 \| x_3 \| x_2 \| x_1 = \eta_{16} \| \eta_{15} \| ... \| \eta_1 = \xi_{16} \| \xi_{15} \| ... \| \xi_1$,

где $x_i \in V_{64}(2),\ \ i = 1, ..., 4;\ \eta_j \in V_{16}(2),\ \ j = 1, ..., 16;\ \xi_k \in V_8(2),\ \ i = 1, ..., 32$.

Обозначим $A(X) = (x_1 \oplus x_2) \| x_4 \| x_3 \| x_2$.

Задается преобразование $P : V_{256}(2) \to V_{256}(2)$ слова $\xi_{32} \| ... \| \xi_1$

$\xi_{\varphi(32)} \| \xi_{\varphi(31)} \| ... \| \xi_{\varphi(1)}$, где $\varphi(i + 1 + 4(k-1)) = 8i + k,\ i = 0, ..., 3,\ \ k = 1, ..., 8.$

Для генерации ключей необходимо использовать следующие исходные данные:

- слова $H, M \in V_{256}(2)$;
- константы: слова C_i (i = 2, 3, 4), имеющие значения $C_2 = C_4 = = 0^{256}$ и $C_3 = 1^8 0^8 1^{16} 0^{24} 1^{16} 0^8 (0^8 1^8)^2 1^8 0^8 (0^8 1^8)^4 (1^8 0^8)^4$.

При вычислении ключей реализуется следующий алгоритм:

1. Присвоить значения $I = 1$, $U = H$, $V = M$.
2. Выполнить вычисление $W = U \oplus V$, $K_1 = P(W)$.
3. Присвоить $i = i + 1$.
4. Проверить условие $i = 5$. При положительном исходе перейти к шагу 7. При отрицательном – перейти к шагу 5.
5. Выполнить $U = A(U) \oplus C_i,\ \ V = A(A(V)),\ \ W = U \oplus V,\ \ K_i = P(W)$.
6. Перейти к шагу 3.
7. Конец работы алгоритма.

Шифрующее преобразование. На данном этапе осуществляется шифрование 64-битных подслов слова H на ключах K_i (i = 1, 2, 3, 4). Для шифрующего преобразования необходимо использовать следующие исходные данные:

$H = h_4 \| h_3 \| h_2 \| h_1,\ \ h_i \in V_{64}(2),\ \ i = 1...4$ и набор ключей K_1 K_2, K_3, K_4.

После выполнения шифрования получают слова $s_i = E_{K_i}(h_i)$, где I = 1, 2, 3, 4, т.е. в результате получается вектор $S = s_4 \| s_3 \| s_2 \| s_1$.

Перемешивающее преобразование. На данном этапе осуществляется перемешивание полученной последовательности с применением регистра сдвига.

Исходными данными являются слова H, $M \in V_{256}(2)$ и слово $S \in V_{256}(2)$.

Пусть отображение $\psi : V_{256}(2) \to V_{256}(2)$ преобразует слово

$\eta_{16} \| \eta_{15} \| ... \| \eta_1 \in V_{16}(2),\ i = 1...16$ в слово

$\eta_1 \oplus \eta_2 \oplus \eta_3 \oplus\ \eta_4 \oplus \oplus \eta_{13} \oplus \eta_{16} \| ... \| \eta_2$.

Тогда в качестве значения шаговой функции хэширования принимается слово $\kappa(M,\ \ H) = \psi^{61}(H \oplus \psi(M \oplus \psi^{12}(S)))$, где ψ^i – i-я степень преобразования ψ.

Процедура вычисления хэш-функции. Исходными данными для процедуры вычисления значения функции h является подлежащая хэшированию последовательность $M \in B$. Параметром является стартовый вектор хэширования H – произвольное фиксированное слово из $V_{256}(2)$.

Процедура вычисления функции h на каждой итерации использует следующие величины: $M \in B^*$ – часть последовательности M, не прошедшая процедуры хэширования на предыдущих итерациях;

$H \in V_{256}(2)$ – текущее значение хэш-функции;

$\Sigma \in V_{256}(2)$ – текущее значение контрольной суммы;

$L \in V_{256}(2)$ – текущее значение длины обработанной на предыдущих итерациях части последовательности M.

Алгоритм вычисления функции h включает в себя три этапа.

Этап 1

Присвоить начальные значения текущих величин

$$M := M;\ H := H;\ \Sigma := 0^{256};\ L := 0^{256}.$$

Этап 2

Проверить условие $|M| > 256$. Если да, то перейти к этапу 3. В противном случае выполнить последовательность вычислений:

$$L := <L + |M|>_{256};\ M' := 0^{256-|M|}\ ||M;\ \Sigma := \Sigma\ [+]\ M';$$

$$H := \kappa(M', H);\ H := \kappa(L, H);\ H := \kappa(\Sigma, H).$$

Конец работы алгоритма. H содержит значение хэш-функции.

Этап 3

Вычислить подслово $M_S \in V_{256}(2)$ слова M ($M = M_P||M_S$). Далее выполнить последовательность вычислений:

$$H := \kappa(M_S, H);\ L := <L + 256>_{256};\ \Sigma := \Sigma\ [+]\ M_S;\ M := M_P.$$

Перейти к этапу 2.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Изложите принципиальную схему организации обмена документами, заверенными цифровой подписью.
2. Перечислите основные требования, предъявляемые к хэш-функции, пригодной для использования при вычислении цифровой подписи документа.
3. **Назовите и охарактеризуйте методы, реализующие ЭЦП.**
4. Расскажите, каким образом можно организовать Стандарт цифровой подписи *DSS*. Приведите примеры.
5. Расскажите, каким образом с помощью криптосистемы *RSA* можно организовать передачу сообщений, подлинность которых подтверждена цифровой подписью. Приведите примеры.
6. Охарактеризуйте и поясните на примерах работу стандартов цифровой подписи ГОСТ Р 34.10–94 и ГОСТ Р 34.10–2001
7. Сравните наиболее распространенные стандарты ЭЦП.

5. УПРАВЛЕНИЕ КРИПТОГРАФИЧЕСКИМИ КЛЮЧАМИ

5.1. ОБЫЧНАЯ СИСТЕМА УПРАВЛЕНИЯ КЛЮЧАМИ

Помимо выбора подходящей для конкретной информационной системы средств криптографической защиты информации, важной проблемой является управление ключами. Как бы ни была сложна и надежна сама криптосистема, она основана на использовании ключей. Если для обеспечения конфиденциального обмена информацией между двумя пользователями процесс обмена ключами тривиален, то в информационной системе, где количество пользователей составляет сотни и тысячи, управление ключами – серьезная проблема.

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

Управление ключами – информационный процесс, включающий в себя три элемента: генерацию ключей; накопление ключей; распределение ключей.

Не стоит использовать неслучайные ключи с целью легкости их запоминания. В серьезных информационных системах используются специальные аппаратные и программные методы генерации случайных ключей. Как правило, используют датчики ПСЧ. Однако степень случайности их гене-

рации должна быть достаточно высокой. Идеальными генераторами являются устройства на основе "натуральных" случайных процессов. Например, появились серийные образцы генерации ключей на основе белого радиошума. Физический датчик случайных чисел встроен в ядро процессора Pentium-III. Другим случайным математическим объектом являются десятичные знаки трансцендентных чисел, например π или e, которые вычисляются с помощью стандартных математических методов.

В ИС со средними требованиями защищенности вполне приемлемы программные генераторы ключей, которые вычисляют ПСЧ как сложную функцию от текущего времени и (или) числа, введенного пользователем.

Под накоплением ключей понимается организация их хранения, учета и удаления.

Поскольку ключ является самым привлекательным для злоумышленника объектом, открывающим ему путь к конфиденциальной информации, то секретные ключи никогда не должны записываться в явном виде на носителе, который может быть считан или скопирован.

В достаточно сложной ИС один пользователь может работать с большим объемом ключевой информации, и иногда даже возникает необходимость организации мини-баз данных по ключевой информации. Такие базы данных отвечают за принятие, хранение, учет и удаление используемых ключей.

Вся информация об используемых ключах должна храниться в зашифрованном виде. Ключи, зашифровывающие ключевую информацию, называются мастер-ключами. Желательно, чтобы мастер-ключи каждый пользователь знал наизусть, и не хранил их вообще на каких-либо материальных носителях.

Очень важным условием безопасности информации является периодическое обновление ключевой информации в ИС. При этом переназначаться должны как обычные ключи, так и мастер-ключи. В особо ответственных ИС обновление ключевой информации желательно делать ежедневно.

Для описания алгоритмов и методов в этой главе потребуются следующие обозначения:

I_A – идентификатор стороны A;

D_A – секретное криптопреобразование стороны A (с использованием секретного ключа асимметричной криптосистемы);

E_A – открытое криптопреобразование стороны A (с использованием открытого ключа асимметричной криптосистемы);

T_A – временной штамп стороны A;

R_A – случайное число, выбранное стороной A.

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

1. оперативность и точность распределения;
2. скрытность распределяемых ключей.

В рамках симметричной (одноключевой) системы шифрования двум пользователям, желающим установить безопасное взаимодействие, необходимо сначала установить безопасный общий ключ. Одной из возможностей является использование третьей стороны, такой как курьера. На практике по соображениям безопасности необходимо время от времени менять ключ. Это может сделать использование курьера или другой подобной схемы дорогостоящим и неэффективным.

Альтернативой является получение двумя пользователями общего ключа от центрального органа – центра распределения ключей (ЦРК), с помощью которого они могут безопасно взаимодействовать. Для организации обмена данными между ЦРК и пользователем последнему при регистрации выделяется специальный ключ, которым шифруются сообщения, передаваемые между ними. Поскольку каждому пользователю выделяется отдельный ключ, его компрометация не приведет к особо неприятным последствиям. Слабое место этого подхода заключается в следующем: в ЦРК, обладающий доступом к ключам, возможно проникновение злоумышленника. Вследствие концентрации доверия одно нарушение безопасности скомпрометирует всю систему. Кроме того, ЦРК может, например, долгое время участвовать в пассивном подслушивании, прежде чем это будет обнаружено, хотя доказать это может оказаться трудно.

В больших сетях эта процедура может стать узким местом, поскольку

каждой паре пользователей, нуждающейся в ключе, необходимо хотя бы один раз обратиться к центральному узлу. Кроме того, сбой центрального органа может разрушить систему распределения ключей. Иерархическая (древовидная) система с пользователями, находящимися на листьях, и центрами распределения ключей в промежуточных узлах является одним из способов смягчения этой проблемы. Однако, это создает новую проблему безопасности, поскольку создается множество точек входа для злоумышленника. Более того, данная система будет неэффективной, если пара часто взаимодействующих пользователей не будет находиться в одном поддереве, поскольку в этом случае корень дерева вновь окажется узким местом.

Некоторые из этих недостатков можно устранить, применив подход к распределению ключей, основанный на системах с открытым ключом.

5.2. УПРАВЛЕНИЕ КЛЮЧАМИ, ОСНОВАННОЕ НА СИСТЕМАХ С ОТКРЫТЫМ КЛЮЧОМ

До использования криптосистемы с открытым ключом для обмена обычными секретными ключами пользователи *A* и *B* должны обменяться своими открытыми ключами. Эта проблема проще, чем обмен секретными ключами, поскольку открытые ключи не требуют секретности при хранении и передаче. Управление открытыми ключами может быть организовано с помощью оперативной или автономной службы каталогов, пользователи могут также обмениваться ключами непосредственно. Однако, проблемой здесь является аутентичность. Если *A* думает, что E_c в действительности является $E_в$, то *A* может зашифровать сообщение с помощью E_c и ненамеренно дать возможность *C* расшифровать сообщение, используя D_c. Второй проблемой является целостность: любая ошибка в передаче открытого ключа сделает его бесполезным. Поэтому желательно наличие какой-либо формы обнаружения ошибок. Вне зависимости от схемы, выбранной для распределения открытых ключей, скорее всего на каком-либо этапе будет участвовать центральный орган. Однако, обмен открытыми ключами между пользователями не требует участия центрального органа, поскольку основной проблемой является аутентичность. Следовательно, последствия компрометации центрального органа будут не столь тяжелыми, как в случае обычной ключевой системы.

Еще одним аспектом проблемы является достоверность: открытый ключ пользователя может оказаться недостоверным вследствие компрометации соответствующего секретного ключа или по какой-либо иной причине, например, из-за истечения срока действия, что создает проблему устаревших данных, если открытые ключи хранятся или доступ к ним осуществляется через каталог.

В качестве примера можно привести протокол обмена ключом Диффи-Хеллмана. Системными параметрами этого протокола являются большое простое число *p* и число *g*, являющееся примитивным элементом *GF*(*p*).

Пусть теперь пользователи *A* и *B* желают получить общий секретный ключ. Сначала *A* генерирует случайное число *a*, которое он держит в секрете, а *B* генерирует случайное число *b*, которое он также держит в секрете. Затем они вычисляют $g^a \bmod p$ и $g^b \bmod p$ соответственно и передают вычисленные значения друг другу, причем это можно сделать по открытым каналам связи. После этого *A* вычисляет $k_{ab} = (g^b \bmod p)^a \bmod p$, а *B* вычисляет $k_{ba} = (g^a \bmod p)^b \bmod p$. Поскольку $k_{ab} = k_{ba} = k$, у *A* и *B* теперь есть общий секретный ключ *k*.

Однако протокол Диффи-Хеллмана является уязвимым для атаки, называемой "человек в середине". Злоумышленник *C* может перехватить открытое значение, посылаемое от *A* к *B*, и послать вместо него свое открытое значение. Затем он может перехватить открытое значение, посылаемое от *B* к *A*, и также послать вместо него свое открытое значение. Тем самым *C* получит общие секретные ключи с *A* и *B* и сможет читать и/или модифицировать сообщения, передаваемые от одной стороны к другой.

5.3. ПРОТОКОЛ ОБМЕНА СЕКРЕТНЫМ КЛЮЧОМ

Для защиты от атаки, рассмотренной в предыдущем параграфе, можно использовать протокол, описанный ниже. При этом предполагается, что *A* и *B* обладают возможностью проверить аутентичность открытых ключей

друг друга.

Предположим, что *A* и *B* желают определить общий секретный ключ (*K*). При этом если они получили открытые ключи друг друга, то тогда можно использовать трехэтапный протокол рукопожатия.

A может послать *B* сообщение $C = E_B(R_A, I_A)$, где E_B – процедура шифрования с открытым ключом *B*, I_A – идентификатор *A* и R_A – случайное число. Теперь *B* может расшифровать *C* и получить I_A. Теперь *B* выбирает случайное число R_B и посылает $C' = E_A(R_A, I_B)$ *A*. После расшифрования *C′* *A* может в реальном времени проверить, что *B* получил R_A, поскольку только *B* может расшифровать *C*. Наконец, *A* посылает *B* $C'' = E_B(K_B)$, и когда *B* расшифрует *C″*, он сможет проверить в реальном времени, что *A* получил R_B, поскольку только *A* может расшифровать *C′*. Тем самым *A* и *B* аутентифицировали друг друга, т.е. каждый из них знает, что они общаются именно друг с другом.

Теперь *A* посылает *B* $E_B(D_A(K))$, *B* расшифровывает сообщение и получает *K*. Данная процедура обеспечивает как секретность, так и аутентичность при обмене ключом *K*.

Рассмотренная выше процедура является вариантом так называемого механизма "запрос-ответ", который заключается в том, что для аутентификации контрагента пользователь направляет ему некоторое непредсказуемое заранее сообщение, на которое тот должен дать ответ, выполнив некоторую заранее обусловленную операцию над сообщением-запросом. После получения ответа пользователь может быть уверен в подлинности сеанса связи. Недостатком этого метода является возможность установления, хотя и сложной, закономерности между запросом и ответом.

5.4. ИСПОЛЬЗОВАНИЕ СЕРТИФИКАТОВ

Метод достижения одновременно аутентичности и целостности при распределении открытых ключей заключается в использовании сертификатов. Система, основанная на сертификатах, предполагает, что имеется центральный орган (ЦО), как и в случае распределения секретных ключей. Далее предполагается, что каждый пользователь может осуществлять безопасное взаимодействие с ЦО. Для этого требуется, чтобы у каждого пользователя был открытый ключ ЦО – $E_{\text{ЦО}}$. Тогда каждый пользователь *A* может зарегистрировать в ЦО свой открытый ключ E_A. Поскольку $E_{\text{Д}}$ является открытым, это можно сделать по почте, по открытому каналу электросвязи и т.п.

Обычно при регистрации в ЦО *A* будет следовать определенной аутентификационной процедуре. Альтернативным вариантом может быть обработка регистрации системой, имеющей древовидную структуру: ЦО выдает сертификаты местным представителям, которые в дальнейшем действуют в качестве посредников в процессе регистрации пользователя на более низких уровнях иерархии.

В любом случае, в ответ *A* получает сертификат, подписанный ЦО и содержащий E_A, т.е. ЦО формирует сообщение *M*, содержащее E_A, идентификационную информацию для *A* (I_A), период действия сертификата и т.п. Затем ЦО вычисляет $CERT_A = D_{\text{ЦО}}(M)$, который и становится сертификатом *A*. $CERT_A$ делается общедоступным документом, который содержит E_A и одновременно аутентифицирует его, поскольку сертификат подписан ЦО. Сертификаты могут распространяться ЦО, пользователями или использоваться в иерархической системе. Включение срока действия является обобщением временного штампа, что обеспечивает защиту от использования скомпрометированных ключей.

Однако, проблема устаревших данных не может быть решена только с помощью временных штампов, поскольку сертификат может стать недействительным до истечения срока его действия вследствие компрометации или по административным причинам. Поэтому, если сертификаты хранятся у пользователей (а не выдаются каждый раз ЦО при их использовании), ЦО должен время от времени публиковать списки аннулированных сертификатов.

Некоторые свойства рассмотренных схем могут быть объединены в подход, известный под названием "телефонный справочник", с использованием электронного эквивалента, такого как гибкий диск, содержащий сер-

тификаты. Это облегчит использование, поскольку пользователь сможет быстро связаться с другим, очень быстро получая доступ к сертификату последнего.

5.5. ПРОТОКОЛЫ АУТЕНТИФИКАЦИИ

Предположим, что A желает установить связь с B, получил сертификационный путь от A до B, например обратившись к каталогу, и использовал этот путь для получения открытого ключа B.

Пусть I_A – идентификатор стороны A; D_A – секретное криптопреобразование стороны A (секретный ключ); E_A – открытое криптопреобразование стороны A (открытый ключ); T_A – временной штамп стороны A; R_A – случайное число, выбранное стороной A; C_A – сертификат стороны.

I_B – идентификатор стороны B; D_B – секретное криптопреобразование стороны B (секретный ключ); E_B – открытое криптопреобразование стороны B (открытый ключ); T_B – временной штамп стороны B; R_B – случайное число, выбранное стороной B.

Идентификаторы – это уникальные имена A и B. Временной штамп, включаемый в сообщение M, содержит также дату истечения срока действия M. Дополнительно он также может включать время создания M. Случайные числа могут быть заменены последовательными числами, которые не должны повторяться в течение срока действия, указанного во временном штампе в том же сеансе связи.

Тогда односторонний протокол аутентификации будет выглядеть следующим образом:

Пользователь A:

1. Выбирает R_A.
2. Формирует сообщение $M = (T_A, R_A, I_B, \langle$данные$\rangle)$, где $\langle$*данные*$\rangle$ произвольны. Данные могут быть зашифрованы с помощью E_B для секретности, например, когда A передает B ключ шифрования данных.
3. Посылает $(C_A, D_A(M))$ пользователю B.

Пользователь B:

1. Расшифровывает C_A и получает E_A. Проверяет дату окончания срока действия сертификата;
2. Использует E_A для расшифрования $D_A(M)$, проверяя как подлинность подписи A, так и целостность подписанной информации.
3. Проверяет I_B, содержащееся в M, на точность.
4. Проверяет T_A в M.
5. Дополнительно проверяет R_A, содержащееся в M.

Широкое распространение основанных на интеллектуальных картах систем доступа для различного рода приложений (как гражданского, так и военного назначения) потребовало создать схему обеспечения безопасной аутентификации субъекта. При этом секретный ключ владельца карты становится неотъемлемым признаком его личности, и для обеспечения защиты от возможной компрометации этого ключа был предложен ряд схем, называемых протоколами доказательства с нулевым разглашением или с нулевым знанием (*zero-knowledge proofs*), подтверждающий полномочия субъекта, не раскрывая значения секретного ключа.

Первая схема подобного рода предложена в 1986 г. Фейге, Фиатом и Шамиром. Суть ее состоит в следующем.

Для группы пользователей, которым придется доказывать свою подлинность, выбирается большое (длиной более 512 бит) случайное целое число n, являющееся произведением двух простых чисел. В процессе аутентификации участвуют две стороны: сторона A, доказывающая свою подлинность, и сторона B – проверяющий.

Доверенный арбитр (центр распределения ключей) выбирает некоторое целое число v, являющееся квадратичным вычетом по модулю n, т.е. $\exists x : x^2 = v(\mathrm{mod}\, n)$, и взаимно простым с n. Это значение v передается A в качестве открытого ключа. Затем вычисляется наименьшее значение s, такое что $s = (v^{-1})^{1/2}(\mathrm{mod}\, n)$. Это значение будет секретным ключом стороны A.

Далее протокол аутентификации выглядит следующим образом:

1. Сторона A выбирает случайное число r, $0 < r < n$. Затем она вычис-

ляет $x = r^2 \bmod n$ и отправляет его стороне *B*.

2. Сторона *B* посылает *A* случайный бит *b.*

3. Если $b = 0$, то *A* отправляет *B* число *r.* Если $b = 1$, то *A* отправляет *B*: $y = rs \pmod n$.

4. Если $b = 0$, то *B* проверяет, что $x = r^2 \pmod n$, чтобы убедиться, что А знает квадратный корень из *x.* Если $b = 1$, то сторона *B* проверяет, что $x = y^2 v \pmod n$, чтобы убедиться, что *A* знает квадратный корень из v^{-1}.

Шаги 1 – 4 образуют один цикл протокола. Стороны повторяют этот цикл *t* раз при разных случайных значениях *r* и *b.* Если сторона *A* не знает значения *s,* она может выбрать такое *r*, которое позволит ей обмануть *B* в случае $b = 0$ или $b = 1$, но не в обоих случаях одновременно. Вероятность обмана в одном цикле составляет 0,5. Вероятность обмана в *t* циклах равна 2^{-t}.

Недостатком данной схемы является большое число циклов протокола, необходимое для доказательства с требуемой вероятностью, если эта вероятность достаточно мала. Способ, требующий только одного раунда обмена, но требующий большего объема вычислений, был предложен Гиллоу и Кискатером (*Guillou*, *Quisquater*).

Пусть *I* – идентификационная информация стороны *A* (или значение ее хэш-функции); *n* – открытое произведение двух секретных простых чисел; *v* – открытое значение (показатель степени).

Секретный ключ *g* стороны *A* выбирается так, что $Ig^v = 1 \pmod n$.

Сторона *A* отправляет *B* свои идентификационные данные *I*. Протокол доказательства выглядит следующим образом.

1. *A* выбирает случайное целое *r* ($1 < r < n - 1$), вычисляет $T = r^v \pmod n$ и отправляет это значение стороне *B*.

2. *B* выбирает случайное целое *d* ($1 < d < n - 1$) и отправляет это число стороне *A*.

3. *A* вычисляет $D = rg^d \pmod n$ и отправляет это значение *B*.

4. *B* вычисляет $T' = D^v I^d \pmod n$ и проверяет выполнение равенства $T' = T$. Если оно выполняется, то проверка считается завершенной успешно.

5.6. АНОНИМНОЕ РАСПРЕДЕЛЕНИЕ КЛЮЧЕЙ

Если мы полагаем, что пользователи сами не могут выбирать собственные ключи, то они должны пользоваться услугами центра распределения ключей. Проблема заключается в том, что ключи должны распределяться так, чтобы никто не мог определить, кто получил какой ключ. Процедура распределения ключей в этом случае может выглядеть так:

1. *A* выбирает пару ⟨открытый ключ, секретный ключ⟩ (для этого протокола он держит оба ключа в секрете).
2. ЦРК генерирует непрерывный поток ключей.
3. ЦРК шифрует ключи, один за одним, своим открытым ключом.
4. ЦРК передает зашифрованные ключи, один за одним, в сеть.
5. *A* выбирает ключ случайным образом.
6. *A* шифрует выбранный ключ своим открытым ключом.
7. *A* ожидает некоторое время и посылает дважды зашифрованный ключ обратно в ЦРК.
8. ЦРК расшифровывает дважды зашифрованный ключ своим секретным ключом, оставляя ключ зашифрованным один раз открытым ключом *A*.
9. ЦРК посылает зашифрованный ключ назад пользователю *A*.
10. *A* расшифровывает ключ своим секретным ключом.

Контрольные вопросы

1. Перечислите основные понятия, связанные с управлением ключевой информацией.
2. **Поясните работу обычной системы управления ключами.**
3. Расскажите, каким образом осуществляется управление ключами, основанное на системах с открытым ключом. Охарактеризуйте протокол обмена секретным ключом.
4. Укажите преимущества систем использующих сертификаты безопасности.
5. Поясните работу протоколов аутентификации.

6. АППАРАТНО-ПРОГРАММНЫЕ СРЕДСТВА КРИПТОГРАФИЧЕСКОЙ ЗАЩИТЫ ИНФОРМАЦИИ

В соответствии с законодательством Российской Федерации в области защиты информации государственным организациям и предприятиям предписывается использовать криптографические средства защиты информации, сертифицированные ФСБ России (до марта 2003 г. сертификация криптографических средств возлагалась на Федеральное агентство правительственной связи и информации (ФАПСИ) при Президенте РФ). В настоящее время на рынке средств защиты информации России активно работают фирмы "АНКАД", "Информзащита", "ВАЛИДАТА", ФГУП "НТЦ "Атлас", Пензенский научно-исследовательский электротехнический институт (ПНИЭИ), ЗАО "Московское отделение (МО) ПНИЭИ" и др. Спектр выпускаемой продукции широк. Это аппаратные, аппаратно-программные средства защиты документальной и речевой информации, комплексы для реализации *PKI* (инфраструктуры открытых ключей), средства защиты электронной почты и пр. Ниже приведены краткие характеристики криптографических средств защиты информации, которые нашли широкое применение в государственных предприятиях, правоохранительных органах и финансово-кредитных организациях.

6.1. УСТРОЙСТВА КРИПТОГРАФИЧЕСКОЙ ЗАЩИТЫ ДАННЫХ СЕРИИ "КРИПТОН"

Устройства криптографической защиты данных (УКЗД) серии "КРИПТОН" – это аппаратные шифраторы для *IBM PC*-совместимых компьютеров (рис. 6.1) применяются в составе средств и систем криптографической защиты данных и используются в государственных и коммерческих структурах.

Рис. 6.1. Внешний вид устройства "КРИПТОН"

Устройства "КРИПТОН" гарантируют защиту информации, обрабатываемой на персональном компьютере и передаваемой по открытым каналам связи, выполнены в виде плат расширения *ISA* и *PCI* персонального компьютера с процессором 1386 и выше.

Устройства "КРИПТОН" разработаны, производятся и реализуются фирмой "АНКАД". Они построены на разработанных этой фирмой специализированных 32-разрядных шифр процессорах. Устройства серии КРИПТОН сертифицированы ФАПСИ и ФСБ (в том числе в составе абонентских пунктов и автоматизированных рабочих мест для защиты информации, содержащей сведения, составляющие государственную тайну).

Преимущества устройств серии "КРИПТОН":

- аппаратная реализация алгоритма криптографического преобразования гарантирует *целостность алгоритма*;
- шифрование производится и ключи шифрования хранятся в самой плате, а *не в оперативной памяти компьютера*;
- загрузка ключей шифрования в устройство "КРИПТОН" со смарт-карт и идентификаторов *Touch Memory* производится напрямую, минуя ОЗУ и системную шину компьютера, что исключает возможность перехвата ключей;
- применение специализированного шифрпроцессора для выполнения криптографических преобразований разгружает центральный процессор компьютера.

Для устройств "КРИПТОН" разработано специальное программное обеспечение, которое позволяет:

- шифровать компьютерную информацию (файлы, группы файлов и разделы дисков), обеспечивая их конфиденциальность;
- осуществлять электронную цифровую подпись файлов, проверяя их целостность и авторство;
- создавать прозрачно шифруемые логические диски, максимально облегчая и упрощая работу пользователя с конфиденциальной информацией;
- формировать криптографически защищенные виртуальные сети, шифровать *IP*-трафик;
- создавать системы защиты информации от несанкционированного доступа и разграничения доступа к компьютеру.

Основные технические данные и характеристики

Алгоритм шифрования	ГОСТ 28147–89
Размерность ключа шифрования, бит	256
Поддерживаемые операционные системы	MS-DOS, Windows 95(98)/ME/NT 4.0/2000/XP/2003 UNIX

Фирма "АНКАД" выпускает следующие модели УКЗД: КРИПТОН-4, КРИПТОН-4/PCI, КРИПТОН-8/PCI, КРИПТОН-9/PCI, КРИПТОН-ЗАМОК и др. Модели отличаются скоростью шифрования и сервисными возможностями.

КРИПТОН-4/PCI, КРИПТОН-8/PCI – устройства криптографической защиты данных и ограничения доступа к компьютеру.

Шина	PCI
Реализация алгоритма шифрования	аппаратная
Скорость шифрования, Кбайт/с	до 1100 (КРИПТОН-4/PCI) до 8500 (КРИПТОН-8/PCI)
Носители ключей	дискеты, смарт-карты с открытой и защищенной памятью, микропроцессорные смарт-карты, носители Touch-Memory

Программы шифрования и ЭЦП серии КРИПТОН/Crypton

Программные средства криптографической защиты информации серии КРИПТОН/Crypton осуществляют функции шифрования, ЭЦП и комплексной защиты информации, призваны обеспечить безопасность различной конфиденциальной информации – коммерческой, банковской, страховой, налоговой тайны, персональных данных и т.п.

Программные продукты серии КРИПТОН/Crypton работают автономно или на базе шифраторов КРИПТОН посредством оригинальной библиотеки *Crypton API*, которая обеспечивает совместимость шифраторов с любым программным обеспечением фирмы "АНКАД". Кроме того, фирмой "АНКАД" разработан программный продукт *Crypton Emulator*, который эмулирует функции шифрования устройств криптографической защиты данных (УКЗД) серии "КРИПТОН", т.е. возможна практически полноценная замена УКЗД продуктом *Crypton Emulator* с использованием программного интерфейса *Crypton API*. В программном обеспечении *Crypton Emulator* реализован алгоритм шифрования ГОСТ 28147–89.

"АНКАД" предлагает следующие комплексные программные продукты:

- КРИПТОН-Шифрование для Windows 95/98/ME/NT 4.0/2000/XP/2003 – архивное и абонентское шифрование.
- КРИПТОН-Подпись для Windows 95/98/ME/NT 4.0/2000/XP/2003 – электронная цифровая подпись, открытый ключ 512 и 1024 бит.
- *CryptonArcMail* для Windows 95/98/ME/2000/NT 4.0/XP/2003 – серия программных продуктов предназначенных для защиты от несанкционированного доступа к документам-файлам, передаваемым по электронной почте в локальных и глобальных сетях, единый сервис архивирования, архивного и абонентского шифрования и ЭЦП. Симметричная и ассиметричная

ключевые системы, открытый ключ 512 и 1024 бит.

• *CryptonDisk* для Windows 95/98/ME/2000/NT 4.0/XP/2003 – прозрачное шифрование логических дисков – создает с помощью устройства "КРИПТОН" или *Crypton Emulator* виртуальный логический диск, шифруемый в автоматическом режиме.

• *Crypton LITE* для Windows 95/98/ME/2000/NT 4.0/XP/2003 – шифрование, ЭЦП с открытым ключом 512 и 1024 бит. Включает в себя: *Crypton API*., *Crypton Emulator*, КРИПТОН-Подпись и КРИПТОН-Шифрование.

• *Crypton Wipe* для Windows-95/98/ME/NT 4.0/2000/XP/2003 – гарантированное уничтожение информации. Для усложнения восстановления удаляемой информации файлы предварительно шифруются.

6.2. СРЕДСТВА КРИПТОГРАФИЧЕСКОЙ ЗАЩИТЫ ИНФОРМАЦИИ "ВЕРБА-О", "ВЕРБА-OW"

Средства криптографической защиты информации (СКЗИ) "Верба-О" (для ОС MS DOS), "Верба-OW" (для ОС Windows) разработаны ЗАО "Московское отделение Пензенского научно-исследовательского электротехнического института" (МО ПНИЭИ) и решают следующие задачи:

• шифрование/расшифрование информации на уровне файлов;

• генерация электронной цифровой подписи (ЭЦП);

• проверка ЭЦП; обнаружение искажений, вносимых злоумышленниками или вирусами в защищаемую информацию.

СКЗИ серии "Верба" поставляются в виде автономного рабочего места или модулей, встраиваемых в программное обеспечение заказчика.

Программные продукты и аппаратно-программные средства "Верба" можно классифицировать следующим образом:

• библиотечные модули, предназначенных для вызова криптографических функций непосредственно из приложения, осуществляющего обработку конфиденциальной информации, обеспечивают шифрование и ЭЦП (программный модуль "*VCrypt*");

• средства криптографической защиты данных пользователя, предназначенные для электронной подписи и шифрования данных пользователей на рабочих местах с возможностью последующего хранения и передачи по каналам связи ("Файловый криптоменеджер");

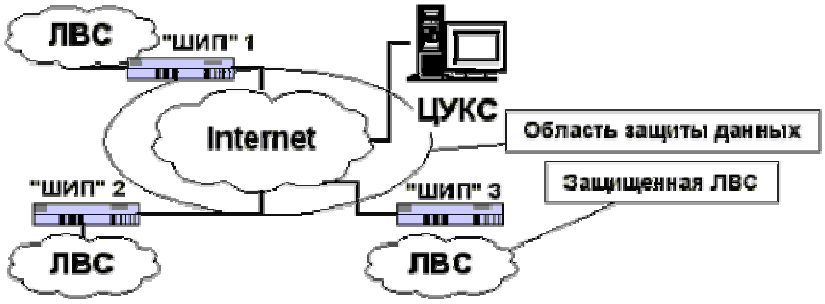


Рис. 6.2. Схема организации виртуальной частной сети с АПК "ШИП" (ЦУКС – Центр управления ключевой системой)

• средства криптографической защиты клиент-серверных технологий, предназначенные для использования в системах типа "клиент-сервер", таких как – доступ к базам данных, системы Банк-Клиент и т.п., и имеющие в своей основе принцип раздельного функционирования систем обработки запросов и систем криптографической защиты информации ("Криптографический сервер");

• средства криптографической защиты каналов связи, предназначенные для защиты информации в каналах связи в режиме *on-line* по протоколам *IP*, X.25, *Fray Relay* и т.д.

• защищенные почтовые технологии, предназначенные как для организации собственных защищенных почтовых систем на базе X.400, так и для организации защищенного документооборота через Internet приложения. Применение шифрования и электронной цифровой подписи самого письма и его вложения позволяет обеспечить конфиденциальность, целостность и авторство передаваемых сообщений (Средства защиты электронной почты "Дионис").

Применение этих средств позволяет создавать виртуальные частные сети (*VPN*) и обеспечивать конфиденциальность передаваемых между ними данных, защищенный выход в Internet, и защищенный *on-line* доступ в частную сеть удаленных (мобильных) пользователей (Аппаратно-программный комплекс "Шип"). Типовая схема организации виртуальной частной сети с АПК "ШИП" приведена на рис. 6.2.

6.3. ПРОГРАММНЫЙ КОМПЛЕКС *VCERT PKI*

Программный продукт *VCERT PKI* является результатом совместной работы компаний ЗАО "МО ПНИЭИ" и ООО "ВАЛИДАТА". Система управления сертификатами *VCERT PKI* – это многокомпонентная система, использующая инфраструктуру открытых ключей для обеспечения конфиденциальности информации, контроля целостности и подтверждения авторства электронных документов на основе использования криптографических процедур, реализованных в соответствии с российскими стандартами и международными рекомендациями.

VCERT PKI условно можно разделить на два компонента: систему управления сертификатами – инфраструктуру открытых ключей (*PKI*) и программный интерфейс к криптографическим функциям для *PKI*-приложений.

Система *VCERT PKI* обеспечивает защиту информации на основе реализации инфраструктуры открытых ключей с использованием международного стандарта X.509, реализована на платформах Windows NT, Windows 95/98.

Программное обеспечение *VCERT PKI* реализовано по модульному принципу, в его состав входят следующие основные программные комплексы и модули:

6.3.1. *VCA* (*VCERT Certification Authority*) – программный комплекс Центр Сертификации (ЦС), предназначенный для создания на основе информации, предоставляемой Центром Регистрации, сертификатов открытых ключей, списков аннулированных сертификатов и их бумажных копий, а также хранения эталонной базы сертификатов и списков аннулированных сертификатов.

6.3.2. *VRA* (*VCERT Registration Authority*) – программный комплекс Центр Регистрации (ЦР), предназначенный для регистрации пользователей и обеспечения взаимодействия пользователя с Центром Сертификации.

6.3.3. *VCS* (*VCERT Certificates Store*) – программный комплекс Справочник Сертификатов, обеспечивающий администрирование справочника сертификатов, формирование служебных сообщений на рабочем месте пользователя, а также генерацию секретных и открытых ключей на рабочем месте пользователя и запись их на ключевые носители.

6.3.4. *VCrypt* – программный модуль реализации криптографических функций и генерации ключевой информации (из состава СКЗИ "Верба-OW").

Цифровая подпись соответствует требованиям ГОСТ Р 34.10–94 "Информационная технология. Криптографическая защита информации. Система электронной цифровой подписи на базе асимметричного криптографического алгоритма". Функция хэширования выполнена в соответствии с требованиями ГОСТ Р 34.11–94 "Информационная технология. Криптографическая защита информации. Функция хэширования", а алгоритм шифрования реализован в соответствии с требованиями ГОСТ 28147–89 "Системы обработки информации. Защита криптографическая".

Длины секретного и открытого ключей электронной цифровой подписи составляют соответственно 256 бит и 512 бит (или 1024 бита), такие же длины имеют секретный и открытый ключи шифрования. Секретные ключи подписи могут храниться на ключевых носителях – дискетах 3.5", носителях *Touch-Memory* или смарт-картах.

Система *VCERT PKI* обеспечивает:

- генерацию и верификацию электронных цифровых подписей под файлом или областью памяти в соответствии с ГОСТ Р34.10–94 и ГОСТ Р34.11–94;
- конфиденциальность и контроль целостности информации посредством ее шифрования и имитозащиты в соответствии с ГОСТ 28147–89;
- регистрацию электронных запросов пользователей на сертификаты открытых ключей подписи;

• формирование электронных сертификатов открытых ключей подписи пользователей.

Клиентское программное обеспечение *VCERT PKI* позволяет пользователям на своих рабочих местах формировать запросы на сертификаты открытых ключей, генерировать секретные и открытые ключи подписи и шифрования, а также получать сообщения о компрометации секретных ключей и информацию из справочника сертификатов.

Инструментарий разработчика дает возможность встраивать в прикладное программное обеспечение криптографические функции генерации/верификации цифровой подписи, шифрования/ расшифрования информации.

6.4. СРЕДСТВО КРИПТОГРАФИЧЕСКОЙ ЗАЩИТЫ ИНФОРМАЦИИ КРИПТОПРО CSP

Средство криптографической защиты информации КриптоПро CSP, разработанное совместно компанией "Крипто-Про" и ФГУП "НТЦ "Атлас", реализовано в соответствии с криптографическим интерфейсом корпорации *Microsoft – CSP* (*Cryptographic Service Provider*) и российскими криптографическими алгоритмами электронной цифровой подписи (ГОСТ Р34.10–94), хэширования (ГОСТ Р34.11–94) и шифрования и имитозащиты данных (ГОСТ 28147–89).

Программный комплекс *Удостоверяющий центр* – КриптоПро УЦ позволяет в полном объеме реализовать инфраструктуру открытых ключей.

В состав КриптоПро УЦ входят следующие компоненты (рис. 6.3): Центр сертификации (ЦС), Центр регистрации (ЦР), АРМ администратора ЦР, АРМ пользователя, программный интерфейс взаимодействия с УЦ.

Центр сертификации – базовый компонент системы, предназначенный для формирования сертификатов открытых ключей пользователей и администраторов *Удостоверяющего центра*, хранения эталонной

Рис. 6.3. Структура средства криптографической защиты информации КриптоПро CSP

базы сертификатов и списков аннулированных сертификатов. Центр сертификации реализован на платформе Microsoft Windows 2000 Server и взаимодействует только с Центром регистрации на базе защищенного сетевого протокола.

Центр регистрации также работает на платформе Microsoft Windows 2000 Server и использует базу данных Microsoft SQL 2000. Взаимодействие пользователей с *Удостоверяющим центром* осуществляется через АРМ пользователя. ЦР является единственной точкой входа (регистрации) поль-

зователей в системе. Только зарегистрированный пользователь может получить сертификат открытого ключа в *Удостоверяющем центре.*

Компонент АРМ Администратора ЦР предназначен для выполнения организационно-технических мероприятий, связанных с регистрацией пользователей, генерацией ключей и сертификатов пользователей и взаимодействием с Центром регистрации. АРМ администратора ЦР функционирует в ОС Microsoft Windows 2000 Professional. АРМ администратора взаимодействует с Центром регистрации в отдельном сегменте локальной сети на базе защищенного сетевого протокола.

АРМ администратора ЦР обеспечивает проверку состояния и обработку запросов пользователей на регистрацию, выдачу и аннулирование сертификатов открытых ключей, просмотр протоколов работы ЦР и поиск информации в базе данных ЦР, относящейся к зарегистрированным пользователям.

АРМ пользователя – это web-приложение, размещенное на сервере ЦР. АРМ пользователя функционирует в ОС Microsoft Windows 95 и выше (браузер – MS IE 5.0 и выше). АРМ пользователя обеспечивает шифрование информации, передаваемой ЦР с использованием протокола *TLS* (протокол *TLS* обеспечивает конфиденциальность и целостность данных при коммутации двух приложений и позволяет приложениям "клиент-сервер" взаимодействовать защищенным способом, предотвращающим перехват информации и подделку сообщений).

К основным функциям АРМ *пользователя* относятся:

- обеспечение взаимодействия пользователя с ЦР;
- заполнение форм запросов на сертификаты;
- выбор типа сертификата;
- проверка состояния запросов на сертификаты и статусасертификатов;
- генерация секретных ключей;
- получение сертификатов.

Программный интерфейс пользователя предназначен для просмотра персональной информации из базы данных ЦР, списка сертификатов, полученных пользователем, и запросов на сертификаты для загрузки, получения и аннулирования сертификатов.

Интеграция российских средств криптографической защиты информации со средствами операционной системы и приложениями, использование инфраструктуры открытых ключей позволяет пользователям и разработчикам обеспечить конфиденциальность, авторство и целостность информации. Применяя средство криптографической защиты информации КриптоПро CSP, пользователи операционной системы MS Windows могут воспользоваться стандартными программными средствами корпорации Microsoft для реализации решений, основанных на инфраструктуре открытых ключей.

6.5. ТЕЛЕФОННЫЙ СКРЕМБЛЕР "ГРОТ"

Телефонный скремблер "Грот" (рис. 6.4) предназначен для защиты конфиденциальной информации. Он обеспечивает шифрования речевого сигнала и защиту факсимильных сообщений, передаваемых по телефонной сети общего применения.

Характеристика работы в канале связи и пользовательские свойства:

- напряжение постоянного тока в абонентской линии: 30…60 В;
- высокая помехоустойчивость;
- автоматическая адаптация к телефонному аппарату абонента, абонентской линии, нелинейности трактов АТС;
- устойчивость работы в реальных телефонных каналах России и стран СНГ, включая междугородные и международные с радиорелейными вставками и любыми видами уплотнения;
- совместимость с любым типом телефонного и факсимильного аппарата, с мини-АТС любого типа, имеющей аналоговый выход;

- работа в линиях, оборудованных системами уплотнения и используемых для охранной сигнализации;
- высокая степень эхокомпенсации;
- низкий уровень шумов в телефонной трубке;
- высокое качество восстановленной речи;
- энергонезависимая память индивидуальных ключей-идентификаторов;
- упрощенный алгоритм ввода индивидуальных ключей-идентификаторов за счет использования электронного блокнота индивидуальных ключей;

Рис. 6.4. Внешний вид скремблера "Грот"

Шифрование:

- метод шифрования – мозаичный: частотные и временные перестановки;
- метод открытого распределения ключей, позволяющий работать без ручного набора ключей;
- общее количество ключевых комбинаций 2×10^{18};
- возможность введения дополнительного семизначного ключа для идентификации абонента;
- высокая степень криптографической защиты за счет наличия дополнительных мастер-ключей, которые устанавливаются по желанию Заказчика.

6.6. АППАРАТУРА КРИПТОГРАФИЧЕСКОЙ ЗАЩИТЫ ИНФОРМАЦИИ Е-20

Аппаратура криптографической защиты речевой и документальной информации с гарантированной Е-20 стойкостью (рис. 6.5) обеспечивает:

- режим телефонного аппарата общего пользования;
- режим криптографической защиты речи;
- режим передачи и криптографической защиты данных с встроенным устройством имитозащиты.

Рис. 6.5. Внешний вид аппаратуры Е-20

Работа в закрытом режиме осуществляется при установленном ключевом носителе *Data Key*.

Аппаратура Е-20 обеспечивает встречную работу с аппаратурой "М-459-1С" по предварительно коммутированным телефонным каналам общего пользования на скоростях 2400, 4800, 9600 бит/с при использовании модемов "УПС-1Ф". Схема возможного соединения абонентов показана на

рис. 6.6.

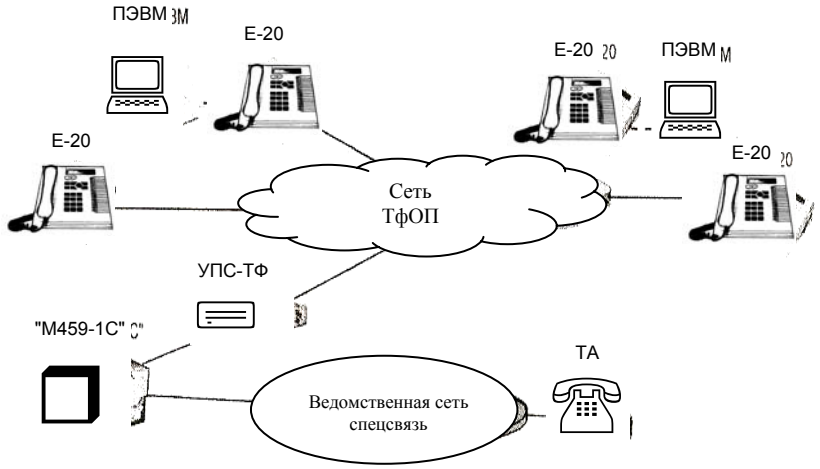


Рис. 6.6. Схема возможного соединения абонентов

Аппаратура "М459-1С" предназначена для криптографической защиты конфиденциальной и секретной телефонной и документальной информации и обеспечивает:

- работу по выделенным и предварительно коммутированным каналам связи совместно с модемом УПС-ТФ в дуплексном режиме на скоростях 2400, 4800, 9600 бит/с;
- работу по предварительно коммутированным телефонным каналам общего пользования для встречной работы с аппаратурой "Е-20";
- передачу/прием документальной информации от ПЭВМ через аппаратуру "Адаптер-ДС".

Внешний вид аппаратуры "М459-1С" показан на рис. 6.7.

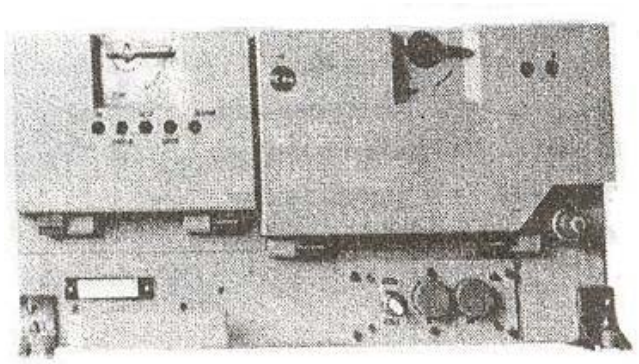


Рис. 6.7. Внешний вид аппаратуры "М459-1С"

Контрольные вопросы

1. Каковы назначения и основные характеристики устройства защиты данных серии КРИПТОН?
2. Каковы назначения и основные характеристики устройства защиты данных серии "Верба"?
3. Каковы назначения и основные характеристики программного комплекса VCERT PKI?
4. Каковы назначение и основные характеристики средства криптографической защиты информации КриптоПро CSP?
5. Опишите особенности защиты информации скремблером "Грот"?
6. Опишите назначение и способ применения аппаратуры криптографической защиты информации Е-20?
7. Опишите назначение и способ применения аппаратуры криптографической защиты информации "М459-1С"?

ЗАКЛЮЧЕНИЕ

Выбор методов криптографической защиты информации для конкретных информационных систем должен быть основан на глубоком анализе слабых и сильных сторон тех или иных методов защиты. Обоснованный выбор той или иной системы защиты должен опираться на какие-то критерии эффективности. К сожалению, до сих пор не разработаны подходящие методики оценки эффективности криптографических систем.

Наиболее простой критерий такой эффективности – вероятность раскрытия ключа или мощность множества ключей. По сути это то же самое, что и криптостойкость. Для ее численной оценки можно использовать также и сложность раскрытия шифра путем перебора всех ключей.

Однако, этот критерий не учитывает других важных требований к криптосистемам:

- невозможность раскрытия или осмысленной модификации информации на основе анализа ее структуры,
- совершенство используемых протоколов защиты,
- минимальный объем используемой ключевой информации,
- минимальная сложность реализации (в количестве машинных операций), ее стоимость,
- высокая оперативность.

Желательно использование некоторых интегральных показателей, учитывающих указанные факторы.

Для учета стоимости, трудоемкости и объема ключевой информации можно использовать удельные показатели – отношение указанных параметров к мощности множества ключей шифра.

Часто более эффективным при выборе и оценке криптографической системы является использование экспертных оценок и имитационное моделирование.

В любом случае выбранный комплекс криптографических методов должен сочетать как удобство, гибкость и оперативность использования, так и надежную защиту от злоумышленников информации, циркулирующей в информационной системе.

СПИСОК ЛИТЕРАТУРЫ

1. Баричев, С.Г. Основы современной криптографии / С.Г. Баричев, В.В. Гончаров, Р.Е. Серов. – М. : Горячая линия – Телеком, 2001. – 120 с.
2. ГОСТ 28147–89. Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования информации. – М. : Госкомитет СССР по стандартам, 1989.
3. ГОСТ Р 34.10–2001. Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронно-цифровой подписи. – М. : Госстандарт России, 2001.
4. ГОСТ Р 34.10–94. Информационная технология. Криптографическая защита информации. Процедуры выработки и проверки электронной цифровой подписи на базе асимметричного криптографического алгоритма. – М. : Госстандарт России, 1994.
5. ГОСТ Р 34.11–94. Информационная технология. Криптографическая защита информации. Функция хэширования. – М. : Госстандарт России, 1994.
6. ГОСТ Р 51583. Защита информации. Порядок создания автоматизированных систем в защищенном исполнении. – М. : Госстандарт России, 1994.
7. Гостехкомиссия России – точка зрения на техническую защиту информации. // JetInfo. 1999. – № 11. – С. 2 – 12 (www.jetinfo.ru).
8. Грушо, А.А. Теоретические основы защиты информации / А.А. Грушо, Е.Е. Тимонина. – М. : Изд-во Агенства "Яхтсмен", 1996. – 192 с.
9. Программно-аппаратные средства обеспечения информационной безопасности. Теоретические основы компьютерной безопасности : учебное пособие для вузов / П.Н. Девянин, О.О. Михальский, Д.И. Правиков, А.Ю. Щербаков. – М. : Радио и связь, 2000. – 192 с.
10. Жельников, В.Г. Криптография от папируса до компьютера / В.Г. Жельников. – М. : ABF, 1996. – 336 с.
11. Завгородний, В.И. Комплексная защита информации в компьютерных системах : учебное пособие для вузов / В.И. Завгородний. – М. : Логос, 2001. – 264 с.
12. Зегжда, Д.П. Основы безопасности информационных систем / Д.П. Зегжда, А.М. Ивашко. – М. : Горячая Линия – Телеком, 2000. – 452 с.
13. Малюк, А.А. Введение в защиту информации в автоматизированных системах / А.А. Малюк, С.В. Пазизин, Н.С. Погожин. – М. : Горячая Линия – Телеком, 2001. – 148 с.
14. Молдовян, А.А. Криптография / А.А. Молдовян, Н.А. Молдовян, Б.Я. Советов. – СПб. : Лань, 2000.
15. Нечаев, В.И. Элементы криптографии (Основы теории защиты информации) : учебное пособие / В.И. Нечаев ; под ред. В.А. Садовничего. – М. : Высшая школа, 1999. – 109 с.
16. Обзор криптотехнологий. // JetInfo. – 2001. – № 3. – 16 с. (www.jetinfo.ru).
17. Руководящий документ. Временное положение по организации разработки, изготовления и эксплуатации программных и технических средств защиты информации от несанкционированного доступа информации в автоматизированных системах и средствах вычислительной техники // Сборник руководящих документов по защите информации от несанкционированного доступа. – М. : ГТК при президенте РФ, 1998. – 120 с.
18. Руководящий документ. Защита от несанкционированного доступа к информации. Термины и определения // Сборник руководящих документов по защите информации от несанкционированного доступа. – М. : ГТК при президенте РФ, 1998. – 120 с.
19. Саломаа, А. Криптография с открытым ключом / А. Саломаа ; пер. с англ. – М. : Мир, 1995. – 318 с.
20. Мао, В. Современная криптография: теория и практика / В. Мао. – СПб. : Вильямс, 2005.

Рис. 1.1. Классификация методов криптографического закрытия информации

Разновидность преобразования	Реализация	
	П	А
Механическое	x	x
Смысловое	x	
По кодовому алфавиту	x	x
По специальным таблицам (словарям)	x	
Гаммирование+ гаммирование	x	x
Перестановка + гаммирование	x	x
Замена + гаммирование	x	x
Замена + перестановка	x	x
С бесконечной гаммой	x	x
С конечной длинной гаммой	x	x
С конечной короткой гаммой	x	x
По особым зависимостям	x	
По правилам алгебры матриц	x	
Усложненная по маршрутам (Гамильтона)	x	
Усложненная по таблице (Вижинера)	x	
Простая	x	
Многоалфавитная многоконтурная	x	
Многоалфавитная одноконтурная моно-фоническая	x	
Многоалфавитная одноконтурная обыкновенная	x	
Простая (одноалфавитная)	x	

Рис. 1.1. Классификация методов криптографического закрытия информации