ВЫСШЕЕ ОБРАЗОВАНИЕ

серия основана в 1996 г.

П.С. ШЕВЧУК
С.В. СОКОЛОВ
С.О. КРАМАРОВ
Е.Н. ТИЩЕНКО
О.Ю. МИТЯСОВА

КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА ИНФОРМАЦИИ

УЧЕБНОЕ ПОСОБИЕ

Под редакцией доктора физико-математических наук,
профессора С.О. Крамарова

Москва
РИОР
ИНФРА-М

УДК 004.056(075.8)
ББК 32.973.202я73
К82

ФЗ
№ 436-ФЗ
Издание не подлежит маркировке
в соответствии с п. 1 ч. 2 ст. 1

Коллектив авторов:
Шевчук П.С. — д-р техн. наук, профессор кафедры МТС СКФ МТУСИ;
Соколов С.В. — д-р техн. наук, профессор кафедры ИТ РГУПС;
Крамаров С.О. — д-р физ.-мат. наук, профессор, директор Института информационных систем Южного университета (ИУБиП);
Тищенко Е.Н. — д-р экон. наук, профессор, зав. кафедрой ИТЗИ РГЭУ (РИНХ);
Митясова О.Ю. — аспирант кафедры ИТПМ Южного университета (ИУБиП).

Рецензенты:
Безуглов Д.А. — д-р техн. наук, профессор;
Таран В.Н. — д-р физ.-мат. наук, профессор.

К82 **Криптографическая защита информации**: учеб. пособие / П.С. Шевчук, С.В. Соколов, С.О. Крамаров [и др.]; под ред. проф. С.О. Крамарова. — М. : РИОР : ИНФРА-М, 2019. — 321 с. — (Высшее образование). — DOI: https://doi.org/10.12737/1716-6

ISBN 978-5-369-01716-6 (РИОР)
ISBN 978-5-16-013274-7 (ИНФРА-М, print)
ISBN 978-5-16-106001-8 (ИНФРА-М, online)

В данном учебном пособии рассмотрены основные понятия и методы криптографии как прикладной науки, тесно связанной с развитием техники и технологии, средств связи и способов передачи информации. Приводятся методы преобразования информации, которые не позволяют извлекать ее из перехватываемых сообщений; современные методы шифрования; вопросы истории криптографии; теория кодирования. Даются примеры защиты информации в ситуациях, связанных с государственной, военной, коммерческой, юридической или врачебной тайной.

Учебное пособие предназначено для студентов бакалавриата, магистратуры, аспирантов, обучающихся по направлениям «Информационная безопасность», «Бизнес информатика», «Инфокоммуникационные технологии и системы связи» по профилю «Защищенные системы и сети связи».

УДК 004.056(075.8)
ББК 32.973.202я73

ISBN 978-5-369-01716-6 (РИОР)
ISBN 978-5-16-013274-7 (ИНФРА-М, print)
ISBN 978-5-16-106001-8 (ИНФРА-М, online)

ГЛАВА 1. ВВЕДЕНИЕ В КРИПТОГРАФИЮ

1.1. ВВЕДЕНИЕ

Каким образом можно направить необходимую информацию нужному адресату тайным образом? Размышляя над этой задачей, нетрудно понять, что возможны несколько вариантов:

1. Использование абсолютно надежного и недоступного другим пользователям канала связи.

2. Сокрытие факта передачи информации при использовании общедоступного канала связи.

3. Передача информации в преобразованном виде по общедоступному каналу связи, с возможностью восстановления только на стороне адресата.

Эти возможности нуждаются в комментариях.

1. Развитие науки и техники на современном этапе не позволяет создать канал связи между адресатами, находящимися на большом расстоянии, для использования его в целях неоднократной передачи информации в большом объеме.

2. Разработка средств и методов, позволяющих скрыть факт передачи информации, является основной целью науки стеганографии.

Еще в глубокой древности были предприняты первые попытки создания стеганографических методов. Например, в некоторых источниках описан способ скрытия письменного сообщения, заключающийся в нанесении текста на обритую кожу головы раба: после того как волосы отрастали, посланца направляли к получателю сообщения. Детективные произведения «подарили» такие способы тайнописи, как нанесение сообщения при помощи химических реактивов в дополнение к обычному, незащищаемому тексту. Получатель сообщения выполняет специальную обработку, после чего скрытое сообщение становится доступным. В некоторых детективных произведениях описан метод «микроточки», заключающийся в записи сообщения на микроскопический носитель с помощью современной техники, и впоследствии такой носитель можно переслать вместе с обычным письмом в заранее обусловленном месте (например, под маркой). Широкое распространение компьютеров позволило создать методы сокрытия защищаемой информации, основанные на ее размещении внутри больших объемов информации.

3. Наука криптография ставит целью разработку методов преобразования, или шифрования, которые позволили бы защитить нужную информацию от несанкционированного доступа.

Криптография относится к прикладным наукам и «берет на вооружение» новейшие достижения таких фундаментальных наук как математика. Следует отметить, что уровень развития техники и технологии, применяемые средства связи и способы передачи информации оказывают влияние на большинство конкретных задач криптографии.

Для ответа на вопрос о предмете криптографии используемые понятия также следует прокомментировать.

Прежде всего следует отметить, что возникновение задачи сокрытия передачи информации связано только с информацией, содержащей тайну и нуждающейся в защите. Такую информацию называют защищаемой, приватной, конфиденциальной, секретной. Некоторые ситуации указанного типа обусловили введение таких специальных понятий, как:

– государственная тайна;
– военная тайна;
– коммерческая тайна;
– юридическая тайна;
– врачебная тайна и т.д.

Далее, говоря о защищаемой информации, будем иметь в виду такие присущие ей признаки, как:

– наличие определенного круга законных пользователей, обладающих правом владеть определенной информацией;

– наличие незаконных пользователей, стремящихся овладеть определенной информацией, чтобы получить от нее пользу и нанести вред законным пользователям.

Далее будет рассмотрена одна из угроз защищаемой информации — угроза разглашения. Существуют и другие, такие как подмена, имитация.

Изобразим ситуацию, в которой возникает задача скрытой передачи информации, схемой, показанной на рис. 1.1.

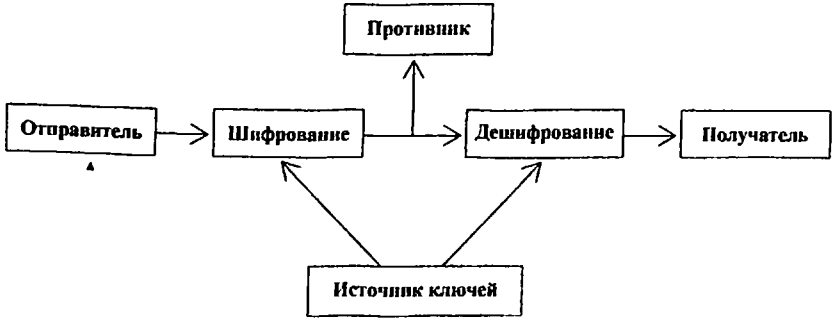


Рис. 1.1. Схема защиты информации

Здесь отправитель и получатель — законные пользователи защищаемой информации, находящиеся на расстоянии и желающие обмениваться информацией с использованием общедоступного канала связи. Противник — это незаконный пользователь, имеющий возможность перехватить передаваемые по каналу связи данные и извлечь из них необходимую информацию. Приведенная формальная схема — примерная модель типичной ситуации, в которой проявляется необходимость применения криптографических методов защиты информации.

Исторически закрепившиеся в криптографии некоторые слова и фразы, взятые из военной отрасли, являются наиболее точным отражением смысла соответствующих криптографических понятий и явлений (противник, атака на шифр и др.). Вместе с тем другая терминология, тоже военная, но имеющая в своей основе понятие кода, уже не применяется в теоретической криптографии. К таким понятиям можно отнести понятия «военно-морские коды», «коды Генерального штаба», «кодовые книги», «кодобозначения» и др. Причиной этого является сформировавшееся обширное научное направление, изучающее и разрабатывающее методы и способы защиты информации в каналах связи от случайных искажений, — теория кодирования. Следует отметить, что сегодня употребление терминов «кодирование» и «шифрование» в качестве синонимов недопустимо. Иными словами, распространенное выражение «кодирование — это разновидность шифрования» сейчас является неверным.

Криптография занимается изучением и разработкой таких средств и методов трансформации данных, которые не дали бы противнику возможности извлечь информацию из передаваемых сообщений в случае их перехвата. Иными словами, по каналу связи передается результат преобразования информации, защищаемой при помощи шифра, и противник должен решить сложную задачу вскрытия шифра, чтобы получить доступ к ней.

Однако имеются и другие пути, по которым противник может получить доступ к защищаемой информации. Кроме перехвата и вскрытия шифра, хорошо известным является агентурный способ, заключающийся в принуждении или склонении к сотрудничеству кого-либо из законных пользователей и в получении доступа к защищаемой информации при помощи этого агента. В данном случае можно констатировать бессилие криптографии. Кроме того, целью противника может быть даже не получение, а уничтожение или модификация защищаемой информации в процессе ее передачи, что является для информации совсем другим типом угроз, отличающимся от перехвата или вскрытия шифра. Поэтому ведется разработка отдельных специфических методов для защиты от таких угроз.

Итак, необходимы различные способы обеспечения безопасности информации, защищающие ее от различных угроз в процессе передачи сообщения от одного законного пользователя к другому. При этом законные пользователи должны учитывать в своей стратегии защиты то обстоятельство, что противник будет искать слабейшее звено в цепи разнотипных звеньев, защищающих информацию, чтобы с наименьшими затратами добраться до нее. Следовательно, нет смысла усиливать какое-либо звено при наличии более слабых («принцип равнопрочности защиты»).

Существует и еще одна важная проблема, о которой не следует забывать. Это проблема соотношения цены информации, затрат на ее добывание и на защиту. Сегодня техника достигла такого уровня развития, что сами средства связи, средства перехвата или защиты информации, их разработка требуют вложения больших средств.

Следующие вопросы, существенно влияющие на выбор подходящих средств защиты (физических, стеганографических, криптографических), необходимо обдумать, прежде чем приступить к защите информации:

– выше ли ценность информации для противника, чем стоимость атаки;

– выше ли ценность информации для ее законного владельца, чем стоимость защиты.

Далее приводится историческая справка, поскольку для некоторых понятий криптографии лучше всего подходит иллюстрация историческими примерами.

1.2. ИСТОРИЯ РАЗВИТИЯ КРИПТОГРАФИИ

Историю криптографии окутывают туманы легенд [1, 46–49, 51]. В наиболее полной книге, содержащей более тысячи страниц, описано большое число дипломатических и военных тайн. Книга увидела свет в 1967 г., а на русском языке доступна в сокращенном виде [36]. Также на русском языке известен фундаментальный труд по истории криптографии в России [33].

Древний Египет. В городе Менет-Хуфу на берегу Нила некий египетский писец почти четыре тысячи лет назад нарисовал иероглифы, которыми описывалась история жизни его господина. Эти надписи сделали того писца родоначальником документально зафиксированной истории криптографии.

При создании той, дошедшей до наших дней и датируемой примерно 1900-м г. до н. э. надписи на гробнице знатного человека по имени Хнумхотеп не применялась тайнопись в том смысле, в каком ее понимают современные исследователи, и не был использован какой-либо полноценный шифр. Лишь отдельные места текста записаны необычными иероглифическими символами вместо более привычных. Многие из таких необычных символов были замечены в двадцати заключительных столбцах, в которых описываются монументы, возведенные Хнумхотепом во славу фараона Аменемхета II. Безымянным писцом была предпринята попытка придания тексту большей важности, (как, например, вместо «в 1863 г.» иногда пишут «в год одна тысяча восемьсот шестьдесят третий от Рождества Христова»). То есть целью было не усложнение текста для прочтения, или тайнопись. Однако это самый древний известный текст, в котором был применен один из существенных элементов шифрования — умышленное преобразование письменных символов.

По мере того как древнеегипетская цивилизация достигала своего расцвета и совершенствовалась письменность, росло число усыпальниц почитаемых умерших, тексты на стенах гробниц подвергались все более изощренным преобразованиям. Практиковалась замена обычной иероглифической формы буквы иной (например, изображение рта анфас заменялось изображением рта в профиль). Вводились в употребление новые иероглифы. В некоторых случаях первым звуком произношения выражалась нужная буква. Иногда произношение двух иероглифов, имеющих похожее изображение, различалось. Применялся и принцип ребуса (как, например, в английском языке для обозначения буквы «В» можно использовать изображение пчелы). Причем иероглифам, со временем становившимся лишь более искусственными и сложными, были присвоены прису-

щие им звуковые значения в результате таких изначально свойственных обычному египетскому письму трансформаций.

Такие тексты, как надгробные надписи, восхвалявшие умерших, гимн в честь Тота, целью которых не было сокрытие смысла текста, содержат измененные иероглифы. Большая часть текста записана повторно в неизмененной форме. Считается, что создатели текстов, используя вышеописанные приемы, преследовали такие цели, как создание особого впечатления, отражение соответствующего времени произношения. Часто целью было просто украсить надписи или похвастаться каллиграфией.

Но постепенно при создании текстов стала возникать одна из самых важных для криптографии цель — обеспечение секретности, — возникшая, как считается, из желания заставить прохожего прочесть эпитафии и тем самым выразить почтение умершим. В некоторых случаях секретность служила созданию эффекта усиления тайны поминальных текстов и, следовательно, их колдовской силы. Писцы нарочно добавляли текстам таинственности, так как к определенному моменту число надгробных надписей выросло настолько, что интерес к ним резко упал, и необходимо было этот интерес возродить. Криптографические знаки должны были служить привлечению внимания читателя, чтобы он задумался и захотел разгадать их смысл. Однако все вышло наоборот, и от «надгробной» криптографии отказались вскоре после ее появления, так как криптографические символы отбивали желание читать каждую из множества надгробных надписей.

Таким образом, криптография зародилась вместе с появлением элемента секретности в преобразовании иероглифов и больше напоминала игру, стремясь задержать разгадку лишь на короткий промежуток времени. А в решении такой головоломки заключался криптоанализ, который был в тот период квазинаукой (так называемая «наивная» криптография). Современный же криптоанализ стал чрезвычайно серьезной областью научных знаний.

В иероглифах Древнего Египта в действительности были такие важные элементы, являющиеся основными атрибутами криптографии, как секретность и преобразование письма. Хотя форма этих элементов была на тот момент несовершенной.

Так зародилась криптология, развивавшаяся в течение 3000 лет отнюдь не поступательно. В одних местах появление криптологии было независимым. Исчезновение этих цивилизаций привело и к забвению достижений криптологии. В других местах криптология сохранилась благодаря памятникам литературы, опираясь на которые, последующие поколения могли делать новые открытия в данной области, хотя развитие не было равномерным и быстрым, так

как сохранено было меньше, чем утеряно. Значительную часть древней истории криптологии можно сравнить с плохо подобранным, разношерстным, составленным из расцветающих, распустившихся и одновременно увядающих цветов букетом.

Древняя Иудея. Неизбежным можно назвать факт наличия шифров (точнее, из-за отсутствия элемента секретности, — предшественников шифров) в Библии. У ее авторов, как и в случае с иероглифами на гробнице Хнумхотепа, не было цели или желания при помощи трансформации скрыть содержание текста. Однако можно увидеть стремление путем изменения текста обрести бессмертие, так как впоследствии тщательно переписанный текст позволил сохранить частицу личности переписчика.

В Библии рассказана история о человеческой руке, возникшей в разгар пира у вавилонского царя Валтасара. После своего появления она написала на стене слова: «мене, текел, фарес» на арамейском языке, родственном древнееврейскому, и означают «исчислил», «взвешен» и «разделено». Эту надпись можно считать древней «криптограммой», а загадка кроется не в значении слов. Нет объяснений тому, почему мудрецам царя не удалось объяснить значение надписи, а пророк Даниил, вызванный Валтасаром, без усилий смог прочесть надпись, дать ее толкование: «мене — исчислил Бог царство твое и положил конец ему; текел — ты взвешен и найден очень легким; фарес — разделено царство твое и отдано мидянам и персам».

Также существовали денежные единицы: мина, текел ($^1/_{60}$ мины), фарес ($^1/_2$ мины). Перечисление их в определенной последовательности является символом крушения Вавилонской империи.

С учетом возможности всех интерпретаций непонятно, почему вавилонским мудрецам оказалось не под силу разгадать смысл послания. Возможно, они расшифровали текст, но не решились передать правителю плохие вести. Возможно, только Даниилу было суждено прочесть надпись. В любом случае, Даниила можно считать одним из первых известных криптоаналитиков. И, согласно Библии, за успешный криптоанализ последовала награда, намного превзошедшая другие более поздние вознаграждения за аналогичные успехи в области дешифрования: «Тогда... облекли Даниила в багряницу, и возложили золотую цепь на шею его, и провозгласили его третьим властелином в царстве».

В 600–500 гг. до н. э. древними иудеями была разработана упорядоченная система криптографии «Атбаш» (известная в России как «тарабарская грамота»), заключавшаяся в постоянной замене при письме одной буквы алфавита другой (например, буква «а» заменяется на «я»).

Древняя Индия. Жители Индии — древней высокоразвитой цивилизации — использовали несколько разновидностей тайнописи. В тексте написанного между 321 и 300 гг. до н. э. классического древнеиндийского трактата, посвященного искусству управления государством, содержатся рекомендации главе шпионской спецслужбы, заключающиеся в использовании тайнописи при назначении заданий агентам. В этом же трактате дается совет дипломатам: «При невозможности беседовать с людьми пусть посол осведомится о происходящем у врага из речей нищих, пьяных, сумасшедших, спящих или из условных знаков, надписей, рисунков в храмах и местах паломничества», т.е. для получения разведывательных данных рекомендуется использование криптоанализа. В тексте не дано никакой информации о конкретных методах чтения тайнописи. Однако то обстоятельство, что автору известно о возможности ее дешифрования, свидетельствует о некотором опыте в области криптоанализа. Более того, первое в истории человечества упоминание о применении криптоанализа в политических целях найдено именно в данном произведении.

Малоизвестным фактом является упоминание в древнеиндийской книге «Кама Сутра» криптографии в качестве одного из 64 искусств, обязательных к изучению.

Древний Китай. Примерно в 500 г. до н. э. китайский ученый Сун Цзы завершил свой труд «Искусство войны». В этой книге были впервые сформулированы основные принципы разведки и контрразведки (в том числе и методы обработки информации).

Древние Греция и Рим. С именем спартанского полководца Лисандра и с шифром «Сцитала» связаны первые сведения о применении шифров в военном деле. Позднее появился «Шифр Цезаря», использовавшийся императором для переписки. Шифр под названием «квадрат Полибия» появился в Древней Греции. Эти шифры нуждаются в более подробном рассмотрении [48].

Шифр «Сцитала» появился в V в. до н. э., во времена войны между Спартой и Афинами, и основой его реализации был жезл цилиндрической формы (сцитала), на который, не допуская появления просветов и нахлестов, виток к витку наматывали узкую папирусную ленту. Шифруемый текст записывали вдоль оси сциталы (на размотанной ленте буквы оказывались в беспорядке, что для непосвященных представлялось непонятным). Чтобы прочесть сообщение, полученную ленту адресат наматывал на такую же сциталу.

Класс шифров, к которым относится, в том числе, и шифр «Сцитала», называют шифрами перестановки, так как трансформация открытого текста заключается в перестановке символов (букв) по определенному алгоритму.

Шифр Цезаря реализуется при помощи замены каждой буквы открытого текста третьей буквой по счету после нее. При этом считают, что алфавит записан по кругу (иными словами, за «я» следует буква «а»). Следует отметить, что возможна замена не только на третью букву, но и на любую другую. Главное, чтобы величина сдвига была известна адресату. Шифр Цезаря относится к классу шифров, называемых шифрами замены.

а	б	в	г	д	е
ё	ж	з	и	й	к
л	и	н	о	п	р
с	т	у	ф	х	ц
ч	ш	щ	ъ	ы	ь
э	ю	я	–	–	–

Рис. 1.2. «Полибианский квадрат» для русского алфавита

Авторство шифра *«Полибианский квадрат»* приписывают греческому писателю Полибию. В основе идеи шифра лежит моноалфавитная подстановка, проводимая с использованием случайно заполненной буквами алфавита таблицы квадратной формы (например, 5 × 5 для греческого алфавита). Каждая буква защищаемого текста заменяется буквой, записанной в квадрате ниже исходной (например, «а» будет заменена на «ё», если используется русский алфавит).

Ни один из упомянутых выше случаев использования тайнописи не содержит подтверждений существования в те древние времена криптоанализа в качестве науки, а лишь указывает на нечасто возникавшие факты дешифрования текста. Пример тому — отдельные разгаданные иероглифические надписи на могильных памятниках в Египте или история с пророком Даниилом. Была только криптография, а научный крипто анализ еще только зарождался. Его не было ни в Египте, ни в Греции и Риме, ни в Индии, ни вплоть до 1400 г.— в Европе.

Древний Восток. Арабами была создана одна из самых развитых цивилизаций из когда-либо известных. Процветала наука, а математика и медицина были на тот момент лучшими. Ремесла тоже получили широкое распространение. Несмотря на то что ислам лишил арабскую культуру живописи и скульптуры, неоспоримы достижения этого народа в литературе. Получили распространение словесные загадки, ребусы, каламбуры, а грамматика имела статус важнейшего учебного предмета, тайнопись же была частью данной дисциплины.

Ранний интерес арабов к криптографии способствовал тому, что ими впервые были описаны методы криптоанализа. Арабским ученым Абу Бакр Ахмед бен-Али бен-Вахшия ан-Набати в 855 г. в «Книге о большом стремлении человека разгадать загадки древней письменности» были описаны некоторые классические шифроалфавиты. Например, для шифрования трактатов по черной магии использовали «дауди» — один из таких шифроалфавитов. Название этот шифр получил от имени израильского царя Давида и был составлен из видоизмененных букв древнееврейского алфавита. Еще один шифр был использован в письме шпиона, направленном регенту Алжира в 1775 г.

В 1412 г. Шехаб Калкашанди завершил обширную 14-томную энциклопедию, написанную для того, чтобы дать систематический обзор всех важных областей знания. В энциклопедии в разделе под общим заголовком «Относительно сокрытия в буквах тайных сообщений», содержащем две части, подробно изложены и достижения арабов в области криптологии на тот момент. В одной части говорится о символических действиях и намеках, другую часть Калкашанди посвятил криптологии и симпатическим чернилам. Впервые в истории шифров в книге описаны и системы перестановки, и системы замены. Кроме того, упоминался шифр с более чем одной заменой букв открытого текста (описан в пятом пункте перечня шифров Калкашанди).

Вероятно, в том, что в многочисленных школах арабских грамматиков много времени посвящали интенсивному и скрупулезному изучению Корана, кроется причина появления в энциклопедии первого в истории описания криптоаналитического исследования шифротекста. Кроме всего прочего, в школах грамматиков занимались подсчетом частот встречаемости слов и делали попытки составить хронологию глав Корана, занимались исследованием фонетики слов для подтверждения арабского происхождения слов или выявления факта заимствования. Развитие лексикографии также оказало влияние на исследования лингвистических закономерностей, приведших к возникновению криптоанализа у арабов. В процессе составления словарей авторы фактически должны были принимать во внимание частоты встречаемости букв и возможность их соседства.

Криптоаналитик обязан владеть языком, на котором написана криптограмма — с этой важной идеи Калкашанди начинает изложение криптоаналитических методов в своей энциклопедии. Следом он дает обширное описание лингвистических характеристик арабского языка, так как это «самый благородный и самый прекрасный из всех языков», который является «одним из наиболее распространенных». Калкашанди перечислил буквы арабского языка, которые нельзя встретить в одном слове или которые редко становятся соседями, а также никогда не встречающиеся в словах буквенные комбинации. Далее, отметив, что «в произведениях, не связанных с Кораном, частота использования может быть иной», автор приводит последовательность букв арабского алфавита, составленную на основе «частоты их использования в арабском языке в свете результатов изучения священного Корана». После этого следует подробное объяснение:

«Если вы хотите прочесть сообщение, которое вы получили в зашифрованном виде, то прежде всего начните подсчет букв, а затем сосчитайте, сколько раз повторяется каждый знак, и подведите итог в каждом отдельном случае. Если изобретатель шифра был очень внимателен и скрыл в сообщении все границы между словами, то первая задача, которая должна быть решена, заключается в нахождении знака, разделяющего слова. Это делается так: вы берете букву и работаете, исходя из предположения, что следующая буква является знаком, делящим слова. И таким образом вы изучаете все сообщение с учетом различных комбинаций букв, из которых могут быть составлены слова... Если получается, тогда все в порядке; если нет, то вы берете следующую по счету букву, и т.д., пока вы не сможете установить знак раздела между словами. Затем нужно найти, какие буквы чаще всего встречаются в сообщении, и сравнить их с образцом частоты встречаемости букв, о котором упоминалось прежде. Когда вы увидите, что одна буква попадается чаще других в данном сообщении, вы предполагаете, что это буква «алиф». Затем вы предполагаете, что следующая по частоте встречаемости будет буквой «лам». Точность вашего предположения должна подтверждаться тем фактом, что в большинстве контекстов буква «лам» следует за буквой «алиф»... Затем первые слова, которые вы попытаетесь разгадать в сообщении, должны состоять из двух букв. Это делается путем оценки наиболее вероятных комбинаций букв до тех пор, пока вы не убедитесь в том, что вы стоите на правильном пути. Тогда вы глядите на их знаки и выписываете их эквиваленты всякий раз, когда они попадаются в сообщении. Нужно применять точно такой же принцип по отношению к трехбуквенным словам этого сообщения, пока вы не убедитесь, что вы на что-то напали. Вы выписываете эквиваленты из всего сообщения. Этот же принцип применяется по отношению к словам, состоящим из четырех и пяти букв, причем метод работы прежний. Всякий раз, когда возникает какое-либо сомнение, нужно высказать два–три предположения или еще больше, и выписать каждое из них, пока оно не подтвердится на основании другого слова».

Приведенное четкое разъяснение Калкашанди снабдил примером дешифрования криптограммы, состоящей из двух стихотворных

строк, зашифрованных с использованием условных символов. Пример дешифрирования завершается замечанием, что несколько букв не встретились в дешифрируемом двустишии и что это как раз те буквы, которые располагаются в конце составленного по частоте встречаемости перечня. Несмотря на это, автор пишет: «Однако это простая случайность: буква может быть поставлена не на то место, которое она должна занимать в вышеупомянутом перечне». Выказав это замечание, Калкашанди демонстрирует наличие обширного опыта в области криптоанализа. Раздел о криптологии завершается анализом второй, довольно длинной криптограммы, которая была приведена, чтобы окончательно расставить все точки над «i».

Несмотря на недостаток информации о воздействии криптографии на мусульманскую культуру или о применении арабами для дешифрования военных и дипломатических текстов криптоаналитических способностей, блестяще продемонстрированных Калкашанди, совершенно ясной оказывается произошедшая деградация. Все достижения и познания были забыты и перестали использоваться на практике, на что указывает эпизод, имевший место позднее.

В 1600 г. марокканским султаном Ахмед аль-Мансур было направлено посольство к английской королеве Елизавете I для заключения союза против Испании. Во главе посольства был министр Абдель Вахид ибн Масуд ибн Мухаммед Анун. Неизвестным путем в руки одного араба попала депеша, отправленная Ануном на родину и зашифрованная простым шифром замены. Об этом эпизоде говорит записка:

«Хвала Аллаху! Относительно письма министра Абдель Вахид ибн Масуд ибн Мухаммед Ануна...

Я нашел письмо, написанное его рукой, в котором он с помощью тайных знаков изложил некоторые сведения, предназначенные для нашего покровителя Ахмеда аль-Мансура. Эти сведения касаются султанши христиан (да покарает их Аллах!), которая жила в стране под названием Лондон... С того момента как это письмо попало ко мне, я постоянно время от времени изучал содержавшиеся в нем знаки. Прошло примерно 15 лет, пока не наступило то время, когда Аллах позволил мне понять эти знаки, хотя никто не обучал меня этому...».

Возможно, получивший депешу араб не испытывал недостатка ума. Но, не имея представления о наследии своего народа в области криптоанализа, на задачу, которую Калкашанди решил бы за несколько часов, он потратил 15 лет.

Средневековая Европа. Вплоть до начала эпохи Возрождения европейская криптография находилась в состоянии застоя, а применявшиеся шифры были максимально простыми. Например, использовались шифры, состоявшие в вертикальной или зеркальной записи текста, в замене гласных точками, использовании иностранных алфавитов (например, древнееврейского и армянского), замене каждой буквы исходного текста следующей за ней. В немалой степени это было обусловлено распространением среди людей того времени убежденности в том, что криптография и криптоанализ имеют в своей основе черную магию или даже являются ее разновидностями.

С возникновением формализованных шифросистем, относительно устойчивых против ручного криптоанализа, связывают начало **этапа формальной криптографии (конец XV — начало XX вв.)**, совпавшее с эпохой Возрождения. В этот период возрос спрос на надежные способы защиты информации, так как наука и торговли переживали в своем развитии подъем. Исследования Леона Батиста Альберти, итальянского архитектора, в числе первых предложившего многоалфавитную подстановку, оказали большое влияние на развитие европейской криптографии на данном этапе. На рис. 1.3 показан принцип оригинального шифра замены, предложенный Альберти. Для реализации шифрования необходимы два концентрических круга. На внешней и внутренней окружности располагаются буквы алфавита и шифротекста. Буквы шифроалфавита можно смещать на любое число позиций, причем буквы алфавита необязательно должны располагаться последовательно: АБВГ... ЭЮЯ (их можно располагать и произвольно, например АЭВЮГ...).

В этом принципе шифрования была впервые реализована идея повышения стойкости шифрования при помощи смены последовательности символов шифроалфавита и его сдвига относительно алфавита открытого текста, а сочинение Альберти «Трактат о шифре», созданное в 1466 г., признано первой научной работой в области криптологии.

Труд «Полиграфия», написанный в 1508 г. немецким аббатом Иоганном Трисемусом, — одна из первых печатных работ, в которой автор сформулировал и обобщил все известные к тому моменту алгоритмы шифрования. Трисемусу также принадлежат способ заполнения полибианского квадрата, состоявший в заполнении первых его позиций запоминаемым ключевым словом и записи неиспользованных букв алфавита в остальные ячейки квадрата, а также способ шифрования биграмм, т.е. пар букв.

Известным математиком Джироламо Кардано, которого считают «отцом» теории вероятностей и математической статистики, в 1566 г. была опубликована работа, положившая начало научной криптологии. В этой работе содержалось описание системы шифрования, позднее получившей название «решетка Кардано».

Рис. 1.3. Шифр замены [43]

В 1586 г. был предложен шифр на основе нескольких алфавитов (полиалфавитный). Шифрование реализуется путем присвоения символу исходного текста на основе соответствующего алфавита и буквенного ключа. Для облегчения процедуры часто строят специальную таблицу. Автором данного шифра стал дипломат Блез Виженер.

Франция XVI в., кроме того, оставила в истории криптографии шифры короля Генриха IV и Ришелье. Английский ученый сэр Фрэнсис Бэкон в XVII в. спроектировал собственное шифровальное устройство. По идее ученого, каждой букве алфавита могли соответствовать пять вариантов шифровки, т.е. предлагалось представление букв при помощи пятизначного двоичного кода: А — 00001, Б — 00010 и т.д. Несмотря на относительно слабую стойкость, через три столетия эта идея способствовала развитию электрической и электронной связи на основе кодов Морзе, Бодо, телеграфных кодов.

Шифр Плейфейера был открыт в начале XIX в. и представляет собой простой, но стойкий способ многоалфавитной замены, или подстановки биграмм. Чарльз Уитстон, предложив шифрование «двойным квадратом», усовершенствовал шифр Плейфейера. Ручной криптоанализ этих двух шифров вызывал трудности, поэтому их применяли вплоть до начала Первой мировой войны.

Голландский ученый Огюст Керкгоффс в XIX в. сформулировал остающееся актуальным и сейчас требование к криптографическим системам, состоящее в том, что секретность шифров должна быть основана не на секретности алгоритма. Секретным должен быть ключ.

В 1790 г. Томас Джефферсон, будущий президент США, создал механическую машину, ставшую одной из первых роторных криптосистем. В этой машине была реализована многоалфавитная подстановка при помощи вариации взаиморасположения вращающихся роторов, каждый из которых осуществляет запрограммированную подстановку.

Роторные криптосистемы, получившие практическое распространение только в начале XX в., позволили повысить криптостойкость и способствовали механизации и автоматизации процесса шифрования.

В Германии 1917 г. Эдвард Хеберн разработал машину *Enigma*, которую позже усовершенствовал Артур Кирх. *Enigma* стала одной из первых практически используемых криптосистем. Во время Второй мировой войны активно применялись, помимо немецкой машины *Enigma*, устройства *Sigaba* (США), *Typex* (Великобритания), *Red. Orange* и *Purple* (Япония). Роторные системы достаточно просто позволяли реализовать очень стойкие шифры. Поэтому их можно считать вершиной формальной криптографии. Только в начале 1940-х гг., с появлением ЭВМ стали возможны успешные криптоатаки на роторные системы.

Научно-техническая революция и ее влияние на криптографию

Телеграф. Проблема оперативной передачи сообщений на большие расстояния встала перед людьми еще в древние времена. Огни костров, дымовые сигналы, барабаны «там-там» и другие примитивные способы связи уже требовали разработки условных кодов, позволявших представлять информацию в виде, пригодном для передачи по линии связи. По территории империи древних римлян функционировали более 3000 вышек для передачи световых сигналов. В 1794 г. Клод Шапп создал эффективную и получившую широкое распространение во Франции систему передачи информации между Парижем и Лиллем, названную «воздушный телеграф», с семафорами в качестве основного элемента. В обеспечении связи между кораблями нашли применение «флажковые коды».

Ознаменовав наступление эры новых оперативных средств и способов связи, Сэмюэл Морзе в 1844 г. при помощи специальной азбуки для кодирования букв, получившей название «Азбука Морзе», передал по проводному буквопечатающему телеграфу (рис. 1.4) первую в мире телеграмму: «Вот что сотворил Бог!». Экзотические замены букв замысловатыми знаками и другие «старые» шифры оказались принципиально неприемлемыми и непригодными для использования в телеграфных линиях связи.

Юристом и компаньоном Сэмюэля Морзе Френсисом Смитом в 1845 г. был опубликован коммерческий код под названием «Словарь для тайной корреспонденции; приспособлен для применения на электромагнитном телеграфе Морзе» для уменьшения длины и стоимости телеграмм. Безопасность предлагалось обеспечивать при помощи легко реализуемого в телеграфных линиях связи кода с перешифровкой. Возможность использования в телеграфной связи учитывалась в процессе дальнейшего развития криптографии и механических шифровальных устройств.

Рис. 1.4. Пишущий телеграфный аппарат Морзе, выпущенный заводами «Сименс и Гальске» в России [49]

Соблюдение исторической справедливости требует упоминания и имени российского подданного барона Павла Львовича Шиллинга. Этот выдающийся ученый и изобретатель в октябре 1832 г. публично продемонстрировал работу первого практически пригодного электромагнитного телеграфа, в основу работы которого был положен эффект отклонения магнитной стрелки в результате действия на нее электромагнитного поля, возникающего вокруг электрических проводов. Аппараты для передачи и приема были соединены состоящим из восьми проводов кабелем. Передача одной буквы часто требовала нажатия трех–четырех клавиш одновременно, так как каждый провод активировался определенной клавишей. На приеме каждый проводник был соединен с соответствующим («своим») электромагнитом, над которым висела магнитная стрелка, поворачивавшаяся, когда по проводу проходил ток, т.е. каждая буква кодировалась набором нажатий клавиш, а восстанавливалась в соответствии с набором состояний стрелок над электромагнитами.

Несмотря на это, буквопечатающий электромагнитный телеграф Морзе получил широкое распространение и был удобен в практической реализации.

Определяемая числом букв передаваемого сообщения стоимость телеграфной передачи была высокой (особенно остро эта проблема проявилась после прокладки трансатлантического кабеля США — Европа в 1866 г.). Поэтому начались разработки эффективных методов «сжатия» информации. Создавали несекретные телеграфные коды, предполагающие замену букв, слов, фраз короткими буквенно-цифровыми единицами передаваемого текста. В случае необходимости при передаче секретных сообщений такое несекретное кодирование должно было дополняться шифрами, т.е. осуществлялось шифрование кодированного сообщения.

При передаче сообщений по телеграфным линиям связи неизбежны их искажения. Это обусловило появление помехоустойчивого кодирования, заключающегося в устранении искажений с помощью вводимой в передаваемое сообщение избыточной информации. Нередкое получение адресатами текста с искаженным под воздействием помех содержанием вызывало немало неприятных ситуаций и способствовало повышению интереса к такому свойству шифров как помехоустойчивость. Уместно дополнить рассказ о практике использования «азбуки Морзе» историческими примерами.

Потеря соответствующей в «азбуке Морзе» букве «Е» точки «•» может трансформировать французский глагол *citerons* («мы укажем») в слово *citrons* («лимоны»). Увеличение пробела при кодировании буквы М, которой соответствует комбинация «——», может дать в результате биграмму ТТ (букве Т соответствует тире «–»). Слово «*baneful*» («губительный»), имеющее в азбуке Морзе вид «–••• •– –• • ••–• ••– •–••», может превратиться в слово «*dutiful*» («обязательный»), имеющее код «–•• ••– – •• ••–• ••– •–••». В результате подобных искажений могут получиться сообщения, имеющие далекий от оригинала смысл, что способно привести к дезинформации приемной стороны и стать причиной серьезных негативных последствий. В 1887 г. некий торговец шерстью из США направил телеграмму, в которой агенту торговца было дано указание о продаже большого объема шерсти. Сообщение было защищено с помощью секретного кода, в котором шифрообозначения букв также имели вид азбуки Морзе. При передаче телеграммы в результате искажений слово «продай» трансформировалось в слово «купи». Естественно, указание было выполнено. Торговец, понесший убытки на сумму более 20 000 долл., подал иск в суд. Решением суда телеграфную компанию обязали возместить стоимость (примерно один доллар) искаженной телеграммы.

Вследствие появления такого рода искажений возникла необходимость разработки защитных мер. Применялся «двухбуквенный дифференциал», предполагавший возможность отличия ключевых слов телеграммы не менее чем на две буквы. Это способствовало возникновению большого числа неологизмов и выработке принявшего вид жаргонных кодов телеграфно-кодового языка. Часто осуществляли дублирование наиболее важных мест сообщений, из-за чего расходы на связь росли. Расширялось применение помехоустойчивых кодов, позволявших обнаруживать и устранять искажения. Это также способствовало повышению расходов на связь.

К точности передачи шифрованных телеграмм предъявлялись особые повышенные требования, поэтому телеграфные компании назначали стоимость их передачи выше. Работники телеграфных

компаний утверждали, что по сравнению с передачей обычных «осмысленных» сообщений тщательная и побуквенная передача «нечитаемых» текстов снижала эффективность их работы. Ответом на это стали сделанные пользователями шифров попытки придания «осмысленного вида» (хотя бы на уровне имеющихся в шифртексте «слов») шифрованному (кодированному) сообщению. В свете указанных событий в Лондоне в 1889 г. провели специальную конференцию, которую посвятили толкованию понятия «шифрованная телеграмма». Конференцией в Париже в 1890 г. был введен в обращение официальный словарь кодового языка, вызвавший множество протестов, так как он содержал лишь «читаемые слова», запретив, по существу, передачу секретных (кодированных) сообщений. От этого единого словаря отказались после проведения Конференции в Лондоне в 1903 г. Было разрешено применение искусственных слов длиной не более 10 символов с условием, что сообщения будут состоять из «читаемых и произносимых слов». На Мадридской конференции 1932 г. были сняты все ограничения относительно кодирования и шифрования.

В 1904 г. в Англии был опубликован словарь кодобозначений Уайтло. Согласно утверждению автора, в словаре содержалось примерно 400 млн пятибуквенных произносимых слов (например, «*FREAN*», «*LUFFA*», «*LOZOI*», «*FORAB*»). Допускалось соединение слов, чтобы обозначить новое словообразование. Идея была поддержана и развита Э. Бентли в 1905 г., а результатом стал дошедший до наших дней универсальный пятибуквенный код, предполагавший разбиение шифртекстов на пятибуквенные сочетания. Коды, в основе которых лежал принцип использования словарных величин, были вытеснены кодами с «нормированной» длиной.

Возникли секретные коды, используемые различными промышленными или коммерческими компаниями (автомобильные компании, банки, биржи и др.) для собственных нужд. Это способствовало появлению особых «криптографических групп», которые профессионально занимались составлением для пользователей секретных кодов (кодовых книг, сравнимых по объему со словарем английского языка) с учетом специфического языка клиентов («под заказ»). Такие коды превратились в обычный дорогостоящий товар, обусловив появление рынка торговли кодами. Дошли до наших дней разработанные для говорящих на разных языках корреспондентов разноязычные и многоязычные кодовые книги, а сигнал «*SOS*» («*Save our souls*», «Спасите наши души») воспринимается как сигнал бедствия жителями всех стран.

Современные коды, с учетом богатства лексики открытого языка, позволяют «сжимать» передаваемую по техническим каналам связи

информацию более чем в 10 раз. Задача сжатия информации, учитывая не только экономические аспекты, но и требования оперативности, и в наши дни остается актуальной, породив научное направление, занимающее одно из первых мест в теории связи и получившее название «математическая теория кодирования».

Появление телеграфа породило в криптографии проблемы, связанные с соединением передающей телеграфные сообщения аппаратуры и шифрующих устройств. Это способствовало повышению требований к быстродействию шифрования, так как в таких условиях оно должно было производиться в диктуемом телеграфным аппаратом темпе.

Значительный рост числа передаваемых при помощи телеграфной связи сообщений (в том числе и секретных) вызвал необходимость создания новых алгоритмов шифрования, позволяющих легко сменить ключ, и дал процессу развития криптографии новый стимул. Осуществлять перехват телеграфных сообщений в техническом смысле оказалось труднее, чем перехватить пересылаемые курьером или через почтамт документы, а вербовка телеграфиста оказалась недостаточно эффективной. Началась разработка техники для тайного съема информации с телеграфных линий связи и методов обнаружения фактов такого съема. Несмотря на это, в XIX в. для абонентов, располагающихся на удаленном расстоянии, эффективным методом обеспечения оперативной связи стал именно телеграф.

Применявшееся в XIX в. предварительное шифрование сообщений, когда отправитель самостоятельно зашифровывал передаваемое сообщение (с удовлетворяющим требованиям телеграфной передачи шифротекстом) и относил на телеграф, в XX в. часто оказывалось неприемлемым. Возникла необходимость в методах «линейной» передачи шифрованных сообщений, с непосредственным внедрением шифратора аппаратуру передачи сообщений. Реализация таких методов сделало передачу шифрованного и несекретного сообщений в техническом смысле почти неотличимыми. Дэвид Кан, американский историк, утверждал: «свой современный вид шифровальное дело получило благодаря телеграфу». Вероятно, он прав.

Радио, появившееся в 1895 г. благодаря русскому ученому А.С. Попову (рис. 1.5), также оказало существенное влияние на процесс развития криптографии.

Для защиты информации велась разработка все новых и новых шифров. Возникали сети засекреченной связи со значительным числом абонентов, породив проблему смены и эффективного распределения ключей между ними, а также проблему повышенного риска компрометации абонентов. В случае изъятия у абонента ключевой информации вся сеть могла оказаться под контролем противника.

Рис. 1.5. Радиоаппарат Попова [49]

К тому времени в техническом смысле перехват радиосообщений уже не был принципиальной проблемой, а работа дешифровщиков стала проще. Поэтому имели место случаи отказа защищающейся стороной от использования более эффективной радиосвязи в пользу проводного телеграфа (а иногда, и специальных курьеров). По сравнению с проводной связью, радиоканал порождал значительно более серьезные искажения, что привело к повышению требований к помехоустойчивости шифров. Поскольку засекреченные сети включали большое число абонентов, не исключалась возможность внедрения противника в качестве абонента, который мог бы от имени других абонентов давать необходимого содержания шифрованные распоряжения и указания. Это подтвердившееся в дальнейшем предположение вызывало серьезные негативные последствия для «лояльных» абонентов сети и сделало актуальным вопрос об имитостойкости сети засекреченной связи.

Рис. 1.6. Комната, оборудованная радио Маркони [49]

На фоне развития радиосвязи получили импульс к развитию стеганографические методы защиты информации. Стала возможной передача секретной информации на фоне «невинной» передачи (таким «фоном» могло стать музыкальное произведение). Результатом широкого развития радиосвязи стала «радиоэлектронная война». Противник для нанесения ущерба сети связи соответствующей техникой зашумления мог создать мощные радиопомехи, что стало причиной предъявления новых повышенных требований к помехоустойчивости шифров. Для отвлечения сил и средств противника на перехват и анализ «псевдосообщений» стали использовать «псевдосети» (ложные сети связи). Возникло явление «радионгр».

Мобильность и более низкая (по сравнению с проводной) стоимость радиосвязи дали толчок к активизации информационного обмена между военными подразделениями и позволили наладить связь с такими подвижными объектами, как автомобили, самолеты, корабли.

Рис. 1.7. Береговой центр морской радиосвязи [49]

Сравнительная простота перехвата секретных зашифрованных радиосообщений в условиях резкого увеличения объемов их передач подтолкнула дешифровщиков к принесшей определенные результаты идее, что отдельную перехваченную криптограмму необходимо анализировать вместе со всем остальным массивом перехваченной информации, в контексте которого криптограмма была обнаружена. Дэвид Кан справедливо отметил: «Телеграф создал современное шифровальное дело, радио — современный криптоанализ».

Число абонентов сетей засекреченной связи росло скачками, в результате число технических работников шифрслужб (шифровальщиков) также неуклонно росло. Когда в сети секретной связи было небольшое число корреспондентов, шифрование выполняли сами абоненты. Но когда произошло усложнение шифров и расширение сетей секретной связи, возникла необходимость привлечения операторов-шифровальщиков, справлявшихся значительно лучше абонента,

специально не подготовленного для такой работы, с задачами подготовки и передачи секретных сообщений. Кроме того, стало нецелесообразным возлагать на абонента сложные для него технические функции передачи информации и отвлекать от основной его деятельности. В результате подготовка технических сотрудников спецслужб — шифровальщиков — приобрела массовый характер наряду с разработкой и внедрением шифраторов. Это были различных механические (позднее, и электромеханические) приборы, предназначенные для шифрования и дешифрования сообщений.

Лавинообразный рост числа передаваемых по каналам связи шифрованных сообщений стал причиной увеличения числа допускаемых при шифровании ошибок, которые быстро научились выявлять дешифровщики. В криптоанализе родилось новое направление, заключающееся поиска ошибок при шифровании с целью их использования при дешифровании. К таким типичным ошибкам можно отнести повторное использование закрытого ключа, повторное шифрование открытого текста другим ключом и др. В ответ стали вестись разработки «сверхнадежных» шифраторов и технических приспособлений для «блокирования» ошибок шифрования.

В технической сфере можно было наблюдать аналогичную картину. Требования обеспечения оперативности и массовости секретной связи с неизбежностью вызвали одновременное появление шифраторов. Однако технические отказы аппаратуры становились причиной передачи «слабо зашифрованных» сообщений, создавая возможность их использования противной стороной. Такие случавшиеся нередко отказы становились причиной дезорганизации самой системы закрытой связи. Однако победа «машинного» века криптографии, несмотря ни на что (даже несмотря на недовольство пользователей из-за отказов «машинных системам»), бесспорна.

Технические возможности, появившиеся в процессе развития радиосвязи, позволяли достаточно точно вычислить источник передачи секретного сообщения, т.е. стала возможной идентификация абонентов. Однако анализ сети засекреченной связи, ее абонентов и их иерархии может дать много полезной информации даже без дешифрования. Появилось дошедшее до наших дней направление, занимающееся исследованием состояния сетей защищенной связи противника (интенсивность, адресация, длины передаваемых сообщений, динамика изменения данных параметров во времени) для получения разведывательной информации. Во время Второй мировой войны немцами по резкому увеличению числа шифрованных сообщений BBC на континент был сделан верный вывод об открытии «второго фронта» и о подготовке союзниками скорой высадки десанта во Франции. Однако была допущена серьезная ошибка относительно предполагаемого места начала операции.

Постепенно телеграфное и радиокодирование с целью защиты информации вытеснялось применением более мобильных и дешевых шифров. Смена громоздких и не совсем удобных секретных кодовых книг вызывала серьезные проблемы. Кроме того, они могли попасть, и попадали в руки противника. Секретное кодирование стало использоваться реже, но полностью не исчезло, так как стало применяться вместе с шифрованием. Подобное сочетание и дошло до наших дней, доказав свою эффективность. Следует подчеркнуть, что компрометация шифра не требует смены всех кодовых книг. Смены ключей шифра в данном случае вполне достаточно. Кроме того, применение шифров и смысловое содержание открытого текста не связаны между собой, а чувствительность кодов к лексике и словарному запасу языка общения вызывала необходимость обновлять кодовые книги при появлении новых терминов и понятий.

Рис. 1.8. Настенный телефонный аппарат производства L.M. Ericsson & Co, Стокгольм [49]

Телефон. Первый проводной телефон был публично продемонстрирован и запатентован в 1876 г. американцем Александром Беллом. На звание изобретателя телефона претендуют несколько ученых (то же касается и радио). Однако вопросы приоритета различных изобретателей находятся за рамками данного пособия. Телефон достаточно быстро получил широкое признание в мире. Для решения проблемы передачи по телефону конфиденциальной информации в 1881 г. другой главный электротехник Капитолия американец Джеймс Роджерс предложил свой телефонный шифратор: «Мое изобретение состоит в том, что сообщение... посылается по двум (или более) цепям поочередными импульсами в быстрой последователь-

ности..., так, что тот, кто подключается лишь к одной из цепей, может принимать лишь отдельные неразборчивые сигналы... Две (или более) линии, по которым передаются сигналы речи, могут быть проведены к оконечной станции на значительном расстоянии друг от друга, что исключает возможность для пытающегося подслушать... подключиться одновременно к обеим линиям». Несмотря на то что чисто технические причины помешали изобретению получить широкое распространение, подобная идея быстрой смены несущей частоты в широком диапазоне по сложному закону была реализована в XX веке в СИЧ-передачах (СИЧ можно расшифровать как «Скачкообразное Изменение Частоты»).

Чтобы снизить влияние неизбежных помех, использовали «классический способ», заключающийся в передаче букв в виде имен или коротких слов («Анна» значит «А», «Борис» значит «Б», и т.д). Предварительное шифрование текста с последующей передачей его в шифрованном виде по телефону также стали применять для передачи конфиденциальной информации. Использовались достаточно простые шифры (например, квадрат Полибия), так как абоненты часто не имели возможности использовать какую-либо аппаратуру шифрования. Это в значительной мере снизило оперативность связи, заставив абонентов шифровать только отдельные, особо «секретные» слова и передавать открытой речью остальной текст. Коды, сохраняя указанный недостаток, также нередко применялись вместо шифрования.

Широкое распространение получили «условный язык», жаргонные выражения, иносказания, намеки и т.д. Предполагалось, что они будут безошибочно расшифрованы лишь адресатом, оставшись для противника непонятными. Например, слова «болеть», «больница» и «доктор» в специально разработанных для агентурной связи жаргонных кодах могли означать соответственно «арест» (или «заключение под стражу»), «тюрьма» и «контрразведка». Тогда «невинное» сообщение «Майкл заболел. Вчера был доктор и посоветовал ему лечиться в больнице» адресат расшифрует как «Майкл арестован контрразведкой. Ему грозит заключение в тюрьму».

Были случаи использования собеседниками известного им иностранного языка, предположительно неизвестного противнику. Как пример можно привести использование американцами в обеих мировых войнах XX в. практически неизвестных в воюющих странах языков некоторых малочисленных индейских племен, чьи специально подготовленные представители работали на узлах связи.

Отказ передачи конфиденциальной информации по телефону (к которому прибегают и сейчас) также нередко становился средством ее защиты. В таком случае часто говорят: «Это не телефонный разговор. Поговорим при встрече».

В начале XX в. для преодоления существенных недостатков вышеописанных методов защиты речевой информации (слабая стойкость, низкая оперативность связи и отсутствие возможности массового применения) ученые стали вести исследования по реализации автоматического засекречивания речевого сигнала аппаратурой.

В 1900 г. датский инженер Вальдемар Поульсен предложил разбивать речевой сигнал на части и передавать эти сегменты в обратном направлении, т.е. осуществлять «временную инверсию». В 1918 г. Эрик Тигерстедт предложил метод «временных перестановок», заключавшийся в разбиении речевого сигнала на сегменты перестановкой их во времени. В 1920 г. метод «временных перестановок» был усовершенствован русским ученым Михаилом Александровичем Бонч-Бруевичем. Была введена кадровая структура преобразований, предполагавшая особенные перестановки для каждых N сегментов. Англичанин Хоу-Гольд в 1922 г. для засекречивания радиотелефонной связи предложил использование синхронного изменения несущей частоты передатчика и настройки приемника. Были реализованы и другие подобные изобретения в этот период. А в 20–30-х гг. XX в. стали массово применяться технологии автоматического засекречивания речевого сигнала, и началось серийное производство такой аппаратуры.

Развитие проводной телеграфной и телефонной связи и отсутствие постоянной возможности тайного подключения поставили задачу реализации эффективного снятия информации с линий связи. Английский капитан Р. Стэнли, до этого преподававший в Белфастском университете, в 1915 г. создал аппарат для индуктивного перехвата информации. Аппарат, созданный Стэнли, позволял осуществлять перехват с расстояния до 100 м, позже удалось сконструировать прибор для перехвата с расстояния до трех километров. Подобные аппараты позднее появились и в Германии.

Современная криптография. В период научной криптографии (1930–60-е гг.), став его основной чертой, появились криптосистемы, в которых криптостойкость имеет строгое математическое обоснование. В это время завершилось формирование таких составляющих научную основу криптологии разделов математики, как теория вероятностей и математическая статистика, теория чисел, общая алгебра. Получили широкое развитие теория алгоритмов, теория информации, кибернетика. Клод Шеннон в 1949 г. опубликовал работу «Теория связи в секретных системах», которую можно назвать своеобразным водоразделом. Ученым были введены понятия «рассеивание» и «перемешивание», обоснована возможность создания крип-

тосистем, сколь угодно стойких. Под криптографию и криптоанализ учеными была подведена научная база, и с этого времени стали говорить о **криптологии** (от греческого «kryptos» — «скрытый, тайный» и «logos» — «слово, сообщение, высказывание, речь»). Криптология — наука, изучающая способы и методы преобразования информации в целях обеспечения ее секретности.

К созданию стойких (если сравнивать с роторными криптосистемами) блочных шифров ведущие криптографические школы вплотную приблизились в 1960-х гг. Правда, практическая реализация таких шифров возможна только в виде цифровых электронных устройств.

Компьютерная криптография (с 1970-х гг.) появилась вместе с вычислительными средствами, имеющими производительность, достаточную для создания таких криптосистем, в которых одновременно можно обеспечить высокую скорость шифрования и на несколько порядков более высокую, чем у «ручных» и «механических» шифров, криптостойкость.

Блочные шифры стали первым классом криптосистем, получившим практическое применение, когда появились мощные и компактные вычислительные средства. В 70-е гг. в США был разработан принятый в 1978 г. стандарт шифрования DES. Хорст Фейстель, один из авторов DES, описал модель блочных шифров. Модели более стойких симметричных криптосистем (в том числе отечественный стандарт шифрования ГОСТ 28147–89) были построены на основе модели Фейстеля.

Появление DES обогатило и криптоанализ. Новые его виды, получившие возможность реализации только после появления мощных вычислительных систем, были созданы именно для осуществления атак на американский алгоритм. Речь идет о таких видах криптоанализа как линейный, дифференциальный и др.

С появлением в середине 70-х гг. XX в. асимметричных криптосистем отпала потребность в передаче секретного ключа, и в современной криптографии произошел настоящий прорыв. В работе «Новые направления в современной криптографии», опубликованной в 1976 г., Уитфилд Диффи и Мартин Хеллман впервые описали процесс обмена шифрованной информацией без необходимости передачи секретного ключа и сформулировали основные принципы осуществления такого обмена. Поэтому работу Диффи и Хеллмана можно считать «отправной точкой» в данном направлении. Ральф Меркли подошел к идее асимметричных криптосистем независимо от Диффи и Хеллмана.

Рон Ривест, Ади Шамир и Леонард Адлеман несколькими годами позже создали систему RSA, являющуюся первой практической асимметричной криптосистемой, в основе стойкости которой лежит

разложение на простые множители больших простых чисел (их факторизации).

После появления симметричной криптографии сформировались несколько новых прикладных направлений. В качестве примера можно привести электронную цифровую подпись (ЭЦП) и электронные деньги.

В 1980–90-е гг. актуальность задачи совершенствования симметричных криптосистем не исчерпалась. На этот период приходится разработка нефейстелевских шифров (SAFER, RC6 и др.), а также формирование совершенно новых направлений криптографии (вероятностное шифрование, квантовая криптография и др.), практическую ценность которых еще предстоит определить. Результатом открытого международного конкурса 2000 г. стало принятие AES в качестве нового национального стандарта шифрования США.

В то же время нет никаких оснований говорить о беспроблемности сегодняшней ситуации внедрения криптографических решений.

Стандартизация алгоритмов и протоколов идет медленно, причем до самого недавнего времени политический расклад был таков, что на международном уровне стандартизовались лишь протоколы, а подстановка в них конкретных алгоритмов приводила к неопределенности итоговой оценки надежности и к несовместимости «стандартных» реализаций.

Большая часть современного системного и коммуникационного программного обеспечения включает криптографическую функциональность, но, как правило, она остается в лучшем случае незадействованной (более 90% информации, проходящей по сетям Интернета, передается в открытом виде), а в худшем — используется так, что порождает лишь иллюзию безопасности.

Серьезной проблемой остается неразвитость нотариальной системы заверения ключей — к сожалению, в законодательстве об эквивалентности цифровой подписи собственноручной большинство стран «заложились» на неадекватную, рискованную и бесперспективную иерархическую архитектуру сертификации.

Немалый урон репутации криптографии наносят шарлатаны, периодически пытающиеся продать «криптографические» решения задач, не имеющих отношения к криптографии, а иногда и вовсе неразрешимых (например, так называемая «защита контента» — аудио, видео или текста, правообладатели которых пытаются технологическими средствами ограничить законные права своей аудитории).

Из-за инерции технологической инфраструктуры финансовой отрасли пока крайне медленно внедряются финансово-криптографические решения, потенциально способные «расшить» многие уз-

кие места сетевого бизнеса (анонимную оплату услуг в реальном времени, мини- и микроплатежи).

В России развитие гражданской криптологии и криптографии началось лишь после падения советской власти (еще в середине восьмидесятых попытка издания переводного справочника по криптографии привела к аресту большей части тиража). В значительной степени оно все еще сдерживается недостатком гражданских кадров и попытками вмешательства военизированных организаций. Тем не менее, за последние годы было опубликовано несколько серьезных книг, ряд разработчиков получил неплохие результаты, а также рынок и признание, в том числе за рубежом. Все больше российских программистов работает в международных криптографических проектах. С 1999 г. ассоциация «РусКрипто» проводит ежегодные конференции под тем же названием.

1.3. ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ

Криптология — наука, изучающая математические методы защиты информации путем ее преобразования [1, 41].

Криптография и **криптоанализ** — основные направления криптологии.

Методы преобразования информации, позволяющие обеспечить ее конфиденциальность и аутентичность, изучаются в рамках криптографии.

Конфиденциальность — невозможность извлечения информации из преобразованного массива в отсутствие дополнительной информации (такой информацией является секретный ключ).

Аутентичность — подлинность авторства и целостность информации.

В рамках криптоанализа объединены математические методы, позволяющие нарушить конфиденциальность и (или) аутентичность данных в отсутствие дополнительной информации в виде секретных ключей.

Выделяются следующие крупные разделы современной криптографии:

– симметричные криптосистемы, использующие для шифрования и дешифрования один ключ;

– криптосистемы, имеющие открытый ключ;

– системы электронной подписи (ЭП);

– управление криптографическими ключами (выработка, распределение между пользователями, введение в аппаратуру, контроль использования, смена и уничтожение ключей).

Основными направлениями применения криптографических методов являются:

– осуществление процесса передачи конфиденциальной информации по каналам связи (например, электронная почта);

– обеспечение проверки подлинности передаваемой информации;

– обеспечение хранения в зашифрованном виде информации (документов или баз данных) на различных носителях.

В качестве информации, которая подлежит шифрованию и расшифрованию и к которой можно добавить электронную подпись, понимается текст или сообщение, составленные на основе некоторого алфавита.

Алфавит — конечное множество символов, применяемых в целях кодирования информации.

Текст (сообщение) — упорядоченный набор символов, составленный при помощи элементов алфавита.

Современные информационные системы работают со следующими алфавитами:

– 32 (исключая «ё») русские буквы и пробел (алфавит Z33);

– символы, из которых составлены стандартные коды ASCII и КОИ-8 (алфавит Z256);

– цифры 0 и 1 (двоичный алфавит Z2);

– цифры от 0 до 7 (восьмеричный алфавит);

– цифры от 0 до 9 и латинские буквы от A до F (шестнадцатеричный алфавит).

Между кодами и шифрами, использовавшимися задолго до появления ЭВМ, с теоретической точки зрения нет четкого различия. Однако современная практика делает достаточно четкое разграничение данных понятий. Коды манипулируют лингвистическими элементами и разделяют текст, подвергаемый шифрованию, на смысловые элементы (слова и слоги), а в шифре всегда можно выделить **алгоритм** и **ключ**. При этом, используя алгоритм и сравнительно короткий ключ, можно зашифровать сколь угодно большой текст.

Шифр — некоторое число заданных алгоритмом криптографического преобразования и являющихся обратимыми преобразований множества открытых данных во множество зашифрованных данных.

Существует и более строгое определение.

Шифр — некоторое проиндексированное элементами из множества ключей число инъективных отображений множества открытых данных на множество шифрованных данных.

Как видно из приведенных выше определений, говоря о шифре (криптографической системе), следует иметь в виду семейство являющихся обратимыми преобразований открытого текста в шифрованный, которое можно обозначить T_k, и членам которого может

быть однозначно сопоставлен ключ (k). Следовательно, T_k можно однозначно определить при помощи соответствующих алгоритма и значения ключа k.

Ключом (обычно это последовательность букв алфавита) называют определенное засекреченное состояние, в котором для обеспечения выбора одного из числа всех возможных для криптографического преобразования информации алгоритма вариантов должны находиться определенные параметры данного алгоритма. Соблюдение секретности ключа должно обеспечивать невозможность воссоздания исходной информации по зашифрованной. В качестве примеров можно привести диаметр сциталы и величину сдвига относительно букв открытого текста, которые являются ключами соответственно в шифре «Сцитала» и в шифрах типа шифра Цезаря [46].

Пространством ключей K называют значения, которые может принимать ключ.

Пароль, являясь секретной последовательностью букв алфавита, служит для аутентификации субъектов (не для шифрования). Поэтому понятия «ключ» и «пароль» необходимо различать.

Выделяют **симметричные** криптосистемы (в которых для шифрования и для дешифрования необходим один и тот же ключ) и **асимметричные**.

Асимметричные системы (системы с открытым ключом) построены на применении математически связанных между собой открытого (публичного) и закрытого (секретного) ключей. Данные подвергаются шифрованию при помощи доступного всем желающим открытого ключа. Расшифровать зашифрованные подобным образом данные возможно лишь при наличии закрытого ключа, известного только адресату.

Процессы обработки информации, основное содержание которых заключается в выработке и распределение ключей среди пользователей, связаны с такими терминами как «распределение ключей» и «управление ключами».

Электронная цифровая подпись (ЭЦП) — это криптографическое преобразование информации, присоединяемое к ней и позволяющее осуществить последующую проверку авторства сообщения и его целостность.

Шифрование — термин, описывающий процесс зашифрования или расшифрования данных, используемый также в качестве синонима термина «зашифрование».

Шифрование (зашифрование, зашифровывание) — преобразование при помощи шифра открытых данных в зашифрованные.

Как уже было отмечено выше, использование терминов «кодирование» и «код» вместо соответственно терминов «шифрование» и

«шифр» не является корректным, потому что термин «кодирование» обычно означает представление информации в виде знаков, являющихся буквами некоторого алфавита.

Расшифрование (расшифровывание) — преобразование при помощи шифра зашифрованных данных в открытые.

Часто термин «открытые данные» заменяется терминами «открытый текст» или «исходный текст». Вместо термина «зашифрованные данные» часто используют понятие «шифрованный текст».

Дешифрование — преобразования закрытых данных в открытые при отсутствии ключа и, возможно, при отсутствии сведений об алгоритме шифрования, в данном случае имеются в виду методами криптоанализа.

Криптостойкость — свойство шифра противостоять дешифрованию, определяемое необходимым для дешифрования периодом времени.

Понятие криптостойкости является ключевым для криптографии и легким в понимании. До сих пор не получены необходимые математические результаты для решения проблемы получения строгих доказуемых оценок стойкости в отношении каждого конкретного шифра. Поэтому проблема до сих пор не решена, и оценка стойкости определенного шифра только на основе результатов всевозможных попыток его вскрытия находится в зависимости от квалификации осуществляющих атаку на шифр криптоаналитиков.

Эта процедура, важным подготовительным этапом которой является определение предполагаемых возможностей для атаки на шифр, получила название проверки стойкости и всегда содержит предположения об условиях, целях и возможностях противника, потому что возможности для атаки в определенной мере не зависят от криптографии, могут иметь вид некоторой внешней подсказки и существенным образом влияют на стойкость шифра.

При проведении оценки криптостойкости шифра обычно предполагают, что противнику известны некоторые характеристики открытых текстов (общая тематика, стиль, стандарты, форматы) и сам шифр и доступны возможности для его предварительного изучения.

Более специфические примеры возможностей противника следующие:

– возможность перехвата всех шифрованных сообщений без возможности получения соответствующих им открытых текстов;

– возможность перехвата всех шифрованных сообщений и получения соответствующих им открытых текстов;

– возможность получения доступа к шифру без получения доступа к ключам (т.е. возможность зашифровывания и расшифрования любой информации).

Многие века специалисты продолжали спорить о стойкости шифров и о возможности создания шифра, который был бы абсолютно стойким. Далее приведены три характерных высказывания относительно данного вопроса.

Английский математик Чарльз Беббидж (XIX в.): «Всякий человек, даже если он не знаком с техникой вскрытия шифров, твердо считает, что сможет изобрести абсолютно стойкий шифр, и чем более умен и образован этот человек, тем более твердо это убеждение. Я сам разделял эту уверенность в течение многих лет».

Отец кибернетики Норберт Винер: «Любой шифр может быть вскрыт, если только в этом есть настоятельная необходимость, и информация, которую предполагается получить, стоит затраченных средств, усилий и времени...».

Автор шифра *PGP* Ф. Зиммерманн: «Каждый, кто думает, что изобрел непробиваемую схему шифрования, — или невероятно редкий гений, или просто наивен и неопытен...»; «Каждый программист воображает себя криптографом, что ведет к распространению исключительно плохого криптообеспечения...».

Гамма шифра — псевдослучайная двоичная последовательность, получаемая на основе заданного алгоритма и применяемая в процессе преобразования открытых данных в зашифрованные данные.

Гаммирование — процесс применения гаммы шифра к открытым данным(наложение ее по определенному закону на открытые данные).

Имитозащита — обеспечение защиты в целях предотвращения навязывания ложной информации при помощи добавления **имитовставки** (последовательности данных, имеющей фиксированную длину и созданной на основе определенного правила из открытых данных и ключа).

Криптографической защитой называют защиту информации, основанную на использовании криптографического преобразования, т.е. преобразования данных при помощи шифрования и (или) выработки имитовставки.

Под **синхропосылкой** понимают являющиеся общедоступными и открытыми параметры, передаваемые алгоритму криптографического преобразования в качестве исходных.

Уравнение зашифровывания (расшифровывания) — соотношение, описывающее процесс получения в результате заданных криптографическим алгоритмом преобразований зашифрованной (открытой) информации из открытой (зашифрованной) [50].

ГЛАВА 2. ПОНЯТИЕ О ТРАДИЦИОННЫХ МЕТОДАХ ШИФРОВАНИЯ

2.1. МОДЕЛЬ ТРАДИЦИОННОГО ШИФРОВАНИЯ

На рис. 2.1 показана схема процесса традиционного шифрования.

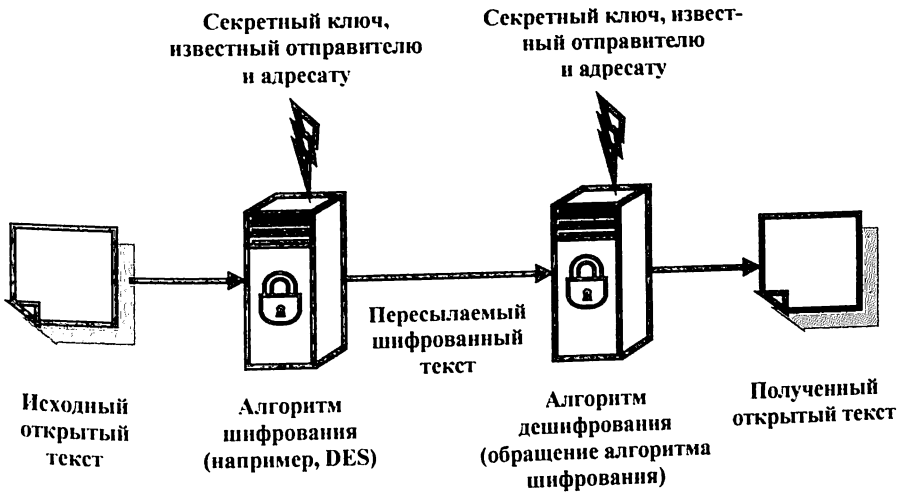


Рис. 2.1. Модель традиционного шифрования

Исходное осмысленное сообщение, обычно называемое открытым текстом, преобразуется в бессмысленную на вид последовательность символов, называемую шифрованным текстом. Процесс шифрования состоит в использовании алгоритма и некоторого ключа. Ключ — это значение, не зависящее от открытого текста. Результат, достигаемый при выполнении алгоритма, зависит от применяемого при этом ключа. Изменение ключа приводит к изменению результата выполнения алгоритма.

Полученный шифрованный текст можно пересылать получателю. После получения шифрованного текста адресатом текст можно снова преобразовать в открытый с помощью соответствующего алгоритма дешифрования и того же ключа, который применяется при шифровании.

Надежность традиционного шифрования зависит от нескольких факторов. Во-первых, алгоритм шифрования должен быть достаточ-

но сложным, чтобы невозможно было расшифровать сообщение при наличии только шифрованного текста. Во-вторых, основным фактором надежности традиционного шифрования является секретность ключа, в то время как сам алгоритм может быть и несекретным. Поэтому предполагается, что должна быть обеспечена практическая невозможность расшифровки сообщения на основе знания шифрованного текста, даже если известен алгоритм шифрования / дешифрования. Другими словами, не требуется обеспечивать секретность алгоритма — достаточно обеспечить секретность ключа.

Именно эта особенность схемы традиционного шифрования обусловливает ее широкую популярность и признание. Отсутствие необходимости хранить в секрете алгоритм дает производителям возможность реализовать алгоритмы шифрования данных в виде дешевых общедоступных микросхем, которыми оснащены сегодня многие современные системы. При использовании традиционного шифрования основная проблема обеспечения безопасности заключается в надежном сохранении секретности ключа.

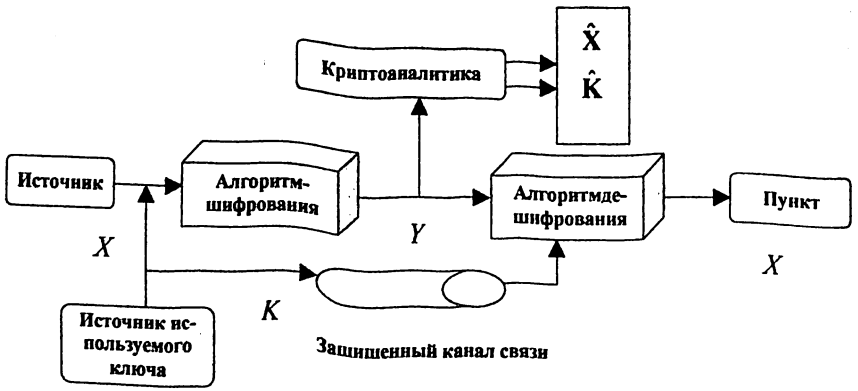


Рис. 2.2. Функциональная схема традиционного шифрования

С помощью приведенного рис. 2.2 рассмотрим основные элементы схемы традиционного шифрования подробнее. Источник создает сообщение в виде открытого текста $\mathbf{X} = [X_1, X_2,\ldots, X_M]$. Элементами X_i открытого текста **X** являются символы некоторого конечного алфавита. Традиционно использовался алфавит, состоящий из 26 прописных букв английского языка, но сегодня все чаще применяется двоичный алфавит {0,1}. Для шифрования генерируется ключ в форме $\mathbf{K} = [K_1, K_2,\ldots, K_j]$. Если ключ генерируется там же, где и само сообщение, то ключ тоже необходимо переправить получателю сообщения по каким-то секретным каналам. Другим решением может быть создание ключа третьей стороной, которая должна защищенным способом обеспечить доставку ключа как отправителю, так и получателю сообщения.

При наличии в качестве исходных данных сообщения **X** и ключа шифрования **K** с помощью алгоритма шифрования формируется шифрованный текст $\mathbf{Y} = [Y_1, Y_2,\ldots, Y_N]$. Это можно записать в виде формулы

$$\mathbf{Y} = \mathbf{EK(X)}.$$

Эта запись означает, что **Y** получается путем применения алгоритма шифрования **E** к открытому тексту **X** при использовании ключа **K**.

Предполагаемый получатель сообщения, располагая ключом **K**, должен иметь возможность выполнить обратное преобразование:

$$\mathbf{X} = \mathbf{DK(Y)}.$$

Противник, обладающий возможностью ознакомиться с **Y**, но не имеющий доступа ни к **K**, ни к **X**, может попытаться восстановить **X** или **K**, или сразу оба эти объекта. При этом подразумевается, что противник знает и алгоритм шифрования (**E**), и алгоритм дешифрования (**D**). Если противник заинтересован распознать только одно конкретное сообщение, ему следует сосредоточить свои усилия на восстановлении **X** путем построения вероятно соответствующего исходному открытого текста. Однако чаще противник будет заинтересован в получении возможности читать и все последующие сообщения. В этом случае все основные усилия должны быть сосредоточены на восстановлении **K** путем построения вероятно соответствующего исходному ключу $\hat{\mathbf{K}}$.

Во многих случаях предполагается наличие у криптоаналитика только шифрованного текста, а алгоритм шифрования ему неизвестен. Однако все же необходимо учитывать возможность наличия у противника информации об используемом алгоритме. В такой ситуации простой перебор всех возможных вариантов ключей становится одним из вероятных подходов к дешифрованию [54].

Несмотря на это, пространство потенциальных ключей большого размера делает нереальным такой подход, заставляя противника применять анализ шифрованного сообщения для выявления его различных статистических особенностей. Поэтому у противника должны иметься некоторые общие предположения относительно содержимого дешифрируемого сообщения (например, что в тексте использован английский или французский язык, что информация представляет собой исполняемый файл MS DOS или исходный код на

языке Java, что в файле содержатся данные относительно банковских счетов).

Таблица 2.1

Типы криптоанализа шифрованного сообщения [54]

Тип криптоанализа	*Данные, известные криптоаналитику*
Анализ только шифрованного текста	– Алгоритм шифрования – Подлежащий расшифровке шифрованный текст
Анализ с известным открытым текстом	– Алгоритм шифрования – Подлежащий расшифровке шифрованный текст – Один или несколько соответствующих фрагментов открытого и шифрованного текста, созданных с одним и тем же секретным ключом
Анализ с избранным открытым текстом	– Алгоритм шифрования – Подлежащий расшифровке шифрованный текст – Выбранный криптоаналитиком открытый текст и соответствующий шифрованный текст, созданный с помощью секретного ключа
Анализ с избранным шифрованным текстом	– Алгоритм шифрования – Подлежащий расшифровке шифрованный текст – Выбранный криптоаналитиком шифрованный текст и соответствующий открытый текст, расшифрованный с помощью секретного ключа
Анализ с избранным текстом	– Алгоритм шифрования – Подлежащий расшифровке шифрованный текст – Выбранный криптоаналитиком открытый текст и соответствующий шифрованный текст, созданный с помощью секретного ключа – Выбранный криптоаналитиком шифрованный текст и соответствующий открытый текст, расшифрованный с помощью секретного ключа

От попыток дешифрования при минимальном объеме информации в распоряжении у противника (имеется только шифрованное сообщение) защититься проще всего. Но чаще криптоаналитик обладает гораздо бóльшим объемом информации, имея возможность перехватить одно или более сообщений открытого текста либо зная об обязательном наличии в сообщении тех или иных наборов символов (идентичная последовательность символов в начале файла в формате Postscript, наличие в файле стандартного заголовка или идентификатора сообщения). Наличие такой информации дает противнику воз-

можность, используя логические умозаключения и знание особенностей преобразований известного открытого текста, восстановить секретный ключ.

Задача криптоанализа с известным открытым текстом и задача криптоанализа с вероятно известными словами являются родственными. При отсутствии информации о теме или содержании сообщения противнику будет нелегко определить направление поиска. Однако известная специфичная информация о содержимом текста повышает вероятность раскрытия части сообщения (сведения о наличии в файле информации о банковских счетах или о наличии в исходном коде разработанной некоторой компанией X программы информации об авторских правах могут дать противнику подсказку о расположении определенных частей сообщения).

Иногда криптоаналитик имеет возможность так или иначе обеспечить себе доступ к сгенерировавшей сообщение системе и обработать с ее помощью определенное сообщение, осуществив криптоанализ с избранным открытым текстом. Дифференциальный криптоанализ имеет в своей основе подобную стратегию. В общем случае, наличие у криптоаналитика возможности выбора сообщения для шифрования позволяет (при удачном подборе текстов) оправдать его надежды относительно разгадки ключа.

Упоминаемые в табл. 2.1 две дополнительные разновидности криптоанализа (анализ с избранным шифрованным текстом и анализ с избранным текстом) применяются реже, но также теоретически возможно их использование криптоаналитиком.

Анализ только зашифрованного сообщения позволяет взломать лишь относительно слабые алгоритмы. В общем случае разработка любого криптоалгоритма ведется с учетом возможности атак с использованием известного открытого текста [54].

Здесь уместно дать два определения. Схема шифрования называется безусловно защищенной (абсолютно стойкой), если порожденный по этой схеме шифрованный текст не содержит информацию, достаточную для однозначного восстановления соответствующего открытого текста, какой бы большой по объему шифрованный текст ни имелся у противника. Это означает, что независимо от того, сколько времени затратит противник на расшифровку, ему не удастся расшифровать шифрованный текст просто потому, что в шифрованном тексте нет информации, требуемой для восстановления открытого текста. Среди алгоритмов шифрования абсолютно стойких нет, за исключением так называемой ленты однократного использования (или схемы с одноразовым блокнотом). Таким образом, максимум, чего может ожидать пользователь, это получение относительно надежного алгоритма, удовлетворяющего следующим требованиям:

1. Стоимость взлома шифра превышает стоимость расшифрованной информации.

2. Время, которое требуется для того, чтобы взломать шифр, превышает время, в течение которого информация актуальна.

Схема шифрования называется защищенной по вычислениям, если она соответствует обоим указанным критериям. Единственная проблема здесь в том, что весьма непросто количественно оценить те усилия, которые необходимы для успешного криптоанализа шифрованного текста, созданного на основе конкретной схемы шифрования.

В качестве первого приближения оценим время, необходимое для простого перебора всех возможных вариантов ключей до тех пор, пока из шифрованного текста не будет получен какой-нибудь постижимый открытый текст. В среднем для этого нужно перебрать примерно половину всех вариантов ключей. В табл. 2.2 показано, сколько времени уйдет на перебор в зависимости от размеров пространства возможных ключей. В первых строках таблицы приведены результаты для трех двоичных ключей разной длины. Ключи размером 56 битов используются в алгоритме DES. В последней строке таблицы приведены оценки для так называемых подстановочных кодов, являющихся 26-символьными ключами, представляющими собой перестановки последовательности из 26 символов. Среднее время, необходимое для нахождения ключа, зависит от длины ключа.

Таблица 2.2

Среднее время анализа при простом переборе ключей

Длина ключа, бит	*Число различных ключей*	*Необходимое время при скорости 1 шифрование/мс*	*Необходимое время при скорости 10^6 шифрований/мс*
32	$2^{32} = 4{,}3 \times 10^9$	2^{31} мс = 35,8 мин	2,15 мс
56	$2^{56} = 7{,}2 \times 10^{16}$	2^{55} мс = 1142 года	10,01 ч
128	$2^{128} = 3{,}4 \times 10^{38}$	2^{127} мс = $5{,}4 \times 10^{24}$ лет	$5{,}4 \times 10^{18}$ лет
26 символов (перестановка)	$26! = 4 \times 10^{26}$	2×10^{26} мс = $6{,}4 \times 10^{15}$ лет	$6{,}4 \times 10^9$ лет

Приведенные в табл. 2.2 величины интервалов времени рассчитывались в предположении, что на одну попытку расшифровки уходит 1 мс (что является вполне реальным для современных компьютеров). При использовании машин с параллельной архитектурой можно достичь гораздо лучших показателей, поэтому в последнем

столбце табл. 2.2 приведены результаты некоторой гипотетической системы, которая в состоянии проверить 1миллион ключей за 1 мс. Как видно из таблицы, при такой производительности алгоритм DES уже нельзя считать защищенным по вычислениям.

При разработке любых форм традиционного криптоанализа для схем традиционного шифрования учитывается факт сохранения при шифровании некоторых особенностей, характерных для структуры исходного сообщения, и их проявления в шифрованном сообщении.

2.2. ТРЕБОВАНИЯ К КРИПТОГРАФИЧЕСКИМ СИСТЕМАМ

Осуществление процесса криптографического закрытия данных возможно как программным, так и аппаратным способом.

Отличие аппаратной реализации заключается в существенно большей стоимости и в таких преимуществах, как простота, обеспечение высокой производительности, защищенности. В более практичной программной реализации допускается гибкость в использовании.

Независимо от того, какой способ реализации выбран в конкретной криптографической системе защиты информации, для современных криптосистем были разработаны следующие общепринятые требования [41]:

- стойкость шифра противостоять криптоанализу должна обеспечивать возможность его вскрытия только решением задачи полного перебора ключей, находящегося за пределами возможностей современных компьютеров (здесь учитывается и организация сетевых вычислений) или требующего дорогостоящих вычислительных систем;
- именно секретность ключа (не алгоритма) должна обеспечивать криптостойкость (отсюда выделились криптосистемы общего использования с доступным потенциальному противнику алгоритмом и имеющие секретный алгоритм криптосистемы ограниченного использования);
- только наличие ключа должно позволить прочесть зашифрованное сообщение;
- стойкость шифра даже при наличии у противника достаточно большого количества исходных и соответствующих им зашифрованных данных;
- вид зашифрованного текста должен существенно меняться при незначительном изменении ключа или исходного текста;
- должна сохраняться неизменность структурных элементов алгоритм шифрования;

- объем шифртекста должен быть сопоставим с объемом исходной информацию, должно обеспечиваться полное и надежное сокрытие в шифрованном тексте вводимых в сообщение в процессе шифрования дополнительных битов;
- возникновение ошибок при шифровании не должно становиться причиной искажений и потерь информации;
- простые и легко устанавливаемые зависимости между последовательно применяемыми в процессе шифрования ключами должны отсутствовать;
- каждым возможным ключом должна обеспечиваться равная криптостойкость;
- процесс шифрования не должен занимать большой объем времени;
- стоимость шифрования и стоимость закрываемой информации должны согласовываться.

2.3. КРАТКИЕ СВЕДЕНИЯ О КРИПТОАНАЛИЗЕ

Главное действующее лицо в процессе криптоанализа — криптоаналитик (противник, нарушитель), в качестве которого выступает лицо (группа лиц), целью которого (которых) является извлечение (прочтение) или изменение сообщений, защищенных криптографическими методами.

В основу математических или иных моделей закладываются следующие допущения [41].

1. Противнику известен алгоритм шифрования (или создания электронной подписи) и особенности конкретной реализации данного алгоритма, но неизвестен секретный ключ.

2. У противника имеются все зашифрованные сообщения и он может обладать доступом к некоторому числу исходных сообщений с известными зашифрованными текстами, соответствующими данным сообщениям.

3. Противнику доступны человеческие, вычислительные, временные и иные ресурсы, объем которых оправдан потенциальной ценностью добываемой в процессе криптоанализа информации.

Криптоатака (атака на шифр) заключается попытке прочтения или изменения зашифрованной информации или в попытке определения секретного ключа. В случае удачи криптоатака становится **взломом.**

Криптостойкость в таком случае определяется как свойство шифра противостоять криптоатаке (расшифрованию при неизвестном ключе) и становится важнейшим параметром каждой криптосистемы, в качестве показателя которого можно использовать:

– число всех возможных ключей или вероятность подбора ключа с использованием определенных ресурсов за определенное время;

– число операций или время, которое при условии наличия определенных ресурсов требуется для взлома шифра с определенной вероятностью;

– значение стоимости поиска ключа или дешифрования исходного сообщения.

Во всех указанных показателях должен быть учтен, кроме всего прочего, уровень возможной криптоатаки.

Однако на эффективность защиты информации влияет не только криптостойкость шифра, но и множество других обстоятельств, таких как особенности реализации криптосистемы (в виде устройства или программы), человеческий фактор. Влияние (в том числе и подкуп) на человека, обладающего необходимой информацией, может обойтись противнику гораздо дешевле использования суперкомпьютера для дешифрования.

Такие математические науки как алгебра, теория чисел, теория вероятностей и математическая статистика, теория алгоритмов стали опорой современного криптоанализа, методы которого можно связать со следующими четырьмя направлениями.

1. Статистический криптоанализ занимается исследованием возможностей взлома криптосистем с использованием статистических закономерностей, существующих в исходных и зашифрованных сообщениях. В настоящее время применение методов статистического анализа осложняется наличием предварительного сжатия информации, превращающего сообщение в случайную последовательность символов, или применением псевдослучайных последовательностей большой длины в случае гаммирования.

2. Алгебраический криптоанализ исследует наличие в криптоалгоритмах математически слабых мест (например, в 1997 г. был существенно упрощен криптоанализ алгоритмов на основе эллиптических кривых после выявления особого класса ключей).

3. Дифференциальный (или разностный) криптоанализ исследует зависимость изменения шифрованного сообщения от преобразования изменения исходной информации (впервые был применен С. Мерфи, а позже был улучшен для атаки на DES Э. Бихамом и А. Шамиром).

4. Линейный криптоанализ, предложенный Мицуру Мацуи, основывается на нахождении линейной аппроксимации между исходной и шифрованной информацией и был впервые использован при взломе DES (его применение в реальных криптосистемах, как и применение дифференциального криптоанализа, ограничено анализом отдельных блоков криптопреобразований).

Практика атак на криптосистемы (например, регулярно устраиваемые RSA *Data Security* конкурсы) показывает, что в большей степени урон криптосистемам наносит небрежность в реализации, а главенствующую роль среди методов продолжает играть «лобовая» атака, или «проба на ключ».

В зависимости от объема имеющейся в распоряжении криптоаналитика информации различают несколько уровней криптоатаки по нарастанию сложности.

1. Уровню КА1 соответствует атака по шифрованному тексту, когда противник имеет доступ ко всем или некоторым зашифрованным сообщениям.

2. Уровню КА2 соответствует атака по паре «исходный текст — шифрованный текст», когда противник имеет в своем распоряжении, кроме зашифрованных сообщений, соответствующие им исходные тексты.

3. Уровень КА3 соответствует атаке по выбранной паре «исходный текст — шифрованный текст», когда у противника есть возможность выбора исходного текста, для которого впоследствии формируется шифрованный текст, вычисляется ключ.

Проектирование всех современных криптосистем осуществляется таким образом, чтобы обеспечить достаточную стойкость даже к атакам уровня КА3, когда противник, по сути, имеет доступ к шифрующему устройству.

2.4. КЛАССИФИКАЦИЯ МЕТОДОВ КРИПТОГРАФИЧЕСКОГО ЗАКРЫТИЯ ИНФОРМАЦИИ

Далее приведена классификация большого числа методов криптографического закрытия (шифрования) информации по различным признакам (тип ключей, размер блока информации и другие). В настоящий момент известны следующие классификации криптоалгоритмов [41].

1. Криптоалгоритмы по типу ключей:
– симметричные;
– асимметричные.
2. Шифры по размеру блока информации:
– потоковые;
– блочные.
3. Методы по характеру производимых на информацию воздействий:
– замена (перестановка);
– подстановка;
– аналитические;
– гаммирование (аддитивные);
– комбинированные.
4. Шифрование по применяемым лингвистическим методикам
– смысловое;
– символьное;
– комбинированное.

Стеганография, сжатие / расширение, рассечение / разнесение также используются для закрытия информации.

ГЛАВА 3. ШИФРОВАНИЕ НА ОСНОВЕ МЕТОДОВ ПОДСТАНОВКИ

3.1. МОНОАЛФАВИТНЫЕ ШИФРЫ

Шифр Цезаря. Самым древним и самым простым из известных подстановочных шифров является шифр, использовавшийся Юлием Цезарем. В шифре Цезаря каждая буква алфавита заменяется буквой, которая находится на 3 позиции дальше в этом же алфавите. Проще всего увидеть это на примере:

Открытый текст: meetmeafterthetogaparty.

Шифрованный текст: phhwphdiwhuwkhwrjdsdumb.

Обратите внимание на то, что алфавит считается «циклическим», поэтому после A идет Z. Определить преобразование можно, перечислив все варианты замены букв, как показано далее:

Открытый текст: abcdefghijklmnopqrstuvwxyz

Шифрованный текст: defghijklmnopqrstuvwxyzabc

Если каждой букве назначить числовой эквивалент (a = 1, b = 2 и т.д.), то алгоритм можно выразить следующими формулами. Каждая буква открытого текста p заменяется буквой шифрованного текста c:

$C = E(p) = (p + 3) \bmod 26$

В общем случае сдвиг может быть любым, поэтому обобщенный алгоритм Цезаря записывается формулой

$C = E(p) = (p + k) \bmod 26$

где k принимает значения в диапазоне от 1 до 25.

Алгоритм дешифрования также прост:

$p = D(C) = (C - k) \bmod 26$

Если известно, что определенный текст был шифрован с помощью шифра Цезаря, то с помощью простого перебора всех вариантов раскрыть шифр очень просто — для этого достаточно проверить 25 возможных вариантов ключей. На рис. 3.1 показаны результаты применения этой стратегии к указанному выше сообщению. В данном случае открытый текст распознается в третьей строке.

Применение метода последовательного перебора всех возможных вариантов оправдано следующими тремя важными характеристиками данного шифра:

1. Известны алгоритмы шифрования и дешифрования.
2. Необходимо перебрать всего 25 вариантов.
3. Язык открытого текста известен и легко узнаваем.

key	*PHHW*	*PH*	*DIWHU*	*WKH*	*WRJD*	*SDUWB*
1.	oggv	og	chvgt	vjg	vqic	rctva
2.	nffu	nf	bgufc	uif	uphb	qbsuz
3.	meet	me	after	the	toga	party
4.	ldds	ld	zesdq	sgd	snfz	ozqsx
5.	kccr	kc	ydrcp	rfc	rmey	nyprw
6.	jbbq	jb	xcqbo	qeb	qldx	mxopv
7.	iaap	ia	wbpan	pda	pksw	lwnpu
8.	hzzo	hz	vaozm	ocz	ojbv	kvmot
9.	gyyn	gy	uznyl	nby	niau	juln s
10.	fxxm	fx	tymxk	max	mhzt	itkmr
11.	ewwl	ew	sxlwj	lzw	lg ys	hsjlq
12.	dvvk	dv	rwkvi	kyv	kfxr	grikp
13.	cuuj	cu	qvjuh	jxu	jewq	fqhjo
14.	btti	bt	puitg	iwt	idvp	epgin
15.	assh	as	othsf	hvs	hcuo	dofhm
16.	zrrg	zr	nsgre	gur	gbtn	cnegl
17.	yqqf	yq	mrfqd	ftq	fasm	bmdfk
18.	xppe	xp	lqepc	esp	ezrl	alcej
19.	wood	wo	kpdob	dro	dyqk	zkbdi
20.	vnnc	vn	jocna	cqn	cxpj	yjach
21.	ummb	um	inbmz	bpm	bwoi	xizbg
22.	tlla	tl	hmaly	aol	avnh	whyaf
23.	skkz	sk	glzkx	znk	zumg	vgxze
24.	rjjy	rj	fkyjw	ymj	ytlf	ufwyd
25.	qiix	qi	ejxiv	xli	xske	tevxc

Рис. 3.1. Криптоанализ шифра Цезаря методом перебора всех вариантов ключей

В большинстве случаев, когда речь идет о защите сетей, можно предполагать, что алгоритм известен. Единственное, что делает криптоанализ на основе метода последовательного перебора практи-

чески бесполезным — это применение алгоритма, для которого требуется перебрать чрезмерно много ключей.

Третья характеристика также важна. Если язык, на котором написан открытый текст, неизвестен, то расшифрованный текст можно не распознать. Более того, исходный текст может состоять из сокращений или быть каким-либо образом сжат — это также затрудняет распознавание. Например, на рис. 3.2 показан фрагмент текстового файла, сжатого с помощью широко известного алгоритма *ZIP*. Если теперь этот файл зашифровать с помощью простого подстановочного шифра (необходимо лишь расширить шифр, чтобы он включал все символы, а не только 26 букв), то при использовании метода последовательного перебора вариантов расшифрованный текст распознать будет очень сложно.

```
        CT%(□B1(SB7'Эз9Osњз9CкЁ/юґвш^/ic™Ёщцд†ГЧъиЦD-
¬Єь#c¶j □¶Рыd9SьЃsfяе†,Є яќЋэЇЄ®rЙ‰ю]Ъ±цЊУЇж\ ВюЄЇњЎчУ»
WxeIC
ђ1ЊOjeCњdl±йЁ/®qP»ш[ї7фояоќж'РЯr}COsњз9OsJчшhJ}•цБђW7¬‰
‹ћfIOM¦Sп)ФюL©‹цQH7¬IцC—W8МГ/(rфзь%9зsГ=Г<†S—
>SйNºRцЛ4и
‹шЕњB-dГь=Юh□Ю□yseи·Л%@f†*-rjзќ©‡ш ¶Н6кЇ'‡I2eЦ¬
рЧL·mO =Ю
«oНїќzL©ыб=ВНКГlSyУвЋz¬L#»Bўs#ЧьfГ¦uЦлQлс»фOz5ъ
ИЬйµfиМЇ7FfєYCAxћг&ё№vћ Чћ6Й:Ж°Чіщиќ Ф„IыќzeћЋдБi5Нг/г
S{;к‡2™къж™9Эh,‡Ё)=яцхўNz 'г<иОК‡жlSoЋАСФ?qOSрл=(пdУi6
LHn=к¤&ЧзьЦ„dz†zLъ¦□Тї'jЧ□GuЋ'ц)µЬ иwц»Рфэ3 цп+цГ»ъ–
cOjґnwIи…4ЮЕ>NГJШх1НкZэцrQ§'Э#ЦякД={ @
```

Рис. 3.2. Пример сжатого текста

Другие моноалфавитные шифры. При наличии всего 25 возможных вариантов ключей шифр Цезаря далек от того, чтобы считаться надежно защищенным. Существенного расширения пространства ключей можно добиться, разрешив использование произвольных подстановок. Давайте еще раз вспомним шифр Цезаря:

Открытый текст:	abcdefghijklmnopqrstuvwxyz
Шифрованный текст:	defghijklmnopqrstuvwxyzabc

Если в строке «Шифрованный текст» допустить использование перестановок 26 символов алфавита, то получается 26! возможных ключей. Это на 10 порядков больше, чем размер пространства ключей, и это кажется достаточным для того, чтобы сделать невозможным успешное применение криптоанализа на основе метода последовательного перебора.

Однако для криптоаналитика существует и другая линия атаки. Если криптоаналитик имеет представление о природе открытого текста (например, о том, что это несжатый текст на английском языке), то он может использовать известную информацию о характерных признаках, присущих текстам на соответствующем языке. Чтобы показать, как этот подход применяется на практике, рассмотрим небольшой пример. Допустим, требуется расшифровать следующий шифрованный текст:

QSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZ
PHZHMDZSHZOWSFPAPPDTSVPQUZWYMXUZUHSX
EIPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ

На первом этапе можно определить относительную частоту появления в тексте различных букв и сравнить их со среднестатистическими данными для букв английского алфавита, показанными на рис. 3.3.

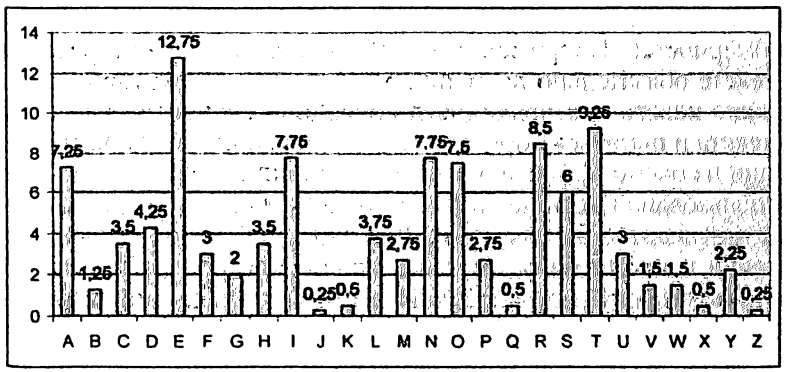


Рис. 3.3. Относительная частота появления букв в английском тексте

Если сообщение достаточно длинное, эта методика уже может быть достаточной для распознавания текста, но в нашем случае, когда сообщение небольшое, точного соответствия ожидать не приходится. В нашем случае относительная частота нахождения букв в шифрованном тексте (в процентах) оказывается табл. 3.1.

Сравнивая эти результаты с данными, показанными на рис. 3.3, можно заключить, что, скорее всего, буквы **P** и **Z** шифрованного текста являются эквивалентами букв ***e*** и ***t*** открытого текста, хотя трудно сказать, какой именно букве — **P** или **Z** — соответствует ***e***, а какой — ***t***. Буквы **S, U, O, M** и **H**, обладающие относительно высокой частотой появления в тексте, скорее всего, соответствуют буквам из множества **{r, n, i, o, a, s}**. Буквы с низкой частотой появления (а именно, **A, B, G, Y, I, J**), по-видимому, соответствуют буквам множества **{w, v, b, k, x, q, j, z}**.

Таблица 3.1

Частота появления букв в тексте

P 13,33	H 5,83	F 3,33	B 1,67	C 0,00
Z 11,67	D 5,00	W 3,33	G 1,67	K 0,00
S 8,33	E 5,00	Q 2,55	Y 1,67	L 0,00
U 8,33	V 4,17	T 2,50	I 0,83	N 0,00
O 7,50	X 4,17	A 1,67	J 0,83	R 0,00
M 6,67				

Дальше можно пойти несколькими путями. Можно, например, принять какие-то предположения о соответствиях и на их основе попытаться восстановить открытый текст, чтобы увидеть, выглядит ли такой текст похожим на что-либо осмысленное. Более систематизированный подход заключается в продолжении поиска в тексте новых характерных закономерностей. Например, может быть известно, что в рассматриваемом тексте обязательно должны присутствовать некоторые слова. Или же можно искать повторяющиеся последовательности букв шифрованного текста и пытаться определить их эквиваленты в открытом тексте.

Один из очень эффективных методов заключается в подсчете частоты использования комбинаций, состоящих из двух букв. Такие комбинации называются биграммами. Для значений относительной частоты появления в тексте биграмм тоже можно построить гистограмму, подобную показанной на рис. 3.3. Известно, что в английском языке самой распространенной является биграмма *th*. В нашем шифрованном тексте чаще всего (три раза) встречается комбинация ZW. Поэтому можно предположить, что Z соответствует *t*, а W — *h*. Тогда из ранее сформированной гипотезы вытекает, что P соответствует *e*. Заметим, что в шифрованном тексте буквосочетание ZWP имеется, и теперь мы можем представить его как *the*. В английском языке *the* является самой распространенной триграммой (т.е. комбинацией из трех букв), поэтому можно надеяться, что мы движемся в правильном направлении.

Теперь обратите внимание на комбинацию ZWSW в первой строке. Конечно, нельзя сказать с полной уверенностью, что эти буквы принадлежат одному и тому же слову, но, если предположить, что это так, они соответствуют слову *th?t*. Отсюда заключаем, что букве S соответствует *a*.

Теперь имеем следующий результат.

```
QS OVU OHXMOP V GP OZP EVS GZWS ZOP F P E S XU DBMETSXAIZ
 a          e  e te  a that e e a          a t
PHZHMDZS H ZOWS F P AP PDTS VP Q UZ WYMX UZU HS X
e  t  ta t ha e ee  a e  th  t  a
EI POP DZS Z UF P OMBZ WPF UP ZH MDJ U DT MOHMQ
  e e tat  e   the  t
```

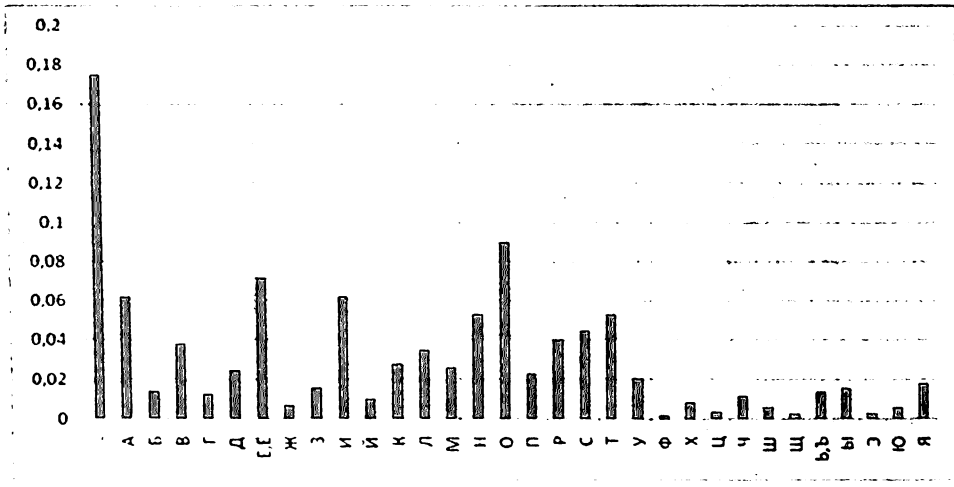



Рис. 3.4. Диаграмма частот букв русского языка

Выяснив значение всего лишь четырех букв, мы расшифровали уже значительную часть сообщения. Продолжая анализ частоты появления букв, остается проделать совсем немного работы, чтобы получить окончательный ответ.

Расшифрованный исходный текст (с добавленными в него пробелами) имеет следующий вид:

It was disclosed yesterday that several informal but direct contacts have been made with political representatives of the viet cong in Moscow

Соответственно, для русскоязычных шифров криптоанализ шифра простой замены основан на использовании статистических закономерностей русского языка. Так, например, известно, что в русском языке частоты букв распределены следующим образом (табл. 3.2).

Для получения более точных сведений об открытых текстах можно строить и анализировать таблицы k-грамм при $k > 2$, однако для учебных целей вполне достаточно ограничиться биграммами. Неравновероятность k-грамм (и даже слов) тесно связана с характерной особенностью открытого текста — наличием в нем большого числа повторений отдельных фрагментов текста: корней, окончаний, суффиксов, слов и фраз. Так, для русского языка такими привычными фрагментами являются наиболее частые биграммы и триграммы:

СТ, НО, ЕН, ТО, НА, ОВ, НИ, РА, ВО, КО,
СТО, ЕНО, НОВ, ТОВ, ОВО, ОВА.

Таблица 3.2

Частоты букв русского языка
(в 32-буквенном алфавите со знаком пробела)

пробел – 0,175	О – 0,090	Е,Ё – 0,072	А – 0.062
И – 0,062	Т – 0,053	Н – 0,053	С – 0.045
Р – 0,040	В – 0,038	Л – 0,035	К – 0,028
М – 0,026	Д – 0,025	П – 0,023	У – 0.021
Я – 0,018	Ы – 0,016	З – 0,016	ь,ъ – 0.014
Б – 0,014	Г – 0,013	ч – 0,012	Й – 0,010
Х – 0,009	Ж – 0,007	ю – 0,006	Ш – 0.006
Ц – 0,004	щ – 0,003	э – 0,003	ф – 0.002

Полезной является информация о сочетаемости букв, т.е. о предпочтительных связях букв между собой, которую легко извлечь из таблиц частот биграмм.

Имеется в виду таблица (табл. 3.3, 3.4), в которой слева и справа от каждой буквы расположены наиболее предпочтительные «соседи» (в порядке убывания частоты соответствующих биграмм). В таких таблицах обычно указывается также доля гласных и согласных букв (в процентах), предшествующих данной букве или следующих за нею.

Таблица 3.3

Таблица частот биграмм русского языка
ЧАСТЬ 1

	А	*Б*	*В*	*Г*	*Д*	*Е*	*Ж*	*З*	*И*	*И*	*К*	*Л*	*М*	*Н*	*О*	*П*
А	2	12	35	8	14	7	6	15	7	7	19	27	19	45	3	11
Б	5					9	1		6			6		2	21	
В	35	1	5	3	3	32		2	17		7	10	3	9	58	6
Г	7				3	3			5		1	5		1	50	
Д	25		3	1	1	29	1	1	13		1	5	1	13	22	3
Е	2	9	18	11	27	7	5	10	6	15	13	35	24	63	7	16
Ж	5	1			6	12			5					6		
З	35	1	7	1	5	3			4		2	1	2	9	9	1
И	4	6	22	5	10	21	2	23	19	11	19	21	20	32	8	13

Продолжение табл. 3.3

	А	*Б*	*В*	*Г*	*Д*	*Е*	*Ж*	*З*	*И*	*И*	*К*	*Л*	*М*	*Н*	*О*	*П*
И	1	1	4	1	3		1	2	4		5	1	2	7	9	7
К	24	1	4	1		4	1	1	26		1	4	1	2	66	2
Л	25	1	1	1	1	33	2	1	36		1	2	1	8	30	2
М	18	2	4	1	1	21	1	2	23		3	1	3	7	19	5
Н	54	1	2	3	3	34			58		3		1	24	67	2
О	1	28	84	32	47	15	7	18	12	29	19	41	38	30	9	18
П	7					15			4			9		1	46	

ЧАСТЬ 2

	р	*С*	*Т*	*У*	*Ф*	*Х*	*Ц*	*Ч*	*Ш*	*щ*	*ы*	*Ь*	*э*	*Ю*	*Я*
А	26	31	27	3	1	10	6	7	10	1			2	6	9
Б	8	1		6						1	11				2
В	6	19	6	7		1	1	2	4	1	18	1	2		3
Г	7			2											
Д	6	8	1	10			1	1	1		5	1			1
Е	39	37	33	3	1	8	3	7	3	3			1	1	2
Ж		1													
З	3	1		2							4				4
И	11	29	29	3	1	17	3	11	1	1			1	3	17
И	3	10	2				1	3	2						
К	10	3	7	10			1								
Л		3	1	6		4		1			2	30		4	9
М	2	5	3	9	1			2			5	1	1		3
Н	1	9	9	7	1		5	2			36	3			5
О	43	50	39	3	2	5	2	12	4	3			2	3	2
П	41	1		6							2				2

Продолжение табл. 3.3

ЧАСТЬ 3

	А	Б	В	Г	д	Е	Ж	З	И	й	К	л	м	Н	О	п
Р	55	1	4	4	3	37	3	1	24		3	1	3	7	56	2
С	8	1	7	1	2	25			6		40	13	3	9	27	11
Т	35	1	27	1	3	31		1	28		5	1	1	11	56	4
У	1	4	4	4	11	2	6	3	2		8	5	5	5	1	5
Ф	2					2			2						1	
Х	4	1	4	1	3	1		2	3		4	3	3	4	18	5
Ц	3					7			10		2				1	
Ч	12					23			13		2			6		
Ш	5					11			14		1	2		2	2	
Щ	3					8			6					1		
Ы		1	9	1	3	12		2	4	7	-3	6	6	3	2	10
Ъ		2	4	1	1	2		2	2		6		3	13	2	4
Э											1			1		
Ю		2	1	2	1			3	1		1		1	1	1	3
Я	1	3	9	1	3	3	1	5	3	2	3	3	4	6	3	6

ЧАСТЬ 4

	р	с	т	у	ф	х	ц	ч	ш	щ	ы	ь	э	ю	я
Р	1	5	9	16		1	1	1	2		8	3			5
с	4	11	82	6		1	1	2	2		1	8			17
т	26	18	2	10				1			И	21			4
у	7	14	7			1		8	3	2				9	1
ф	1	1													
х	3	4	2	2	1			1							
ц				1							1				
ч			7	1					1			1			
ш				1								1			

Окончание табл. 3.3

	р	с	т	у	ф	х	ц	ч	ш	щ	ы	ь	э	ю	я
щ				1											
ы	3	9	4	1		16		1	2						
ь	1	11	3					1	4				1	3	1
э		1	9												
ю	1	1	7				1	1		4					
я	3	6	10			2	1	4	1	1			1	1	1

Таблица 3.4

Таблица частот биграмм английского языка
ЧАСТЬ 1

	A	B	C	D	E	F	G	H	I	J	K	L	M
A	4	20	28	52	2	11	28	4	32	4	6	62	23
B	13	0	0	0	55	0	0	0	8	2	0	22	0
C	32	0	7	1	69	0	0	33	17	0	10	9	1
D	40	16	9	5	65	18	3	9	56	0	1	4	15
E	84	20	55	125	51	40	19	16	50	1	4	55	54
F	19	3	5	1	19	21	1	3	30	2	0	11	1
G	20	4	3	2	35	1	3	15	18	0	0	5	1
H	101	1	3	0	270	5	1	6	57	0	0	0	3
I	40	7	51	23	25	9	11	3	0	0	2	38	25
J	3	0	0	0	5	0	0	0	1	0	0	0	0
K	1	0	0	0	11	0	0	0	13	0	0	0	0
L	44	2	5	12	62	7	5	2	42	1	1	53	2
M	52	14	1	0	64	0	0	3	37	0	0	0	7

ЧАСТЬ 2

	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	167	2	14	0	83	76	127	7	25	8	1	9	1
B	0	11	0	0	15	4	2	13	0	0	0	15	0

Продолжение табл. 3.4

	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
C	0	50	3	0	10	0	28	11	0	0	0	3	0
D	6	16	4	0	21	18	53	19	5	15	0	3	0
E	146	35	37	6	191	149	65	9	26	31	12	5	0
F	0	51	0	0	26	8	47	6	3	3	0	2	0
G	4	21	1	1	20	9	21	9	0	5	0	1	0
H	2	44	1	0	3	10	18	6	0	5	0	3	0
I	202	56	12	1	46	79	117	1	22	0	4	0	3
J	0	4	0	0	0	0	0	3	0	0	0	0	0
K	2	2	0	0	0	6	2	1	0	2	0	1	0
L	2	25	1	1	2	16	23	9	0	1	0	33	0
M	1	17	18	1	2	12	3	8	0	1	0	2	0

ЧАСТЬ 3

	A	B	C	D	E	F	G	H	I	J	к	L	м
N	42	10	47	122	63	19	106	12	30	1	6	6	9
O	7	12	14	17	5	95	3	5	14	0	0	19	41
P	19	1	0	0	37	0	0	4	8	0	0	15	1
Q	0	0	0	0	0	0	0	0	0	0	0	0	0
R	83	8	16	23	169	4	8	8	77	1	10	5	26
S	65	9	17	9	73	13	1	47	75	3	0	7	11
T	57	22	7	1	76	5	2	330	126	1	0	14	10
U	11	5	9	6	9	1	6	0	9	0	1	19	5
V	7	0	0	0	72	0	0	0	28	0	0	0	0
W	36	1	1	0	38	0	0	33	36	0	0	4	1
X	1	0	2	0	0	1	0	0	3	0	0	0	0
Y	14	5	4	2	7	12	2	6	10	0	0	3	7
Z	1	0	0	0	4	0	0	0	0	0	0	0	0

Окончание табл. 3.4

ЧАСТЬ 4

	N	O	P	Q	R	S	m	U	V	W	X	Y	Z
N	7	54	7	1	7	44	124	6	1	15	0	12	0
O	134	13	23	0	91	23	42	55	16	28	0	4	1
P	0	27	9	0	33	14	7	6	0	0	0	0	0
Q	0	0	0	0	0	0	0	17	0	0	0	0	0
R	16	60	4	0	24	37	55	6	11	4	0	28	0
S	12	56	17	6	9	48	116	35	1	28	0	4	0
T	6	79	7	0	49	50	56	21	2	27	0	24	0
U	31	1	15	0	47	39	31	0	3	0	0	0	0
V	0	5	0	0	0	0	0	0	0	0	0	3	0
W	8	15	0	0	0	4	2	0	0	1	0	0	0
X	0	1	5	0	0	0	3	0	0	1	0	0	0
Y	5	17	3	0	4	16	30	0	0	5	0	0	0
Z	0	0	0	0	0	0	0	0	0	0	0	0	0

Пример криптоанализа шифра замены. Шифрование заключалось в замене каждой буквы на двузначное число. Отдельные слова разделены несколькими пробелами, знаки препинания сохранены (таблица частот букв русского языка приведена ранее).

29 15 10 17 29 22 25 31 15 33 35 41 43 45 35 57 45 25 17 59 15 10 25 41 25 69, 59 78 29 82 25 78 25 17 15 10 88 90 78 25 62 25 22 10 57 73 79 35 67 78 90 88 29 45 35 29, 54 57 90 31 90 73 22 88 15 88 29 15 17 69 41 25 15, 70 17 90 57 43 59 15 78 15 62 22 25 17 57 25 69 88 15 82 17 25 88 29 45 35...

Подсчитаем частоты шифрообразований:

Обозначение	29	15	10	17	22	25	31	33	35	41	43	45	57
Количество	7	10	4	7	4	12	2	1	5	3	2	4	5

Обозначение	59	69	78	82	88	90	62	73	79	67	54	70	
Количество	3	3	4	2	6	5	1	2	1	1	1	1	

Из таблицы частот букв русского языка видно, что чаще всего встречается буква О, на втором месте Е. В нашем шифротексте чаще всего встречается обозначение 25 (12 раз), на втором месте идет обозначение 15 (10 раз), остальные обозначения им существенно уступают. Поэтому можем выдвинуть гипотезу: 25 = О, 15 = Е. Однако текст у нас не очень большой, поэтому закономерности русского языка проявляются в нем не обязательно в строгом соответствии с таблицей частот букв русского языка. Поэтому возможен и вариант: 25 = Е, 15 = О. Но тогда последнее слово в третьей строке имеет окончание ЕО, что возможно, но все же более вероятный вариант ОЕ. Итак, будем работать с текстом, считая, что 25 = О, 15 = Е.

Теперь нам поможет знак препинания: «29, ...». Крайне маловероятно, чтобы запятая стояла после согласной. Итак, 29 — гласная, причем вероятнее всего 29 = И или 29 = А, так как гласные Я, Ю, Э, У встречаются в осмысленных текстах на русском языке намного реже, чем И и А, что не противоречит таблице частот шифротекста.

В последней строке: 88 15, но 15 = Е, следовательно, 88 — согласная, причем наиболее вероятные значения — это Н и Т. Итак,

$$25 = O,\ 15 = E,\ 29 = A\begin{pmatrix} A \\ И \end{pmatrix},\ 88 = \begin{pmatrix} H \\ T \end{pmatrix}.$$

Теперь третье слово в третьей строке имеет 4 варианта:

29 = И, 88 = Н:	22 Н Е Н И Е
29 = И, 88 = Т:	22 Т Е Т И Е
29 = А, 88 = Н:	22 Н Е Н А Е
29 = А, 88 = Т:	22 Т Е Т А Е

Из рассмотренных вариантов лишь один является осмысленным, и он позволяет найти значение 22. Имеем: 22 = М, и третье слово в третье строке

М Н Е Н И Е.

Теперь рассмотрим второе слово в первой строке: Е 10 17 И, причем 10 и 17 — согласные, и это не М и не Н. Наиболее вероятное слово Е С Л И, т.е. 10=С, 17=Л. Конечно, если мы, продолжая работать с текстом, вдруг получим «нечитаемое» слово, то придется вернуться к этому этапу и рассмотреть другие варианты. Однако это маловероятно, поскольку вряд ли в стихотворении были слова наподобие Е Р Т И, Е В Л И и т.п.

Далее, первое слово второй строки: 59 78 И, причем 59 и 78 — согласные, и это не С, не Л, не М и не Н. Так что это слово П Р И, т.е. 59 = П, 78 = Р. Тогда шестое слово первой строки 45 О Л П Е, что дает значение 45 = Т, и тогда при 57=В получаем фрагмент

«...В Т О Л П Е...». Также второе слово последней строки П Е Р Е 62 дает нам значение 62 = Д.

Далее рассмотрим начало второй строки: «П Р И 82 О Р О Л Е С Н 90 Р О Д О М ...». Из него следует, что 82 = К и 90 = А.

Зная, что 82 = К, посмотрим на самое последнее слово К Л О Н И Т 35, откуда станет ясно, что 35 = Ь.

Перед последней атакой выпишем текст, заменяя известные обозначения буквами.

И ЕСЛИ М О 31 Е 33 Ь 41 43 Т Ь В Т О Л П Е С О 41 О 69,
П Р И К О Р О Л Е С Н А Р О Д О М С В 73 79 Ь 67 Р А Н И Т Ь
И, 54 В А 31 А 73 М Н Е Н И Е Л 69 41 О Е,
70 Л А В 43 П Е Р Е Д М О Л В О 69 Н Е К Л О Н И Т Ь...

Из последней строки: 69 = Ю, тогда слова Л Ю 41 О Е и С О 41 О Ю определяют 41: 41 = Б. Теперь из четвертого слова первой строки Б 43 Т Ь получаем, что 43 = Ы. А первое слово из последней строки 70 Л А В Ы — это Г Л А В Ы. Слово в первой строке М О 31 Е 33 Ь угадывается из контекста: М О Ж Е Ш Ь, т.е. 31 = Ж, 33 = Ш. Теперь второе слово в третьей строке запишется как 54 В А Ж А 73, откуда, с учетом контекста: 54 = У, 73 = Я. После этого окончание второй строки имеет вид «... С В Я 79 Ь 67 Р А Н И Т Ь». Легко определяются буквы 79 = З, 67 = Х.

Ответ:

И ЕСЛИ МОЖЕШЬ БЫТЬ В ТОЛПЕ СОБОЮ, ПРИ КОРОЛЕ С НАРОДОМСВЯЗЬ ХРАНИТЬИ, УВАЖАЯ МНЕНИЕ ЛЮБОЕ, ГЛАВЫ ПЕРЕД МОЛВОЮ НЕ КЛОНИТЬ...

Моноалфавитные шифры легко раскрываются, так как они наследуют частотность употребления букв оригинального алфавита. Контрмерой в данном случае является применение для одной буквы не одного, а нескольких заменителей (называемых ***омофонами***). Например, букве *е* исходного текста может соответствовать несколько разных символов шифра (скажем, 16, 74, 35 и 21), причем каждый такой символ может использоваться либо поочередно, либо по случайному закону. Если число символов-заменителей, назначенных букве, выбрать пропорциональным частоте появления этой буквы, то подсчет частотности употребления букв в шифрованном тексте становится бессмысленным. Великий математик Карл Фридрих Гаусс (*Carl Friedrich Gauss*) был уверен, что с использованием омофонов он изобрёл шифр, который невозможно взломать. Но даже при употреблении омофонов каждому элементу открытого текста соответствует только один элемент шифрованного текста, поэтому в последнем по-прежнему должны наблюдаться характерные показатели частоты повторения комбинаций

нескольких букв (например, биграмм), и в результате задача криптоанализа по-прежнему остается достаточно элементарной.

Чтобы в тексте, шифрованном с помощью методов подстановок, структура исходного текста проявлялась менее заметно, можно использовать два принципиально разных подхода. Один из них заключается в замещении не отдельных символов открытого текста, а комбинаций нескольких символов, а другой подход предполагает использование для шифрования нескольких алфавитов.

Шифр Плейфейера. Одним из наиболее известных шифров, базирующихся на методе многобуквенного шифрования, является шифр Плейфейера (*Playfair*), в котором биграммы открытого текста рассматриваются как отдельные единицы, преобразуемые в заданные биграммы шифрованного текста.

Алгоритм Плейфейера основан на использовании матрицы букв размерности 5×5, созданной на основе некоторого ключевого слова. Давайте рассмотрим следующий пример, в котором ключевым словом является ***monarchy*** (монархия).

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

Матрица создается путем размещения букв, использованных в ключевом слове, слева направо и сверху вниз (повторяющиеся буквы отбрасываются). Затем оставшиеся буквы алфавита размещаются в естественном порядке в оставшихся строках и столбцах матрицы. Буквы *I* и *J* считаются одной и той же буквой. Открытый текст шифруется порциями по две буквы в соответствии со следующими правилами.

1. Если оказывается, что повторяющиеся буквы открытого текста образуют одну пару для шифрования, то между этими буквами вставляется специальная буква-заполнитель, например *x*. В частности, такое слово как balloon будет преобразовано к *balxloon*.

2. Если буквы открытого текста попадают в одну и ту же строку матрицы, каждая из них заменяется буквой, следующей за ней в той же строке справа, с тем условием, что для замены последнего элемента строки матрицы служит первый элемент той же строки. Например, *ar* шифруется как *RM*.

3. Если буквы открытого текста попадают в один и тот же столбец матрицы, каждая из них заменяется буквой, стоящей в том же столбце сразу под ней, с тем условием, что для замены самого нижнего элемента столбца матрицы берется самый верхний элемент того же столбца. Например, ***mu*** шифруется как *CM*.

4. Если не выполняется ни одно из приведенных выше условий, каждая буква из пары букв открытого текста заменяется буквой, находящейся на пересечении содержащей эту букву строки матрицы и столбца, в котором находится вторая буква открытого текста. Например, *hs* шифруется как *BP*, а *ea* — как *IM* (или *JM*, по желанию шифровальщика).

Шифр Плейфейера значительно надежнее простых моноалфавитных шифров.

С одной стороны, букв всего 26, а биграмм — 26 × 26 = 676, и уже поэтому идентифицировать биграммы сложнее, чем отдельные буквы.

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

По этим причинам долго считалось, что шифр Плейфейера взломать невозможно. Он служил стандартом шифрования в Британской армии во время Первой мировой войны и нередко применялся в армии США и в союзных войсках даже в период Второй мировой войны.

Несмотря на столь высокую репутацию в прошлом, шифр Плейфейера на самом деле вскрыть относительно легко, так как шифрованный с его помощью текст все равно сохраняет многие статистические характеристики открытого текста. Для взлома этого шифра, как правило, достаточно иметь шифрованный текст, состоящий из нескольких сотен букв.

Шифр Хилла. Еще одним интересным многобуквенным шифром является шифр, разработанный математиком Лестером Хиллом (*Lester Hill*) в 1929 г. Лежащий в его основе алгоритм заменяет каждые *m* последовательных букв открытого текста *m* буквами шифрованного текста. Подстановка определяется *m* линейными уравнениями, в которых каждому символу присваивается численное значение (a = 0, b = 1, …, z = 25).

Например, при $m = 3$ получаем следующую систему уравнений:

$$C_1=(k_{11}p_1+ k_{12}p_2+ k_{13}p_3), \pmod{26}$$

$$C_1=(k_{21}p_1+ k_{22}p_2+ k_{23}p_3), \pmod{26}$$

$$C_1=(k_{31}p_1+ k_{32}p_2+ k_{33}p_3), \pmod{26}$$

Эту систему уравнений можно записать в виде произведения вектора и матрицы в следующем виде:

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

или в виде $C = KP$, где **C** и **P** — векторы длины 3, представляющие соответственно шифрованный и открытый тексты; **K** — матрица размерности 3 × 3, представляющая ключ шифрования. Операции выполняются по модулю 26.

Рассмотрим, например, как при использовании ключа будет шифрован текст «**paymoremoney**».

$$K = \begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix}$$

Первые три буквы открытого текста представлены вектором (15, 0, 24). Таким образом,

$\mathbf{K} \times (15,0,24) = (275, 819, 486) \pmod{26} = (11, 13, 18) = \mathbf{LNS}$.

Продолжая вычисления, получим для данного примера шифрованный текст вида LNSHDLEWMTRW.

Для расшифровки нужно воспользоваться матрицей, обратной **K**.

Обратной по отношению к матрице **K** называется такая матрица $\mathbf{K}^{-1}$, для которой выполняется равенство $\mathbf{K}\mathbf{K}^{-1} = \mathbf{K}^{-1}\mathbf{K} = \mathbf{I}$, где **I** — единичная матрица (матрица, состоящая из нулей всюду, за исключением главной диагонали, проходящей из левого верхнего угла в правый нижний, на которой располагаются единицы). Обратная матрица существует не для всякой матрицы, однако когда обратная матрица имеется, для нее обязательно выполняется приведенное выше равенство.

В рассмотренном примере обратной матрицей является матрица

$$K^{-1} = \begin{bmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{bmatrix}$$

Это проверяется следующими вычислениями:

$$\begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix} \begin{bmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{bmatrix} = \begin{bmatrix} 443 & 442 & 442 \\ 858 & 495 & 780 \\ 494 & 52 & 365 \end{bmatrix} \pmod{26} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Легко видеть, что в результате применения матрицы $\mathbf{K}^{-1}$ к шифрованному тексту получается открытый текст.

В общем виде систему Хилла можно записать в следующей форме:

$$\mathbf{C} = \mathbf{Ek}(\mathbf{P}) = \mathbf{KP},$$
$$\mathbf{P} = \mathbf{Dk}(\mathbf{C}) = \mathbf{K}^{-1}\mathbf{C} = \mathbf{K}^{-1}\mathbf{KP} = \mathbf{P}$$

Как и в случае шифра Плейфейера, преимущество шифра Хилла состоит в том, что он полностью маскирует частоту вхождения отдельных букв. А для шифра Хилла чем больше размер матрицы в

шифре, тем больше в шифрованном тексте скрывается информации о различиях в значениях частоты появления других комбинаций символов. Так, шифр Хилла с матрицей 3 × 3 скрывает частоту появления не только отдельных букв, но и двухбуквенных комбинаций.

Хотя шифр Хилла устойчив к попыткам криптоанализа в тех случаях, когда известен только шифрованный текст, этот шифр легко раскрыть при наличии известного открытого текста.

Рассмотрим шифр Хилла с матрицей m × m. Предположим, что известны m пар отрывков открытого и соответствующего шифрованного текстов, каждый длиной m. Выберем такие пары $\mathbf{P}_j = (p_{1j}, p_{2j}, \ldots p_{mj})$ и $\mathbf{C}_j = (C_{1j}, C_{2j}, \ldots C_{mj})$,, чтобы выполнялось условие $\mathbf{C}_j = \mathbf{K}\mathbf{P}_j$ для всех $1 \le j \le m$ и некоторой известной ключевой матрицы K. Теперь определим две такие матрицы $\mathbf{X} = (p_{ij})$ и $\mathbf{Y} = (C_{ij})$ размера m×m, что $\mathbf{Y} = \mathbf{XK}$. Тогда при условии, что для матрицы X существует обратная матрица, матрицу-ключ **K** можно определить по формуле $\mathbf{K} = \mathbf{X}^{-1}\mathbf{Y}$. Если же получить матрицу, обратную матрице **X**, невозможно, необходимо формировать другую матрицу **X** с дополнительными парами соответствия открытого и шифрованного текстов до тех пор, пока не будет найдена обратная матрица.

Рассмотрим **пример**. Предположим, что открытый текст **friday** зашифрован с помощью шифра Хилла с использованием матрицы 2×2, в результате чего получен шифрованный текст **PQCFKU**. Таким образом, мы знаем, что $\mathbf{K} \times (5, 17) = (15, 16)$, $\mathbf{K} \times (8, 3) = (2, 5)$ и $\mathbf{K} \times (0, 24) = (10, 20)$. Используя первые две пары символов открытого и шифрованного текстов, получаем

$$\begin{bmatrix} 15 & 16 \\ 2 & 5 \end{bmatrix} = \begin{bmatrix} 5 & 17 \\ 8 & 3 \end{bmatrix} K.$$

Вычислим матрицу, обратную матрице X:

$$\begin{bmatrix} 5 & 17 \\ 8 & 3 \end{bmatrix}^{-1} = \begin{bmatrix} 9 & 1 \\ 2 & 15 \end{bmatrix}.$$

Таким образом, теперь можно получить значение ключа:

$$K = \begin{bmatrix} 9 & 1 \\ 2 & 15 \end{bmatrix} \begin{bmatrix} 15 & 16 \\ 2 & 5 \end{bmatrix} = \begin{bmatrix} 7 & 19 \\ 8 & 3 \end{bmatrix}.$$

Полученный результат можно проверить с помощью оставшихся пар открытого и шифрованного текстов.

3.2. ПОЛИАЛФАВИТНЫЕ ШИФРЫ

Другая возможность усовершенствования простого моноалфавитного шифра заключается в использовании нескольких моноалфа-

витных подстановок, применяемых в ходе шифрования открытого текста в зависимости от определенных условий. Семейство шифров, основанных на применении таких методов шифрования, называется полиалфавитным и обладает следующими особенностями:

Используется набор связанных моноалфавитных подстановок.

Имеется некоторый ключ, по которому определяется, какое конкретное преобразование должно применяться для шифрования на данном этапе.

Шифр Виженера. Самым широко известным и одновременно самым простым алгоритмом такого рода является шифр Виженера. Этот шифр базируется на наборе правил моноалфавитной подстановки, представленных 26 шифрами Цезаря со сдвигом от 0 до 25. Каждый из таких шифров можно обозначить ключевой буквой, являющейся буквой шифрованного текста, соответствующей букве открытого текста. Например, шифр Цезаря, для которого смещение равно 3, обозначается ключевой буквой *d*.

Для облегчения понимания и применения этой схемы была предложена матрица, называемая «табло Виженера» (табл. 3.5). Все 26 шифров располагаются по горизонтали, и каждому из шифров соответствует своя ключевая буква, представленная в крайнем столбце слева. Алфавит, соответствующий буквам открытого текста, находится в первой сверху строке таблицы. Процесс шифрования прост — необходимо по ключевой букве *x* и букве открытого текста *y* найти букву шифрованного текста, которая находится на пересечении строки *x* и столбца *y*. В данном случае такой буквой является буква **V**.

Чтобы зашифровать сообщение, нужен ключ, имеющий ту же длину, что и само сообщение. Обычно ключ представляет собой повторяющееся нужное число раз ключевое слово, чтобы получить строку подходящей длины. Например, если ключевым является слово deceptive, сообщение «wearediscoveredsaveyourself» шифруется следующим образом.

Ключ: **deceptivedeceptivedeceptive**
Открытый текст: **wearediscoveredsaveyourself**
Шифрованный текст: **ZICVTWQNGRZGVTWAVZHCQYGLMGJ**

Расшифровать текст также просто: буква ключа определяет строку, буква шифрованного текста, находящаяся в этой строке, определяет столбец, и в этом столбце в первой строке таблицы будет находиться соответствующая буква открытого текста.

Таблица 3.5

	a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*	*l*	*m*	*n*	*o*	*p*	*q*	*r*	*s*	*t*	*u*	*v*	*w*	*x*	*y*	*z*
a	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
b	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
c	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
d	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
e	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
f	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
g	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
h	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
i	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
j	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
k	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
l	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
m	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
n	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
o	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
p	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
r	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
s	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
t	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
u	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
v	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
w	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
x	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Преимущество этого шифра заключается в том, что для представления одной и той же буквы открытого текста в шифрованном

тексте имеется много различных вариантов — по одному на каждую из неповторяющихся букв ключевого слова. Таким образом, скрывается информация, характеризующая частотность употребления букв. Но и с помощью данного метода все же не удается полностью скрыть влияние структуры открытого текста на структуру шифрованного — хотя он и обладает явным преимуществом по сравнению с шифром Плейфейера, но, тем не менее, полностью информацию о распределении частоты замаскировать не удается.

Кратко рассмотрим метод взлома шифра Виженера, так как на примере этого метода можно показать некоторые из математических принципов, лежащих в основе большинства современных методов криптоанализа.

Прежде всего предположим следующее: противник уверен в том, что шифрованный текст был получен либо с помощью моноалфавитной подстановки, либо с помощью шифра Виженера. Чтобы выяснить, какой именно из этих двух методов был использован, можно провести простой тест. Если использовалась моноалфавитная подстановка, статистические показатели шифрованного текста не будут отличаться от соответствующих показателей языка, на котором написан открытый текст. Поскольку для анализа используется лишь одно сообщение, точное совпадение статистических показателей можно и не получить. Но если статистика достаточно точно повторяет статистику обычного открытого текста, можно предположить, что использовалась моноалфавитная подстановка.

Если же, наоборот, все указывает на то, что был применен шифр Виженера, то, как будет показано несколько позже, успех дальнейшего анализа текста зависит от того, удастся ли определить длину ключевого слова. Пока сосредоточимся на том, как определить длину ключевого слова. Решение этой задачи основано на следующей особенности данного шифра: если начальные символы двух одинаковых последовательностей открытого текста находятся друг от друга на расстоянии, кратном длине ключа, эти последовательности будут представлены одинаковыми последовательностями и в шифрованном тексте. В рассматриваемом далее примере имеется две последовательности ***red***, и начало второй из них оказывается на девять символов дальше относительно начала первой. Следовательно, в обоих случаях ***r*** будет шифровано с использованием ключевой буквы ***e***, ***e*** — с помощью ключевой буквы ***p***, а ***d*** — с помощью ключевой буквы ***t***. Таким образом, в обоих случаях для шифрованного текста будет получена последовательность **VTW**.

Аналитик, имеющий в своем распоряжении только шифрованный текст, обнаружит повторяющуюся последовательность **VTW** со

смещением в девять символов, и поэтому может предположить, что длина ключевого слова равна либо трем, либо девяти. Конечно, для повторившейся всего два раза последовательности **VTW** совпадение может оказаться и случайным, а потому не соответствовать шифрованным с одинаковыми ключевыми буквами одинаковым фрагментам открытого текста, но если сообщение будет достаточно длинным, то таких повторяющихся последовательностей в нем будет немало. Определив общий множитель для смещения начала таких последовательностей, аналитик получит достаточно надежную основу для предположений о длине ключевого слова.

Дальнейший анализ базируется на другой особенности данного шифра. Если ключевое слово имеет длину **N**, то шифр, по сути, состоит из моноалфавитных подстановочных шифров. Например, при использовании ключевого слова *deceptive* буквы, находящиеся на 1-й, 10-й, 19-й и т.д. позициях, шифруются одним и тем же моноалфавитным шифром. Это дает возможность использовать известные характеристики частотных распределений букв открытого текста для взлома каждого моноалфавитного шифра в отдельности.

Периодичности в ключевой строке можно избежать, используя для ключевой строки периодическую последовательность той же длины, что и само сообщение. Виженер предложил подход, получивший название системы с автоматическим выбором ключа, когда последовательность ключевой строки получается в результате конкатенации ключевого слова с самим открытым текстом. Для рассматриваемого примера получим:

Ключ: **d e c e p t i v e w e a r e d i s c o v e r e d s a v**

Открытый текст: **w e a r e d i s c o v e r e d s a v e y o u r s e l f**

Шифрованный текст: **Z I C V T W Q N G K Z E I I G A S X S T S L V V W L A**

Однако и эта схема оказывается уязвимой. Поскольку и в ключевой строке, и в открытом тексте значения частот распределения букв будут одинаковы, статистические методы можно применить и в данном случае. Например, буква ***r***, шифрованная с помощью ключа ***e***, согласно статистике должна встречаться с частотой $(0{,}1275)^2 \approx 0{,}0163$, тогда как ***d***, шифрованная с помощью ***t***, может встретиться с частотой, примерно в два раза меньшей. Именно такие закономерности позволяют добиться успеха при анализе шифрованного текста.

Лучшей защитой от подобных методов криптоанализа является выбор ключевого слова, по длине равного длине открытого текста, но отличающегося от открытого текста по статистическим показателям.

3.3. ТЕОРИЯ КРИПТОАНАЛИЗА ШИФРА ВИЖЕНЕРА

Рассмотрим шифр модульного гаммирования с уравнением

$$b_i = (a_i + y_i) \pmod{n},$$

для которого гамма является периодической последовательностью знаков алфавита. Такая гамма обычно получается периодическим повторением некоторого ключевого слова. Например, ключевое слово KEY дает гамму KEYKEYKEY... Рассмотрим задачу вскрытия такого шифра по тексту одной криптограммы достаточной длины.

Пусть μ — длина ключевого слова. Обычно криптоанализ шифра Виженера проводится в два этапа. На первом этапе определяется число μ, на втором этапе — само ключевое слово.

Для определения числа μ применяется так называемый тест Казиски, названный в честь Ф. Казиски, применившего его в 1863 г. Тест основан на простом наблюдении о том, что два одинаковых отрезка открытого текста, отстоящих друг от друга на расстоянии, кратном μ, будут одинаково зашифрованы. В силу этого в шифротексте ищутся повторения длины, не меньшей трех, и расстояния между ними. Обратим внимание на то, что случайно такие одинаковые отрезки могут появиться в тексте с достаточно малой вероятностью.

Пусть d_1, d_2,... — найденные расстояния между повторениями и d — наибольший общий делитель этих чисел. Тогда μ должно делить d. Чем больше повторений имеет текст, тем более вероятно, что μ совпадает с d. Для уточнения значения μ можно использовать так называемый индекс совпадения, введенный в практику У. Фридманом в 1920 г.

Для строки $x = (x_1,\dots x_m)$ длины m, составленной из букв алфавита A, индексом совпадения в x, обозначаемым $I_c(x)$, будем называть вероятность того, что две случайно выбранные буквы из x совпадают.

Пусть $A = \{a_i,\dots, a_n\}$. Будем отождествлять буквы алфавита с числами, так что

$$a_1 \equiv 0,\dots, a_{n-1} \equiv n-2,\ a_n \equiv n-1.$$

Теорема. Индекс совпадения в x вычисляется по формуле

$$I_c(x) = \frac{\sum_{i=0}^{n-1} f_i(f_i-1)}{m(m-1)}. \qquad (3.1)$$

где f_i — число вхождений буквы a_i в x, $i \in Z_n$.

Доказательство. Будем вычислять $I_c(x)$ как отношение числа благоприятных исходов к общему числу исходов. Благоприятным

является исход, при котором на выбранных двух позициях в x расположены одинаковые буквы. Общее число исходов равно, очевидно, C^2_m. Число благоприятных исходов есть

$$\sum_{i=0}^{m-1} C^2_{f_i} \qquad (3.2)$$

В самом деле, переупорядочим буквы в x таким образом, чтобы сначала шли f_{a_1} букв a_1, затем — f_{a_2} букв a_2 и т.д. (3.4):

$$\underbrace{a_1,\dots,a_1}_{f_{a_1}} \dots \underbrace{a_n,\dots,a_n}_{f_{a_n}} \qquad (3.3)$$

Теперь заметим, что при случайном выборе мест (i и j) в строке x благоприятными являются следующие исходы:

$$(a_1)\begin{cases} 0\dots i\dots j\dots m-1 \\ \dots a_1\dots a_1\dots \end{cases}$$

$$(a_2)\begin{cases} 0\dots i\dots j\dots m-1 \\ \dots a_2\dots a_2\dots \end{cases}$$

$$\dots\dots\dots\dots\dots$$

$$(a_n)\begin{cases} 0\dots i\dots j\dots m-1 \\ \dots a_n\dots a_n\dots \end{cases}$$

В случае (a_1) можем выбрать пару букв a_1 из набора (3.3) $C^2_{f_{a_1}}$ способами, в случае (a_2) пару букв a_2 из (3.3) — $C^2_{f_{a_2}}$ способами и т.д.

Таким образом, общее число благоприятных исходов выражается величиной (3.2), а индекс совпадения в x — формулой

$$I_c(x) = \frac{\sum_{i=0}^{m-1} C^2_{f_i}}{C^2_m}$$

и, следовательно, формулой (3.1).

Пусть x — строка осмысленного текста (например, английского). Допустим, как и ранее, что буквы в x появляются на любом месте текста с соответствующими вероятностями $p_0, \ldots, p_{n-1}$ независимо друг от друга, где p_i — вероятность появления буквы i в осмысленном тексте, $i \in Z_n$. В такой модели открытого текста вероятность того, что две случайно выбранные буквы из x совпадают с $i \in Z_n$, равна p^2_i, следовательно

$$I_c(x) \approx \sum_{i=0}^{n-1} p_i^2 \qquad (3.4)$$

Взяв за основу значения вероятностей p_i для открытых текстов на английском языке, получаем приближение $\sum_{i=0}^{25} p_i^2 \approx 0{,}066$. Тем самым для английских текстов x можно пользоваться следующим приближением для индекса совпадения:

$$I_c(x) \approx 0{,}066.$$

Аналогичные приближения можно получить и для других языков. Так, для русского языка получаем приближение

$$I_c(x) \approx 0{,}053.$$

Приведем значения индексов совпадения для ряда европейских языков (табл. 3.6).

Таблица 3.6

Индексы совпадения европейских языков

Язык	*Русск.*	*Англ.*	*Франц.*	*Немец.*	*Итал.*	*Испан.*
$I_c(x) \approx$	0,0529	0,0662	0,0778	0,0762	0,0738	0,0775

Рассуждения, использованные при выводе формулы (3.4), остаются, очевидно, справедливыми и в случае, когда x есть результат зашифрования некоторого открытого текста простой заменой. В этом случае вероятности p_i переставляются местами, но сумма $\sum_{i=0}^{n-1} p_i^2$ остается неизменной.

Предположим, что x — реализация независимых испытаний случайной величины, имеющей равномерное распределение на Z_n. Тогда индекс совпадения вычисляется по формуле

$$I_c(x) = \sum_{i=0}^{n-1} \frac{1}{n^2} = n \cdot \frac{1}{n^2} = \frac{1}{n}$$

Вернемся к вопросу об определении числа μ.

Пусть $y = y_1 y_2 \ldots y_n$ — данный шифр-текст. Выпишем его с периодом μ:

$Y_1^{\downarrow}$	$Y_2^{\downarrow}$	...	$Y_\mu^{\downarrow}$
y_1	y_2	...	y_μ
$y_{\mu+1}$	$y_{\mu+2}$	...	$y_{2\mu}$
$y_{2\mu+1}$	$y_{2\mu+2}$	...	$y_{3\mu}$
...	...	...	...

и обозначим столбцы получившейся таблицы через $Y^1_1, \ldots, Y^1_\mu$. Если μ — это истинная длина ключевого слова, то каждый столбец Y^1_i, $i \in 1, \mu$, представляет собой участок открытого текста, зашифрованный простой заменой, определяемой подстановкой

$$\begin{pmatrix} 0 & 1 & 2 & \ldots & n-s & \ldots & n \\ s & s+1 & s+2 \ldots & & 0 & \ldots & s-1 \end{pmatrix} \qquad (3.5)$$

для некоторого $s \in 0, n-1$ (числа берутся по модулю n).

В силу изложенного, для английского языка $I_c(Y^1_i) \approx 0{,}066$ при любом i. С другой стороны, если μ отлично от длины ключевого слова, то столбцы Y^1_i будут более «случайными», поскольку они являются результатом зашифрования фрагментов открытого текста некоторым многоалфавитным шифром. Тогда $I_c(Y^1_i)$ будет ближе (для английского языка) к числу $1/28 \approx 0{,}038$.

Заметное различие значений $I_c(x)$ для осмысленных открытых текстов и случайных последовательностей букв (для английского языка — 0,066 и 0,038, для русского языка — 0,053 и 0,030) позволяет в большинстве случаев установить точное значение μ.

Предположим, что на первом этапе мы нашли длину ключевого слова μ. Рассмотрим теперь вопрос о нахождении самого ключевого слова. Для этого можно использовать так называемый взаимный индекс совпадения.

Пусть $x = (x_1,\ldots,x_n), y = (y_1,\ldots,y_m)$ — две строки букв алфавита A. Взаимным индексом совпадения x и y, обозначаемым $MI_c(x, y)$, называется вероятность того, что случайно выбранная буква из x совпадает со случайно выбранной буквой из y.

Пусть $f_0 f_1 \ldots f_n$ и $f'_0 f'_1 \ldots f'_{n-1}$ — числа вхождений букв алфавита в x и y соответственно.

Взаимный индекс совпадения в x и y вычисляется по формуле (эта теорема доказывается точно так же, как и предыдущая теорема)

$$MI_c(x, y) = \frac{\sum_{i=0}^{n-1} f_i \cdot f'_i}{m \cdot m'} \qquad (3.6)$$

Пусть $k = (k_1,\ldots, k_\mu)$ — истинное ключевое слово. Попытаемся оценить индексы $MI_c(Y^\downarrow_i, Y^\downarrow_j)$.

Для этого напомним, что $Y^\downarrow_3$ является результатом зашифрования фрагмента открытого текста простой заменой, определяемой подстановкой (3.5) при некотором s. Вероятность того, что $Y^\downarrow_i$ и $Y^\downarrow_j$ произвольная пара букв равна 0, имеет вид $p_{n-s_i} \cdot p_{n-s_j}$ (где p_a — вероятность появления буквы a в открытом тексте); вероятность того, что обе буквы есть 1, равна $p_{n-s_i+1} \cdot p_{n-s_j+1}$ и так далее. На основании этого получаем:

$$MI_c(Y_i^\downarrow, Y_j^\downarrow) \approx \sum_{h=0}^{n-1} p_{h-s_i} \cdot p_{h-s_j} = \sum_{h=0}^{n-1} p_h \cdot p_{h+(s_i-s_j)}$$

Заметим, что сумма в правой части последнего равенства зависит только от разности $(s_i - s_j) \bmod n$, которую назовем относительным сдвигом $Y^\downarrow_i$ и $Y^\downarrow_j$. Заметим также, что

$$\sum_{j=0}^{n-1} p_j \cdot p_{(j+s) \bmod n} = \sum_{j=0}^{n-1} p_j \cdot p_{(j-s) \bmod n}, \qquad (3.7)$$

поэтому $Y^\downarrow_i$ и $Y^\downarrow_j$ с относительными сдвигами s и $n - s$ имеют одинаковые взаимные индексы совпадения. Приведем таблицу значений сумм (табл. 3.7) для английского языка.

Таблица 3.7

Взаимный индекс совпадения при сдвиге s

Сдвиг s	0	1	2	3	4	5	6
$MI_c(x, y) \approx$	0,066	0,039	0,032	0,034	0,044	0,033	0,036
Сдвиг s	7	8	9	10	11	12	13
$MI_c(x, y) \approx$	0,039	0,034	0,034	0,038	0,045	0,039	0,043

Обратим внимание на то, что ненулевые «сдвиги» дают взаимные индексы совпадения, изменяющиеся в пределах от 0,032 до 0,045, в то время как при нулевом сдвиге индекс $MI_c(x,y)$ близок к 0,066. Это наблюдение позволяет определить величины относительных сдвигов $s_i - s_j$ столбцов $Y^\downarrow_i$ и $Y^\downarrow_j$. Для этого заметим, что при некотором значении $s(i,j) \in 0, n-1$ столбец $Y^{s(i,j)\downarrow}_j$, полученный из $Y^\downarrow_j$ прибавлением к каждому его элементу числа $S(i, j)$ (по модулю n), имеет нулевой относительный сдвиг с $Y^\downarrow_i$.

Пусть $Y^{0\downarrow}_j, Y^{1\downarrow}_j, \ldots, Y^{n-1\downarrow}_j$ — результаты зашифрования $Y^\downarrow_j$ каждой из простых замен (3.5). Несложно вычислить взаимные индексы

$$MI_c(Y_i^\downarrow, Y_j^\downarrow), \quad 0 \le s \le n-1, \quad 1 \le i < j \le \mu$$

(всего, таким образом, имеется $C^2_\mu n$ значений). Для этого воспользуемся формулой, полученной из (3.6):

$$MI_c(Y_i^\downarrow, Y_j^{s\downarrow}) = \frac{\sum_{h=0}^{n-1} f_h \cdot f'_{h-s}}{m \cdot m'}$$

Если s равно $s_i - s_j$ — (относительному сдвигу $Y^\downarrow_i$ и $Y^\downarrow_j$), то взаимный индекс впадения должен быть (для английского языка) близок к 0,066, так как относительный сдвиг $Y^\downarrow_i$ и $Y^\downarrow_j$ равен нулю. Если же $s \ne s_i - s_j$, то взаимный индекс совпадения должен колебаться в пределах 0,032–0,045.

Используя изложенный метод, мы сможем связать системой уравнений относительные сдвиги различных пар столбцов $Y^\downarrow_i$ и $Y^\downarrow_j$. В результате останется 26 (для английского языка) вариантов для ключевого слова, из которых можно выбрать наиболее предпочтительный вариант (если ключевое слово является осмысленным).

Следует отметить, что предложенный метод будет эффективным для не слишком больших значений μ. Это объясняется тем, что для

хороших сближений индексов совпадения требуются тексты достаточно большой длины.

Пример криптоанализа текста. Задан некоторый текст, зашифрованный шифром Виженера; требуется определить ключевое слово и прочитать открытый текст.

Шифрованный текст:

Влцдутжбюцхъяррмшбрхцэооэцгбрьцмйфктъъюьмшэсяцпунуящэ
йтаьэдкцибрьцгбрпачкъуцпъбьсэгкцъгуущарцёэвърюуоюэкааэбрняф
укабъарпяъафкъиьжяффнйояфывбнэнфуюгбрьсшьжэтбэёчюъюръего
фкбьчябашвёэуъъюаднчжчужцёэвлрнчулбюпцуруньъшсэюъзкцхъяр
рнрювяспэмасчкпэужьжыатуфуярюравртубурьпэщлафоуфбюацмнуб
сюкйтаьэдйюнооэгюожбгкбръницэпотчмёодзцвбцшщвщепчдчдръюьс
касэгъппэгюкдойрсрэвоопчщшоказръббнэугнялёкьсрбёуыэбдэулбюа
сшоуэтъшкрсдугэфлбубуъчнчтртпэгюкиугюэмэгюккъъпэгяапуфуэзь
радзьжчюрмфцхраююанчёчюъыхьъцомэфъцпоирькнщпэтэузуябащу
щбаыэйчдфрпэцъърьцъцпоилуфэдцойэдятррачкубуфнйтаьэдкцкрннц
юабугюуубурьпйюэъжтгюркующоъуфъэгясуоичщщчдцсфырэдщэъу
яфшёчцюйрщвяхвмкршрпгюопэуцчйтаьэдкцибрьцыяжтюрбуэтэбдуя
щэубъибрювъежагибрбагбрымпуноцшяжцечкфодщоъчжшйуъцхчщв
уэбдлдъэгясуахзцэбдэулькнъщбжяцэьрёдъьвюврнуяфуоухфекьгцчч
чгэъжтанопчынажпачкъуъмэнкйрэфщэъьбудэндадъярьеюэлэтчоубъц
эфэвлнёэгфдсэвэёкбсчоукгаутэыпуббцчкпэгючсаъбэнэфъркацхёвает
уфяепьрювържадфёжбьфутощоявьъгупчршуитеачйчирамчюфчоуяю
онкяжыкгсцбрясшчйотъъжрсщчлё

Для определения числа букв в данном ключевом слове применяется так называемый тест Казиски. Тест основан на простом наблюдении того, что два одинаковых отрезка открытого текста, отстоящих друг от друга на расстоянии, кратном μ (число букв в слове), будут одинаково зашифрованы. В силу этого в шифротексте ищутся повторения длины, не меньшей трех, и расстояния между ними. Необходимо обратить внимание на то, что случайно такие одинаковые отрезки могут появиться в тексте с достаточно малой вероятностью

В данном тексте обнаружено четырехкратное повторение буквосочетания «брь». Выясним расстояние между ними и найдем наибольший общий делитель этих расстояний.

В результате получаем:

35, 85, 510 и НОД = 5.

Следовательно, с определенной долей вероятности можно заключить, что длина кодового слова равна 5.

Для подтверждения гипотезы воспользуемся математической статистикой для определения длины ключевого слова. Для этого запишем шифротекст в таблицу с 5 столбцами, предполагая, что длина ключевого слова равна 5.

Вычислим взаимные индексы совпадения $I_c(x)$ букв в каждом из столбцов таблицы, для достоверного установления длины ключевого слова. Для этого посчитаем частоту повторения букв в каждом столбце. Таблица состоит из 5 столбцов, так как на предыдущем этапе нами было установлено, что ключевое слово по НОД может состоять из 5 букв.

№ n/n	Y_1	Y_2	Y_3	Y_4	Y_5
1	2	3	4	5	6
1	в	л	ц	д	у
2	т	ж	б	ю	ц
3	х	ъ	я	р	р
4	м	ш	б	р	х
5	ц	э	о	о	э
6	ц	г	б	р	ь
7	ц	м	й	ф	к
8	т	ъ	ъ	ю	ь
9	м	ш	э	с	я
10	ц	п	у	н	у
11	я	щ	э	й	т
12	а	ь	э	д	к
13	ц	и	б	р	ь
14	ц	г	б	р	п
15	а	ч	к	ъ	у
16	ц	п	ъ	б	ь
17	с	э	г	к	ц
18	ъ	г	у	у	щ
19	а	р	ц	ё	э
20	в	ъ	р	ю	у
21	о	ю	э	к	а
22	а	э	б	р	н

№ n/n	Y_1	Y_2	Y_3	Y_4	Y_5
1	2	3	4	5	6
68	с	к	а	с	э
69	г	ъ	п	п	э
70	г	ю	к	д	о
71	й	р	с	р	э
72	в	о	о	п	ч
73	щ	ш	о	к	а
74	з	р	ъ	б	б
75	н	э	у	г	н
76	я	л	ё	к	ь
77	с	р	б	ё	у
78	ы	э	б	д	э
79	у	л	б	ю	а
80	с	ш	о	у	э
81	т	ъ	ш	к	р
82	с	д	у	г	э
83	ф	л	б	у	б
84	у	ъ	ч	н	ч
85	т	р	т	п	э
86	г	ю	к	и	у
87	г	ю	э	м	э
88	г	ю	к	к	ъ
89	ъ	п	э	г	я

№ n/n	Y_1	Y_2	Y_3	Y_4	Y_5
1	2	3	4	5	6
135	т	э	б	д	у
136	я	щ	э	у	б
137	ъ	и	б	р	ю
138	в	ъ	е	ж	а
139	г	и	б	р	б
140	а	г	б	р	ы
141	м	п	у	н	о
142	ц	ш	я	ж	ц
143	е	ч	к	ф	о
144	д	щ	о	ъ	ч
145	ж	ш	й	у	ъ
146	ц	х	ч	щ	в
147	у	э	б	д	л
148	д	ъ	э	г	я
149	с	у	а	х	з
150	ц	э	б	д	э
151	у	л	ь	к	н
152	ъ	щ	б	ж	я
153	ц	э	ь	р	ё
154	д	ъ	ь	в	ю
155	в	л	р	н	у
156	я	ф	у	о	у

1	2	3	4	5	6
23	я	ф	у	к	а
24	б	ъ	а	р	п
25	я	ъ	а	ф	к
26	ъ	и	ь	ж	я
27	ф	ф	н	й	о
28	я	ф	ы	в	б
29	н	э	н	ф	у
30	ю	г	б	р	ь
31	с	ш	ь	ж	э
32	т	б	э	ё	ч
33	ю	ъ	ю	р	ъ
34	е	г	о	ф	к
35	б	ь	ч	я	б
36	а	ш	в	ё	э
37	у	ъ	ъ	ю	а
38	д	н	ч	ж	ч
39	у	ж	ц	ё	э
40	в	л	р	н	ч
41	у	л	б	ю	п
42	ц	у	р	у	н
43	ь	ъ	ш	с	э
44	ю	ъ	з	к	ц
45	х	ъ	я	р	р
46	н	р	ю	в	я
47	с	п	э	м	а
48	с	ч	к	п	э
49	у	ж	ь	ж	ы
50	а	т	у	ф	у
51	я	р	ю	р	а

1	2	3	4	5	6
90	а	п	у	ф	у
91	э	з	ь	р	а
92	д	з	ь	ж	ч
93	ю	р	м	ф	ц
94	х	р	а	ю	ю
95	а	н	ч	ё	ч
96	ю	ъ	ы	х	ь
97	ъ	ц	о	м	э
98	ф	ъ	ц	п	о
99	и	р	ь	к	н
100	щ	п	э	т	э
101	у	з	у	я	б
102	а	щ	у	щ	б
103	а	ы	э	й	ч
104	д	ф	р	п	э
105	ц	ъ	ь	р	ь
106	ц	ъ	ц	п	о
107	и	л	у	ф	э
108	д	ц	о	й	э
109	д	я	т	р	р
110	а	ч	к	у	б
111	у	ф	н	й	т
112	а	ь	э	д	к
113	ц	к	р	н	н
114	ц	ю	а	б	у
115	г	ю	у	у	б
116	у	р	ь	п	й
117	ю	э	ъ	ж	т
118	г	ю	р	к	у

1	2	3	4	5	6
157	х	ф	е	к	ь
158	г	ц	ч	ч	ч
159	г	э	ъ	ж	т
160	а	н	о	п	ч
161	ы	н	а	ж	п
162	а	ч	к	ъ	у
163	ъ	м	э	н	к
164	й	р	э	ф	щ
165	э	ъ	ь	б	у
166	д	э	н	д	а
167	д	ъ	я	р	ь
168	е	ю	э	л	э
169	т	ч	о	у	б
170	ъ	ц	э	ф	э
171	в	л	н	ё	э
172	г	ф	д	с	э
173	в	э	ё	к	б
174	с	ч	о	у	к
175	г	а	у	т	э
176	ы	п	у	б	б
177	ц	ч	к	п	э
178	г	ю	ч	с	а
179	ъ	б	э	н	э
180	ф	ъ	р	к	а
181	ц	х	ё	в	а
182	е	т	у	ф	я
183	е	п	ь	р	ю
184	в	ъ	р	ж	а
185	д	ф	ё	ж	б

1	2	3	4	5	6
52	в	р	т	у	б
53	у	р	ь	п	э
54	щ	л	а	ф	о
55	у	ф	б	ю	а
56	ц	м	н	у	б
57	с	ю	к	й	т
58	а	ь	э	д	й
59	ю	н	о	о	э
60	г	ю	о	ж	б
61	г	к	б	р	ъ
62	н	ц	э	п	о
63	т	ч	м	ё	о
64	д	з	ц	в	б
65	ц	ш	щ	в	щ
66	е	п	ч	д	ч
67	д	р	ъ	ю	ь

1	2	3	4	5	6
119	ю	щ	о	ъ	у
120	ф	ъ	э	г	я
121	с	у	о	и	ч
122	щ	щ	ч	д	ц
123	с	ф	ы	р	э
124	д	щ	э	ъ	у
125	я	ф	ш	ё	ч
126	ц	ю	й	р	щ
127	в	я	х	в	м
128	к	р	ш	р	п
129	г	ю	о	п	э
130	у	ц	ч	й	т
131	а	ь	э	д	к
132	ц	и	б	р	ь
133	ц	ы	я	ж	т
134	ю	р	б	у	э

1	2	3	4	5	6
186	ь	ф	у	т	о
187	щ	о	я	в	ь
188	ъ	г	у	п	ч
189	р	ш	у	и	т
190	е	а	ч	й	ч
191	и	р	а	м	ч
192	ю	ф	ч	о	у
193	я	ю	о	н	к
194	я	ж	ы	к	г
195	с	ц	б	р	я
196	с	ш	ч	й	о
197	т	ъ	ъ	ж	р
198	с	щ	ч	л	

Частота повторения букв в столбцах:
1-й столбец (общее число букв $m = 198$)

Обозначение	а	б	в	г	д	е	ё	ж	з	и	й	к	л	м	н	о	п
Число	17	2	10	16	14	7	0	1	1	3	2	1	0	3	4	1	0

Обозначение	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я	
Число	1	16	9	14	5	5	23	0	0	5	10	3	2	2	10	11	

$$I_{c1}(x)=\frac{\sum_{i=0}^{n-1} f_i(f_i-1)}{m(m-1)}=$$

$$=\frac{272+2+90+240+182+42+6+12+240+72+182+}{198\times197}$$

$$\frac{+20+20+506+20+90+6+2+2+90+110}{198\times197}=0,05676$$

2-й столбец (общее число букв m = 198)

Обозначение	а	б	в	г	д	е	ё	ж	з	и	й	к	л	м	н	о	п
Число	2	2	0	7	1	0	0	4	4	5	0	3	11	3	5	2	10

Обозначение	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я	
Число	18	0	2	3	14	2	7	9	11	9	26	2	5	14	15	2	

$$I_{c1}(x)=\frac{\sum_{i=0}^{n-1} f_i(f_i-1)}{m(m-1)}=$$

$$=\frac{2+2+42+12+12+20+6+110+6+20+2+90+306+}{198\times197}$$

$$\frac{+2+6+182+2+42+72+110+72+650+2+20+182+210+2}{198\times197}=0,05896$$

3-й столбец (общее число букв m = 198)

Обозначение	а	б	в	г	д	е	ё	ж	з	и	й	к	л	м	н	о	п
Число	9	24	1	1	1	2	4	0	1	0	3	10	0	2	6	17	1

Обозначение	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я	
Число	9	1	3	19	0	1	6	14	4	1	8	4	14	23	3	6	

$$I_{c1}(x)=\frac{\sum_{i=0}^{n-1} f_i(f_i-1)}{m(m-1)}=$$

$$=\frac{72+552+2+12+6+90+2+30+272+72+6+342}{198\times197}$$

$$\frac{+30+182+12+56+12+182+506+6+30}{198\times197}=0,0634$$

4-й столбец (общее число букв m = 198)

Обозначение	а	б	в	г	д	е	ё	ж	з	и	й	к	л	м	н	о	п
Число	0	5	8	5	13	0	9	16	0	3	9	15	2	4	9	4	14

Обозначение	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я	
Число	27	5	3	13	13	2	0	1	0	2	5	0	0	0	9	2	

$$I_{c1}(x)=\frac{\sum_{i=0}^{n-1} f_i(f_i-1)}{m(m-1)}=$$

$$=\frac{20+56+20+156+72+240+2+12+72+12+182+}{198\times197}$$

$$\frac{+702+20+6+156+156+2+2+20+72+2}{198\times197}=0,0581$$

5-й столбец (общее число букв m = 197)

Обозначение	а	б	в	г	д	е	ё	ж	з	и	й	к	л	м	н	о	п
Число	15	18	1	1	0	0	1	0	1	0	2	9	1	1	6	11	5

Обозначение	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я	
Число	5	0	8	19	0	1	6	17	0	4	4	2	13	33	4	9	

$$I_{c1}(x)=\frac{\sum_{i=0}^{n-1} f_i(f_i-1)}{m(m-1)}=$$

$$=\frac{210+306+2+72+30+110+20+20+56+342+30+}{198\times197}$$

$$\frac{+272+12+12+2+156+1056+12+72}{198\times197}=0,0723$$

По полученным индексам совпадения можно утверждать, что длина ключевого слова выбрана верно и равна 5.

После того как мы нашли длину ключевого слова, произведем поиск его истинного значения. Для этого можно использовать так называемый взаимный индекс совпадения

$$MI_c(x,y)=\frac{\sum_{i=0}^{n-1} f_i \cdot f_i^1}{m \cdot m'}$$

где f_i, f_i^1 — частота буквы i в столбцах Y_i, Y_i^1 соответственно;

m, m' — число букв в столбцах Y_i, Y_i^1 соответственно;

Поскольку каждый из столбцов таблицы является результатом зашифрования фрагмента открытого текста простой заменой, определяемой подстановкой, попытаемся оценить взаимные индексы совпадения.

Взаимный индекс совпадения значения ключевого слова для русского языка должен находиться в приделах 0,053–0,07. И для его вычисления предварительно необходимо определить относительный сдвиг всех столбцов относительно первого.

Сдвиг 2-го столбца на 6 позиций:

Обозначение	а	б	в	г	д	е	ё	ж	з	и	й	к	л	м	н	о	п
Число	26	2	5	14	15	2	2	2	0	7	1	0	0	4	4	5	0

Обозначение	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я	
Число	3	11	3	5	2	10	18	0	2	3	14	2	7	9	11	9	

$MI_c(Y1, Y2^6)$= 0.05494.

Сдвиг 3-го столбца на 3 позиции:

Обозначение	а	б	в	г	д	е	ё	ж	з	и	й	к	л	м	н	о	п
Число	23	3	6	9	24	1	1	1	2	4	0	1	0	3	10	0	2

Обозначение	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я	
Число	6	17	1	9	1	3	19	0	1	6	14	4	1	8	4	14	

$MI_c(Y1, Y3^3)$ = 0,5798.

Сдвиг 4-го столбца на 16 позиций:

Обозначение	а	б	в	г	д	е	ё	ж	з	и	й	к	л	м	н	о	п
Число	27	5	3	13	13	2	0	1	0	2	5	0	0	0	9	2	0

Обозначение	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я	
Число	5	8	5	13	0	9	16	0	3	9	15	2	4	9	4	14	

$MI_c(Y1, Y4^{16})$= 0,06068.

Сдвиг 5-го столбца на 3 позиции:

Обозначение	а	б	в	г	д	е	ё	ж	з	и	й	к	л	м	н	о	п
Число	33	4	9	15	18	1	1	0	0	1	0	1	0	2	9	1	1

Обозначение	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я	
Число	6	11	5	5	0	8	19	0	1	6	17	0	4	4	2	13	

$MI_c(Y1, Y5^3)$= 0,06045.

По взаимным индексам совпадения можно судить, что сдвиги между столбцами выбраны верно.

Составим уравнения для определения ключевого слова:

g[1] – g[2]=6; g[1]=g[2] + 6; g[2]=g[1] –6;

g[1] – g[3]=3; g[1]=g[3] + 3; g[3]=g[1] –3;

g[1] – g[4]=16; g[1]=g[4] + 16; g[4]=g[1] –16;
g[1] – g[5]=3; g[1]=g[5] + 3; g[5]=g[1] – 3.

Теперь только необходимо вычислить значение g [1]:

g[l]=1: быюсю; g[l]=2: вьятя; g[l]=3: гэауа; g[l]=4: дюбфб;
g[l]=5: еявхв; g[l]=6: ёагцг; g[l]=7: жбдчд; g[l]=8: звеше;
g[l]=9: игёщё; g[l]=10: йджъж; g[l]=11: кезыз; g[l]=12: лёиьи;
g[l]=13: мжйэй; g[l]=14: нзкюк; g[l]=15: оилял; g[l]=16: пймам;
g[l]=17: ркнбн; g[l]=18: **«слово»**; g[l]=19: тмпгп; g[l]=20: унрдр;
g[l]=21: фосес; g[l]=22: хптёт; g[l]=23: цружу; g[l]=24: чсфзф;
g[l]=25: штхих; g[l]=26: щуцйц; g[l]=27: ъфчкч; g[l]=28: ыхшлш;
g[l]=29: ьщщмщ; g[l]=30: эчънъ; g[l]=31: юшыоы; g[l]=32: ящьпь.

Найдено одно ключевое слово «**СЛОВО**»

Расшифруем зашифрованный текст:

развебытьздоровымтожесамоечтонебытьбольнымопределенноздо
ровьеэтонечтобольшеедлянасфизическоездоровьеэтоисостояниеиспо
собностьиэнергиязаниматьсятемчтонамнеобходимополучатьприэтом
удовольствиеивыздоравливатьбезвсякойпомощиздоровьепарадоксал
ьновынеможетенепосредственнозаставитьсебястатьздоровымвамост
аетсятольконаблюдатьзатемкакудивительнаяспособностьвашегоорга
низмаисцелятьсебяначинаетдействоватьсамасобойивашебогатствоил
ибедностьжестокостьилидобродетельностьнеимеютздесьповидимом
уникакогозначенияздоровьеэтонечтопозитивноеононеозначаетотказо
тудовольствияздоровьеявляетсяестественнымследствиемнашегообра
зажизнивзаимоотношенийдиетыокружающейобстановкиздоровьеэто
непредметсобственностиэтопроцессэтоточтомыделаемрезультатнаш
ихмыслейичувствэтообразсуществованияинтересночтонаправлением
едицинскихисследованийвсебольшеибольшеотклоняетсявсторонуто
йобластикотораядосихпорсчиталасьсферойдеятельностипсихологови
сейчасужетруднопровестичеткиеразграничениямеждуфизическимии
ментальнымифакторамизаболеваний

Ключевое слово верное, текст читается.

Шифр Вернама. Такая система была предложена инженером компании AT&T Гилбертом Вернамом (*Gilbert Vernam*) в 1918 г. Его система оперирует не буквами, а двоичными числами. Кратко ее можно выразить формулой

$$C_i = p_i \oplus k_i ,$$

где p_i — i-я двоичная цифра открытого текста;
k_i — i-я двоичная цифра ключа;
C_i — i-я двоичная цифра шифрованного текста;
$\oplus$ — операция XOR (исключающее «ИЛИ»).

Таким образом, шифрованный текст генерируется путем побитового выполнения операции XOR для открытого текста и ключа. Благодаря свойствам этой операции, для расшифровки достаточно выполнить эту же операцию:

$$p_i = C_i \oplus k_i .$$

Сутью этой технологии является способ выбора ключа. Вернам предложил использовать закольцованную ленту, что означает циклическое повторение ключевого слова, так что его система на самом деле предполагала работу хоть и с очень длинным, но все же повторяющимся ключом. Несмотря на то что такая схема в силу очень большой длины ключа значительно усложняет задачу криптоанализа, ее, тем не менее, можно взломать, имея в распоряжении достаточно длинный фрагмент шифрованного текста, известные или вероятно известные фрагменты открытого текста, либо и то и другое одновременно.

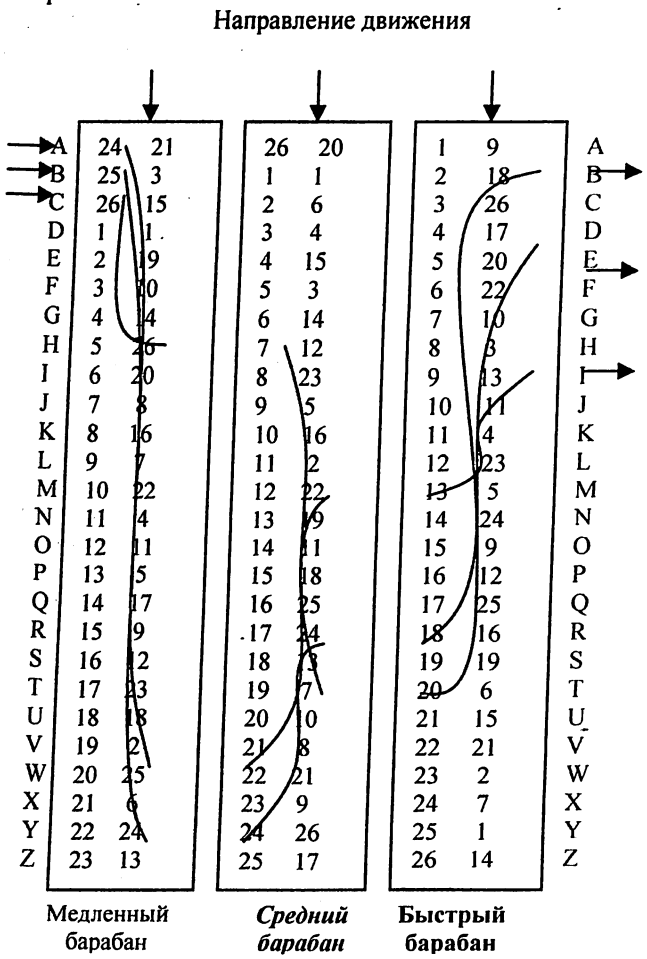
Джозеф Моборн (*Joseph Mauborgne*) предложил улучшения схемы шифрования Вернама, которые сделали ее исключительно надежной. Моборн предложил отказаться от повторений и случайным образом генерировать ключ, по длине равный длине сообщения. Такая схема, получившая название ленты однократного использования (или схемы с одноразовым блокнотом), взлому не поддается. В результате ее применения на выходе получается случайная последовательность, не имеющая статистической взаимосвязи с открытым текстом. Поскольку в этом случае шифрованный текст не дает никакой информации об открытом тексте, нет способа и взломать код.

Сложность практического применения этого метода заключается в том, что и отправитель, и получатель должны располагать одним и тем же случайным ключом и иметь возможность защитить его от посторонних. Поэтому, несмотря на все преимущества шифра Вернама перед другими шифрами, на практике к нему прибегают редко.

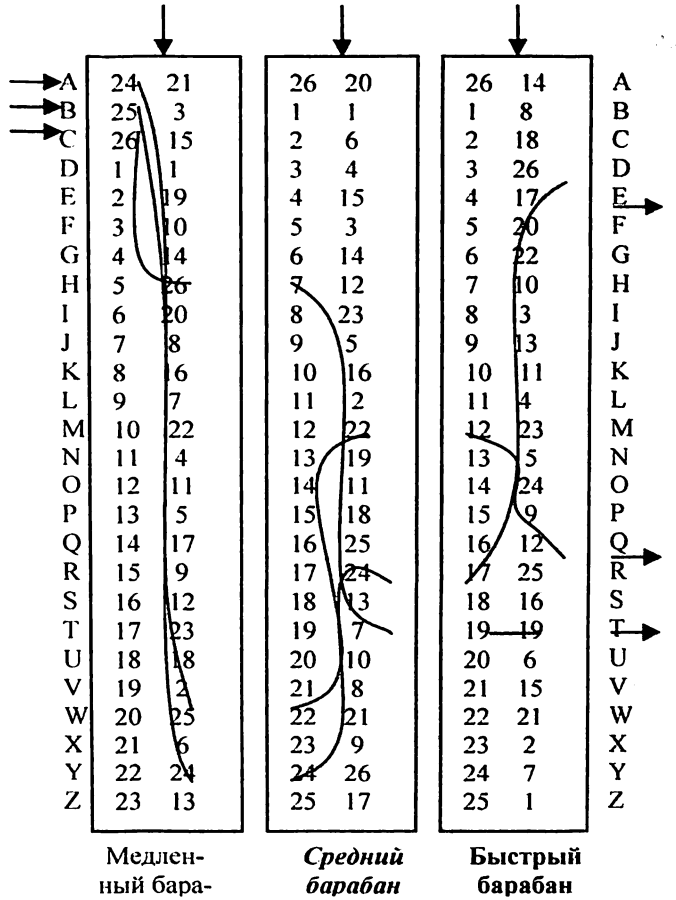
3.4. РОТОРНЫЕ ШИФРОВАЛЬНЫЕ МАШИНЫ

Вполне логично предположить, что повторное шифрование порождает алгоритм, который будет значительно сложнее для криптоанализа, чем базовый шифр. Это утверждение справедливо как для подстановочных, так и для перестановочных шифров. До того как был предложен алгоритм DES, самой важной областью практического применения принципа многопроходного шифрования были системы, использующие так называемые роторные (барабанные) шифровальные машины (*rotor machines*).

Принцип работы роторной (барабанной) шифровальной машины показан на рис. 3.5.



а — исходное состояние



б — состояние после ввода одной буквы

Рис. 3.5. Трехбарабанная шифровальная машина с системой электропроводки, представленной соответствующей нумерацией контактов

Главной частью машины является набор вращающихся барабанов (цилиндров), по которым могут проходить электрические импульсы. Каждый барабан имеет 26 входных и 26 выходных контактов, а также внутреннюю проводку, которая соединяет каждый входной контакт с соответствующим только ему выходным контактом (для на-

глядности на рисунке каждого из барабанов показано только по три внутренних соединения).

Если связать входные и выходные контакты с определенными буквами английского алфавита, то каждый барабан будет реализовывать некоторую моноалфавитную подстановку. Для схемы, изображенной на рис. 3.5, если оператор нажмет клавишу, соответствующую букве **a**, электрический сигнал поступит на первый входной контакт первого барабана и по внутренней проводке потечет к двадцать пятому выходному контакту.

Рассмотрим машину с одним барабаном. После нажатия любой клавиши барабан поворачивается на одну позицию, вследствие чего система внутренних соединений меняется. Поэтому при следующем нажатии клавиши будет использоваться уже другая моноалфавитная подстановка. После ввода 26 букв открытого текста барабан вернется в исходное положение. Это значит, что в данном случае используется полиалфавитный подстановочный алгоритм с периодом 26.

Машина с одним барабаном генерирует тривиальный шифр, который не станет большой проблемой для криптоаналитика. Преимущество барабанной шифровальной машины заключается в использовании нескольких барабанов, в которых выходные контакты одного барабана подключены к входным контактам другого. На рис. 3.5 показана трехбарабанная система. На верхней половине рис. 3.5 отражено положение, в котором полученный от оператора входной сигнал (буква **a** открытого текста), пройдя через три барабана, появляется на выходе на втором выходном контакте последнего барабана (буква **B** шифрованного текста).

При использовании нескольких барабанов тот из них, который находится дальше всего от точки ввода сигнала оператором, проворачивается на одну позицию при каждом нажатии клавиши. В нижней части рис. 3.5 показано положение системы после ввода оператором одного символа. После того как внешний (быстрый) барабан сделает полный оборот, средний цилиндр поворачивается на одну позицию. А когда на полный оборот повернется средний цилиндр, на одну позицию повернется внутренний. Точно так же работает любой одометр (например, в электросчетчике). В результате прежде чем система начинает повторяться, будет использовано $26 \times 26 \times 26 = 17\,576$ различных подстановочных алфавитов. Добавление четвертого и пятого барабанов приведет к увеличению числа задействованных алфавитов до 456 976 и 11 881 376 соответственно.

Значение роторных (барабанных) шифровальных машин в современных условиях состоит в том, что они указали путь к разработке самого популярного на сегодня шифра DES (*Data Encryption Standard*).

ГЛАВА 4. ШИФРОВАНИЕ НА ОСНОВЕ МЕТОДОВ ПЕРЕСТАНОВКИ

4.1. МЕТОДЫ ПЕРЕСТАНОВКИ

Все рассмотренные выше методы основывались на замещении символов открытого текста различными символами шифрованного текста. Принципиально иной класс преобразований строится на использовании перестановок букв открытого текста. Шифры, созданные с помощью перестановок, называют перестановочными шифрами.

Простейший из таких шифров использует преобразование «лесенки», заключающееся в том, что открытый текст записывается вдоль наклонных строк определенной длины («ступенек»), а затем считывается построчно по горизонтали. Например, чтобы шифровать сообщение «meet me after the toga party» по методу «лесенки» со ступеньками длиной 2, запишем это сообщение в виде:

m e m a t r h t g p r y
e t e f e t e o a a t

Шифрованное сообщение будет иметь следующий вид:

MEMATRHTGPRYETEFETEOAAT

Такой «шифр» особой сложности для криптоанализа не представляет. Более сложная схема предполагает запись текста сообщения в горизонтальные строки одинаковой длины и последующее считывание текста столбец за столбцом, но не порядку, а в соответствии с некоторой перестановкой столбцов. Порядок считывания столбцов при этом становится ключом алгоритма. Рассмотрим следующий пример.

Ключ:	3 4 2 1 5 6 7
Открытый текст:	a t t a c k p
	o s t p o n e
	d u n t i l t
	w o a m x y z

Шифрованный текст:

TTNAAPTMTSUOAODWCOIXKNLYPETZ

Простой перестановочный шрифт очень легко распознать, так как буквы в нем встречаются с той же частотой, что и в открытом тексте. Например, для только что рассмотренного способа шифрования

с перестановкой столбцов анализ шифра выполнить достаточно просто: необходимо записать шифрованный текст в виде матрицы и перебрать возможные варианты перестановок для столбцов. Можно использовать также таблицы значений частоты биграмм и триграмм.

Перестановочный шифр можно сделать существенно более защищенным, выполнив шифрование с использованием перестановок несколько раз. Оказывается, что в этом случае примененную для шифрования перестановку воссоздать уже не так просто. Например, если предыдущее сообщение шифровать еще раз с помощью того же самого алгоритма, то результат будет следующим.

Открытый текст:
t t n a a p t
m t s u o a o
d w c o l x k
n l y p e t z

Шифрованный текст:

NSCYAUOPTTWLTMDNAOIEPAXTTOKZ

Чтобы нагляднее представить то, что получится в итоге повторного применения перестановки, сопоставим каждую букву исходного открытого текста с номером соответствующей ей позиции. Наше сообщение состоит из 28 букв и исходной последовательностью будет последовательность:

01 02 03 04 05 06 07 08 09 10 11 12 13 14
15 16 17 18 19 20 21 22 23 24 25 26 27 28

После первой перестановки получим последовательность, которая все еще сохраняет некоторую регулярность структуры:

0310 17 24 04 11 18 25 02 09 16 23 01 08
1522 05 12 19 26 06 13 20 27 07 14 21 28

После второй перестановки получается следующая последовательность:

17 09 05 27 24 16 12 07 10 02 22 20 03 25
15 13 04 23 19 14 11 01 26 21 18 08 06 28

Регулярность этой последовательности уже совсем не просматривается, поэтому ее криптоанализ потребует значительно больше усилий.

В настоящее время методы перестановки продолжают развиваться, иногда в самом неожиданном направлении. Так, в 1991 г. В.М. Кузьмич предложил схему перестановки, основанную 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 числом 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 этой 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.

4.2. БЛОЧНЫЕ ШИФРЫ

Практически все алгоритмы симметричного блочного шифрования, используемые в наши дни, основаны на структуре, получившей название **«блочный шифр Файстеля»** (Feistel block cipher). В этой связи очень важно понять принципы, на которых построен шифр Файстеля.

Блочными называются шифры, в которых логической единицей шифрования является некоторый блок открытого текста, после пре-

образований которого получается блок шифрованного текста такой же длины. Обычно используются блоки размером 64 бита.

Большинство сетевых приложений, в которых применяется схема традиционного шифрования, использует блочные шифры.

Блочный шифр предполагает преобразование n-битового блока открытого текста в блок шифрованного текста такого же размера. Число различных блоков при этом равно 2n, и чтобы шифрование было обратимым (т.е. чтобы обеспечивалась возможность дешифрования), каждый из таких блоков должен преобразовываться в свой уникальный блок шифрованного текста. Такие преобразования называются обратимыми, или несингулярными.

Вот примеры несингулярного и сингулярного преобразований для n = 2:

Открытый текст	*Шифрованный текст*	*Открытый текст*	*Шифрованный текст*
00	11	00	11
01	10	01	10
10	00	10	01
11	01	11	01

В данном примере в случае необратимого отображения шифрованный текст соответствует двум разным блокам открытого текста. Если ограничиться рассмотрением только обратимых отображений, число различных допустимых преобразований окажется равным $2^n!$.

На рис. 4.1 показана общая схема подстановочного шифра для $n = 4$. Поступающее на вход 4-битовое значение определяет одно из 16 возможных начальных состояний, которое отображается подстановочным шифром в одно из 16 конечных состояний, каждое из которых представляется 4-битовым отображением шифрованного текста. Схемы шифрования и дешифрования можно представить в виде таблиц (табл. 4.1). Это самая общая схема процесса блочного шифрования, которая может служить для разработки любого обратимого соответствия между открытым и шифрованным текстом.

Однако при практической реализации данного подхода возникает следующая проблема. Если использовать блок небольшого размера, например, $n = 4$, система оказывается эквивалентной классическому подстановочному шифру. Как уже было отмечено ранее, такие системы уязвимы и могут быть раскрыты с помощью статистического анализа открытого текста. Этот недостаток порождается не самой природой подстановочного шифра, а использованием коротких блоков. При достаточно больших значениях n и при отсутствии каких-либо ограничений на обратимую подстановку, задающую преобразование открытого текста в шифрованный, статистические характеристики исходного открытого текста маскируются настолько хорошо, что такого рода криптоанализ становится практически бесполезным.

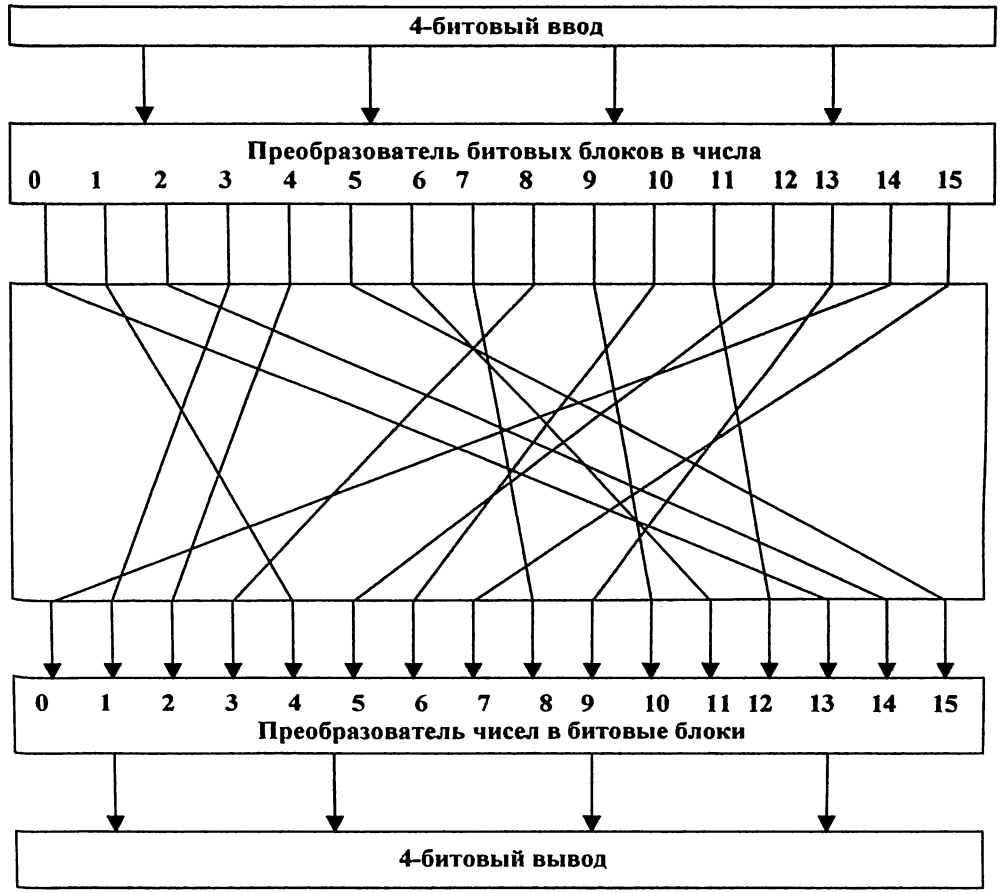


Рис. 4.1. Общая схема поблочной подстановки n-битовых блоков (случай $n = 4$)

В то же время предположение о допустимости любых обратимых подстановок при больших размерах блоков оказывается весьма непрактичным в отношении реализации алгоритма и скорости выпол-

нения соответствующего приложения. Для такого преобразования ключом является само отображение.

Таблица 4.1

Таблицы шифрования и дешифрования для блочного шифра (рис. 4.1)

Открытый текст	*Шифрованный текст*	*Открытый текст*	*Шифрованный текст*
0000	1110	0000	1110
0001	0100	0001	0011
0010	1101	0010	0100
0011	0001	0011	1000
0100	0010	0100	0001
0101	1111	0101	1100
0110	1011	0110	1010
0111	1000	0111	1111
1000	0011	1000	0111
1001	1010	1001	1101
1010	0110	1010	1001
1011	1100	1011	0110
1100	0101	1100	1011
1101	1001	1101	0010
1110	0000	1110	0000
1111	0111	1111	0101

Обратимся снова к табл. 4.1, которая определяет конкретное обратимое отображение пространства открытых текстов в пространство шифрованных для $n = 4$. Отображение можно задать элементами из второго столбца таблицы, в котором приведены значения шифрованного текста для соответствующих значений открытого текста. Это по сути и есть ключ, который отличает данное конкретное отображение от всех других допустимых отображений. В рассматриваемом случае длина ключа оказывается равной 64. Обычно для блочного шифра с n-битовым подстановочным блоком размер ключа равен $n \times 2^n$.

Занимаясь изучением данной проблемы, Файстель предложил для больших n аппроксимировать такую идеальную систему блочного шифрования набором простых для реализации компонентов. Но прежде чем приступить к непосредственному обсуждению предложенного Файстелем подхода, необходимо обратить внимание на следующий момент. Невозможно позволить себе работать с блочным подстановочным шифром общего вида, но для реализации соответствующего алгоритма программным методом можно вместо

всех $2^n!$ допустимых отображений рассмотреть некоторое более узкое подмножество.

Например, предположим, что отображения задаются системой линейных уравнений. Для $n = 4$ получим:

$$У_1 = К_{11}X_1 + К_{12}X_2 + К_{13}X_3 + К_{14}X_4,$$
$$У_2 = К_{21}X_1 + К_{22}X_2 + К_{23}X_3 + К_{24}X_4,$$
$$У_3 = К_{31}X_1 + К_{32}X_2 + К_{33}X_3 + К_{34}X_4,$$
$$У_4 = К_{41}X_1 + К_{42}X_2 + К_{43}X_3 + К_{44}X_4,$$

где X_i обозначает четыре двоичные цифры блока открытого текста;

$У_i$ — четыре двоичные цифры блока шифрованного текста;

$К_{ij}$ — двоичные коэффициенты, а все арифметические операции выполняются по модулю 2.

Длина ключа равна n^2, и в данном случае это всего 16 битов. Опасность задания отображений формулами заключается в том, что в случае, когда структура такого алгоритма оказывается доступной криптоаналитику, система может стать весьма уязвимой. В данном примере система шифрования, по сути, эквивалентна шифру Хилла, но в применении не к текстовым символам, а к двоичным данным. Подобные простые линейные системы защищены весьма слабо.

Шифр Файстеля. В своей работе Файстель предложил аппроксимировать подстановочный шифр продукционными шифрами, которые строятся на последовательном применении, как минимум, двух базовых шифров, с тем чтобы полученный результат обладал более высокой стойкостью по сравнению с любым из этих базовых шифров в отдельности. В частности, Файстель предложил шифр, в котором попеременно используются и подстановки, и перестановки. Фактически данное предложение является практическим применением идеи Клода Шеннона (*Claude Shannon*) разработать продукционный шифр, в котором попеременно использовались бы функции диффузии и конфузии. При этом весьма примечателен тот факт, что структура шифра Файстеля, разработанная четверть века тому назад и, в свою очередь, опирающаяся на идею Шеннона, высказанную еще в 1945 г., сегодня является структурой, на основе которой построены все практически значимые симметричные схемы блочного шифрования.

Диффузия и конфузия. Термины диффузия (*diffusion*) и конфузия (*confusion*) были введены в шифрование Клодом Шенноном для того, чтобы охарактеризовать два основных строительных блока криптографических систем. Основной задачей, которую ставил перед собой Шеннон, было воспрепятствовать попыткам криптоанализа, основанным на статистическом анализе сообщений. Шен-

нон использовал следующие аргументы. Предположим, противник обладает некоторыми сведениями о статистических характеристиках открытого текста. Например, для обычных текстовых сообщений могут быть известны данные о распределении частот использования букв соответствующего языка. Или могут быть известны отдельные слова либо фразы, вероятность присутствия которых в сообщении достаточно высока (вероятные слова). Если подобные статистически выявленные закономерности каким-либо образом проявятся в шифрованном тексте, у криптоаналитика появится возможность определить ключ шифрования или его составляющие, или хотя бы сузить пространство вероятных ключей. Шеннон вводит понятие идеального шифра — шифра, который полностью скрывает в шифрованном тексте все статистические закономерности открытого текста.

Примером такого шифра является рассмотренный ранее (см. рис. 4.1) перестановочный шифр с произвольной подстановкой, но такой шифр практически нереализуем.

Помимо обращения к идеальным системам, Шеннон предложил два метода, задачей которых является затруднение криптоанализа: диффузию и конфузию. Суть диффузии заключается в рассеянии статистических особенностей открытого текста по широкому диапазону статистических характеристик шифрованного текста. Это достигается тем, что значение каждого элемента открытого текста влияет на значения многих элементов шифрованного текста или, что эквивалентно сказанному, любой из элементов шифрованного текста зависит от множества элементов открытого текста.

Примером применения метода диффузии является шифрование сообщения $M = m_1, m_2, m_3, \ldots$ с помощью операции усреднения:

$$y_n = \sum_{i=1}^{k} m_{n+i} \pmod{26},$$

когда для получения буквы y_n шифрованного текста складываются k последовательных букв открытого текста. Можно показать, что в этом случае статистические характеристики открытого текста «распределяются» по шифрованному тексту. Поэтому в шифрованном тексте частотные характеристики использования букв будут более близки к равномерным. Точно так же более близкими к равномерным будут частоты биграмм, триграмм и т.д. В блочных шифрах, имеющих дело с двоичными данными, диффузии можно достичь путем нескольких последовательных перестановок данных с последующим применением к результату перестановки некоторой функции — в итоге в формировании каждого бита шифрованного текста будет участвовать множество битов открытого текста.

В любом блочном шифре используется зависящее от ключа преобразование блока открытого текста в блок шифрованного текста. Механизм диффузии призван сделать статистическую взаимосвязь между открытым и шифрованным текстом как можно сложнее, чтобы в максимальной степени усложнить задачу определения ключа из такой взаимосвязи.

Что касается конфузии, то перед ней ставится задача в максимальной степени усложнить статистическую взаимосвязь между шифрованным текстом и ключом — опять же с целью противостояния попыткам определить ключ. Таким образом, даже если противник сумеет определить какие-либо статистические особенности шифрованного текста, сложность использования ключа для получения шифрованного текста должна оказаться достаточной для того, чтобы попытки определить ключ на основании этих статистических особенностей оказались безрезультатными. Это достигается использованием сложных подстановочных алгоритмов: простые линейные подстановочные функции увеличивают беспорядок лишь в незначительной степени.

Понятия диффузии и конфузии оказались настолько удачными в отношении описания сути желаемых характеристик блочных шифров, что эти термины стали базовыми для всех разработчиков современных шифров этого типа.

Структура шифра Файстеля. На рис. 4.2 показана структура шифра, предложенного Файстелем. На вход алгоритма шифрования подается блок открытого текста длиной $2w$ битов и ключ K. Блок открытого текста разделяется на две равные части L_0 и R_0, которые последовательно проходят через n раундов обработки, а затем объединяются снова для получения блока шифрованного текста соответствующей длины.

Для раунда i в качестве входных данных выступают L_{i-1} и R_{i-1}, полученные на выходе предыдущего, $(i-1)$-го раунда, и подключ K_i, вычисляемый по общему ключу K. Как правило, подключи K_i отличаются как от общего ключа K, так и один от другого. Все раунды обработки проходят по одной и той же схеме.

Сначала для *левой* половины блока данных выполняется операция подстановки. Она заключается в применении к *правой* половине блока данных некоторой функции раунда F и последующем сложении полученного результата с левой половиной блока данных с помощью операции XOR (исключающего «ИЛИ»). Для всех раундов функция раунда имеет одну и ту же структуру, но зависит от параметра — подключа раунда K_i.

После подстановки выполняется перестановка, представляющая собой обмен местами двух половин блока данных.

Вся эта структура в целом является частным случаем так называемой подстановочно-перестановочной схемы (*substitution-permutation network*), предложенной Шенноном.

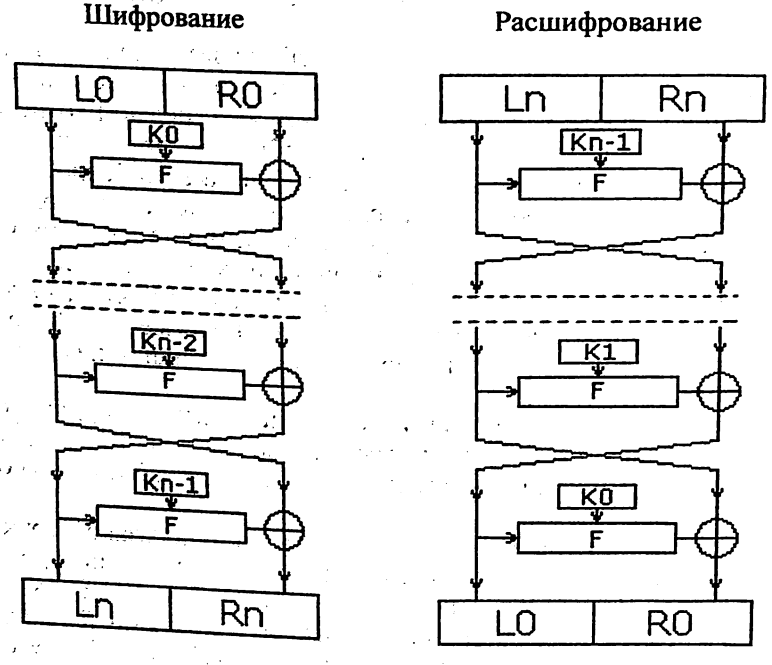


Рис. 4.2. Классическая схема Файстеля

Конкретная реализация схемы Файстеля зависит от выбора значений следующих параметров и конструктивных особенностей.

1. *Размер блока.* Чем больше размер блока, тем выше надежность шифра (при прочих равных условиях), но тем ниже скорость выполнения операций шифрования и дешифрования. Разумным компромиссом является блок размером 64 бита, который является сегодня практически универсальным для всех блочных шифров.

2. *Размер ключа.* Чем длиннее ключ, тем выше надежность шифра, но большая длина ключа тоже может быть причиной слишком медленного выполнения операций шифрования и дешифрования. На сегодняшний день ключи длиной 64 бита и меньше считаются недостаточно надежными — обычно используются 128-битовые ключи.

3. *Число раундов обработки.* Суть идеи шифра Файстеля состоит в том, что за один раунд обработки данных обеспечивается недостаточно высокая надежность, но уровень надежности шифра повышается с каждым новым раундом обработки. Как правило, число раундов выбирают равным 16.

4. *Алгоритм вычисления подключей.* Чем сложнее этот алгоритм, тем в большей степени затрудняется криптоанализ шифра.

5. *Функция раунда.* Здесь, опять же, усложнение, как правило, ведет к повышению стойкости шифра с точки зрения криптоанализа.

Кроме того, имеются еще два фактора, которые приходится принимать во внимание при построении любой реализации шифра Файстеля.

6. *Скорость выполнения программ шифрования / дешифрования.* Часто функции шифрования встраиваются в приложения или утилиты во избежание необходимости аппаратной реализации шифрования. В таких случаях важным фактором оказывается скорость программного выполнения алгоритма.

7. *Простота анализа.* Несмотря на то что основной задачей является создание такого алгоритма, который для криптоанализа был бы настолько сложным, насколько это возможно, все же весьма выгодно, чтобы сам такой алгоритм оставался простым для понимания. Иными словами, если алгоритм прост и понятен, его легче анализировать с позиций уязвимости для криптоанализа и, следовательно, легче совершенствовать с целью достижения более высокого уровня надежности. Алгоритм DES, например, нельзя отнести к алгоритмам, чрезмерно простым для анализа.

Алгоритм дешифрования. Процесс дешифрования шифра Файстеля принципиально не отличается от процесса шифрования. Применяется тот же алгоритм, но на вход подается шифрованный текст, а подключи K_i используются в обратной последовательности: для первого раунда берется подключ K_n, для второго — K_{n-1}, и так далее, пока не будет введен ключ K_1 для последнего раунда. Это свойство данной схемы шифрования оказывается очень удобным, так как для дешифрования не требует вводить второй алгоритм, отличный от алгоритма шифрования.

Чтобы убедиться в том, что тот же алгоритм с обратным порядком выбора ключей приводит к нужному результату (расшифрованному тексту), рассмотрим рис. 4.3, на котором слева изображена схема шифрования из 16 раундов, а справа — соответствующая ей схема дешифрования (результат не должен зависеть от числа раундов, используемых в процессе шифрования данных). Для простоты

обозначим данные, обрабатываемые алгоритмом шифрования, как LE_i и RE_i, а данные, обрабатываемые алгоритмом дешифрования, — как LD_i и RD_i. Из схемы видно, что на каждой стадии процесса дешифрования очередное промежуточное значение с точностью до перестановки половинок этого значения равно промежуточному значению, получаемому на соответствующей стадии процесса шифрования. Чтобы показать это, обозначим выходное значение i-го раунда шифрования через $LE_i \parallel RE_i$ (что означает конкатенацию L_i и R_i). Соответственно, входное значение для $(16 - i)$-го раунда дешифрования будет записываться в виде $LD_i \parallel RD_i$.

Далее с помощью схемы, показанной на рис. 4.2, убедимся в справедливости высказанных ранее утверждений. По окончании последнего раунда процесса шифрования две половины блока данных меняются местами, поэтому блок данных будет иметь вид $RE_{16}||LE_{16}$. Этот блок данных и будет представлять собой шифрованный текст. Возьмем этот шифрованный текст и используем его в качестве входных данных того же самого алгоритма. Тогда на входе первого раунда алгоритма получаем значение $RE_{16}||LE_{16}$, равное значению результата 16-го раунда шифрования с переставленными 32-битовыми половинами.

Теперь покажем, что результат первого раунда процесса дешифрования после 32-битового обмена будет равен значению, поступающему на вход 16-го раунда шифрования. Сначала рассмотрим процесс шифрования. Здесь мы имеем:

$$LE_{16}=RE_{15},$$
$$RE_{16}=LE_{15} \oplus F(RE_{15},K_{16}).$$

С другой стороны, для процесса дешифрования получаем:

$$LD_1=RD_0=LE_{16}=RE_{15},$$
$$RD_1=LD_0 \oplus F(RD_0,K_{16})=RE_{16} \oplus F(RE_{15},K_{16})=[LE_{15} \oplus F(RE_{15},K_{16})] \oplus$$
$$\oplus F(RE_{15},K_{16}).$$

Для последующего анализа необходимо иметь в виду, что операция XOR (исключающее «ИЛИ») имеет следующие свойства:

$$[A \oplus B] \oplus C = A \oplus [B \oplus C],$$
$$D \oplus D = 0, E \oplus 0 = E.$$

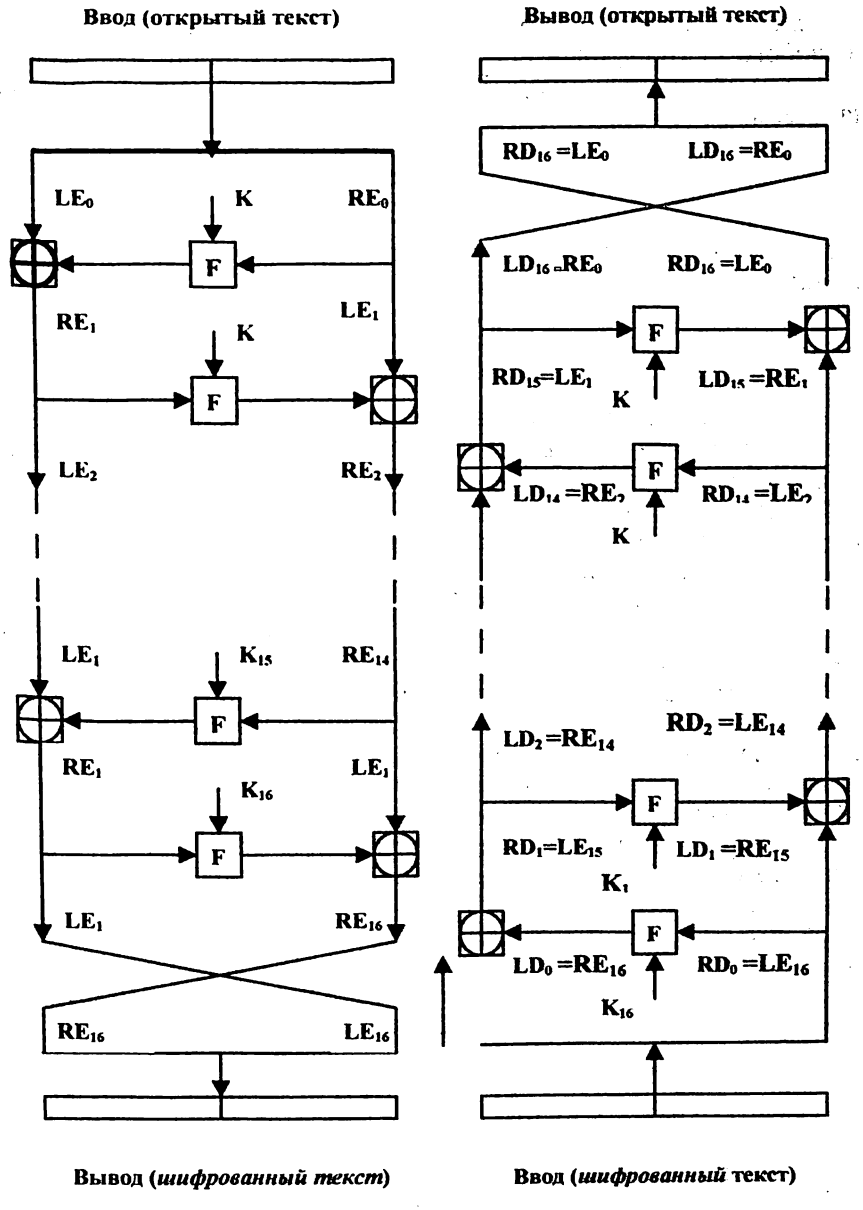


Рис. 4.3. Шифрование и дешифрование по Файстелю

Несложно показать, что такое соответствие будет выполняться для любого из 16 раундов процесса. Для самого процесса можно использовать общую формулу. Для i-й итерации алгоритма шифрования имеем:

$$LE_i = RE_{i-1},$$
$$RE_i = LE_{i-1} \oplus F(RE_{i-1}, K_i).$$

Выполнив несложные преобразования, получим следующие выражения:

$$RE_{i-1} = LE_i,$$
$$LE_{i-1} = RE_i \oplus F(RE_{i-1}, K_i) = RE_i \oplus F(LE_i, K_i).$$

Таким образом, входные данные для i-й итерации выражаются в виде функций от выходных данных, и полученные результаты подтверждают равенства, приведенные на рис. 4.2.

Наконец, заметим, что на выходе последнего раунда процесса дешифрования получается значение $RE_0 \| LE_0$. Для восстановления исходного открытого текста остается лишь 32-битовый обмен, что завершает доказательство пригодности схемы Файстеля для процесса дешифрования.

Следует обратить внимание на то, что приведенное доказательство не требует, чтобы функция F была обратимой. Чтобы убедиться в этом, достаточно рассмотреть частный случай, когда результат применения функции F является константой, т.е. не зависит от поступающих на вход значений двух ее аргументов (например, всегда равен 1). Приведенные выше равенства останутся справедливыми.

Структура шифров Файстеля обладает рядом достоинств:

процедуры шифрования и расшифрования совпадают, за исключением того, 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 существует, 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 преобразований, зависящих от ключа. В этом случае на каждой итерации изменяется весь блок и, соответственно, общее число 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. К 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.

4.3. 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 встречающихся в системах криптографической защиты информации:

– электронная кодировочная книга (*ECB — Electronic Code Book*);

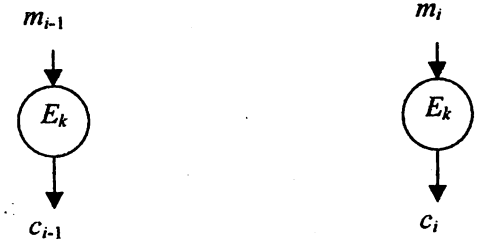
– сцепление блоков шифрованного текста (*CBC — Cipher Block Chaining*);

– обратная связь по шифрованному тексту (*CFB — Cipher Feedback*);

– обратная связь по выходу (*OFB — Output Feedback*).

В режиме электронной кодировочной книги каждый блок исходного текста шифруется блочным шифром независимо от других (рис. 4.4).

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.

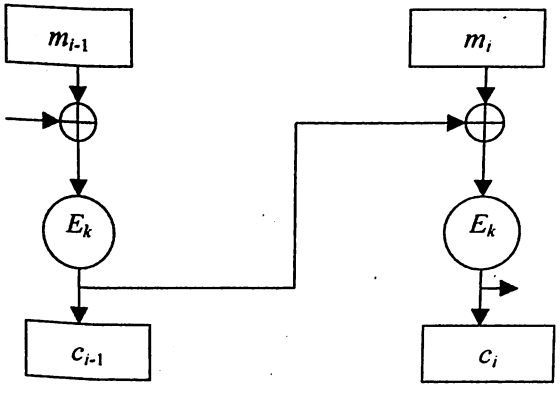


$c_i = E_k(m_i)$; $m_i = D_k(c_i)$

Рис. 4.4. Режим электронной кодировочной книги

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, определенному в ГОСТе 28147-89.

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. 4.5). Для начала процесса шифрования используется синхропосылка (или начальный вектор), которая передается в канал связи в открытом виде.



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

Рис. 4.5. Режим сцепления блоков шифрованного текста

Стойкость режима *CBC* равна стойкости блочного шифра, лежащего в его основе. Кроме того, структура исходного текста скрывается вследствие 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 простой способ 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.

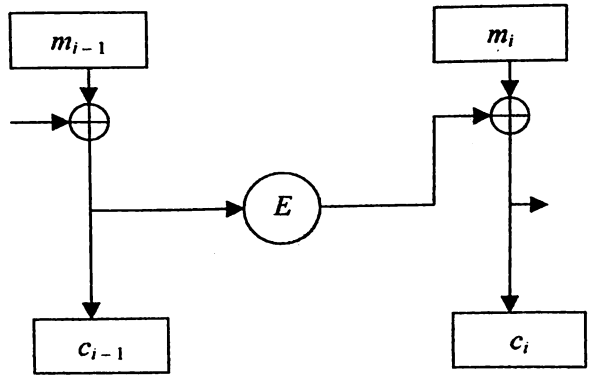
Известны модификации режима *CBC*, состоящие в следующем:

1. 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:

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

2. 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 2 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. 4.6).



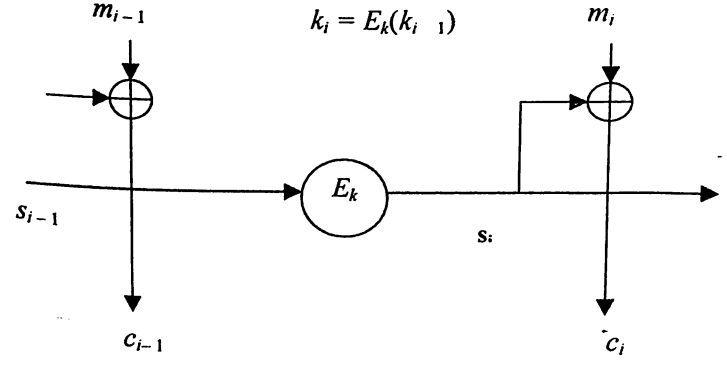
$$c_i = m_i \oplus E_k(c_{i-1});\ m_i = E_k(c_{i-1}) \oplus c_i$$

Рис. 4.6. Режим обратной связи по шифрованному тексту

Стойкость режима CFB равна стойкости блочного шифра, лежащего в его основе, и структура исходного текста скрывается в результате 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 по 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. 4.7). 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» — в этом случае на каждом шаге меняется также и ключ шифрования:



$$c_i = m_i \oplus s_i;\ m_i = c_i \oplus s_i;\ s_i = E_k(s_{i-1})$$

Рис. 4.7. Режим обратной связи по выходу

4.4. КОМПОЗИЦИОННЫЕ МЕТОДЫ ШИФРОВАНИЯ. КОМПОЗИЦИИ (КОМБИНАЦИИ) ШИФРОВ

Шифрование композиционными (комбинированными) методами основывается на результатах, полученных К. Шенноном. Наиболее часто применяются следующие комбинации:

– подстановка и гамма;
– перестановка и гамма;
– подстановка и перестановка;
– гамма и гамма.

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

Неверный выбор шифров-компонентов при составлении комбинированных шифров может привести к исходному открытому тексту. Поэтому необходимо проявлять осторожность.

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 $2n$;

второй преобразователь осуществляет свертку этого числа.

Блок перестановки осуществляет преобразование n-разрядного входного числа в другое 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, что позволило достичь скорости шифрования до 100 Кбайт/с. Опыт, полученный при разработке и эксплуатации этой системы, позволил создать стандарт шифрования данных DES (*Data Encryption Standard*).

ГЛАВА 5. СТАНДАРТ ШИФРОВАНИЯ ДАННЫХ DES (*DATA ENCRYPTION STANDARD*)

5.1. ИСТОРИЯ СОЗДАНИЯ СТАНДАРТА DES

Стандарт шифрования данных DES (*Data Encryption Standard*), который *ANSI* называет Алгоритмом шифрования данных (DEA) (*Data Encryption Algorithm*), а *ISO* — DEA-1, за 20 лет стал мировым стандартом. Он весьма стойко выдержал годы криптоанализа и все еще остается безопасным по отношению ко всем врагам.

В начале 70-х гг. невоенные криптографические исследования были крайне редки. В этой области практически не публиковались исследовательские работы. Большинство людей знали, что для своих коммуникаций военные используют специальную аппаратуру кодирования, но мало кто разбирался в криптографии как в науке. Заметными знаниями обладало Агентство национальной безопасности (*National Security Agency, NSA*), но оно даже не признавало публично своего собственного существования.

Покупатели не знали, что они покупают. Многие небольшие компании изготавливали и продавали криптографическое оборудование, преимущественно — заокеанским правительствам. Все это оборудование отличалось друг от друга и не могло взаимодействовать. Никто не знал, действительно ли какое-либо из этих устройств безопасно, не существовало независимой организации, которая засвидетельствовала бы безопасность. Как говорилось в одном из правительственных докладов:

«Влияние соответствующего изменения ключей и принципов работы на реальную мощь аппаратуры шифрования / дешифрования было (и фактически осталось) неизвестным почти всем покупателям, и было очень трудно принимать обоснованные решения о генерации ключей, правильном диалоговом или автономном режиме, и т.д., которые отвечали бы потребностям покупателей в безопасности».

В 1972 г. Национальное бюро стандартов (*National Bureau of Standards, NBS*), теперь называющееся Институтом стандартов и технологий (*National Institute of Standards and Technology, NIST*), выступило инициатором программы защиты линий связи и компьютерных данных. Одной из целей этой программы была разработка единого – стандартного — криптографического алгоритма. Этот алгоритм мог бы быть проверен и сертифицирован, а использующие его различные криптографические устройства могли бы взаимодействовать. Он мог бы, к тому же, быть относительно недорогим и легко доступным.

15 мая 1973 г. *NBS* опубликовало в *Federal Register* требования к криптографическому алгоритму, который мог бы быть принят в качестве стандарта. Было приведено несколько критериев оценки проекта:

1. Алгоритм должен обеспечивать высокий уровень безопасности.
2. Алгоритм должен быть полностью определен и легко понятен.
3. Безопасность алгоритма должна основываться на ключе и не должна зависеть от сохранения в тайне самого алгоритма.
4. Алгоритм должен быть доступен всем пользователям.
5. Алгоритм должен позволять адаптацию к различным применениям.
6. Алгоритм должен позволять экономичную реализацию в виде электронных приборов.
7. Алгоритм должен быть эффективным в использовании.
8. Алгоритм должен предоставлять возможности проверки.
9. Алгоритм должен быть разрешен для экспорта.

Реакция общественности показала, что к криптографическому стандарту существует заметный интерес, но опыт в этой области чрезвычайно мал. Ни одно из предложений не удовлетворяло предъявленным требованиям.

27 августа 1973 г. *NBS* опубликовало в *Federal Register* повторное предложение. Наконец, у Бюро появился подходящий кандидат: алгоритм под именем «ЛЮЦИФЕР», в основе которого лежала разработка компании *IBM*, выполненная в начале 70-х. Несмотря на определенную сложность, алгоритм был прямолинейный. Он использовал только простые логические операции над небольшими группами битов и мог быть довольно эффективно реализован на аппаратуре.

NBA попросила *NSA* помочь оценить безопасность алгоритма и определить, подходит ли он для использования в качестве федерального стандарта. IBM уже получила патент, но желала сделать свою интеллектуальную собственность доступной для производства, реализации и использования другими компаниями. В конце концов *NBS* и *IBM* выработали соглашение, по которому NBS получало неисключительную бесплатную лицензию изготавливать, использовать и продавать устройства, реализующие этот алгоритм. Наконец, 17 марта 1975 г. NBS опубликовало в *Federal Register* и подробности алгоритма, и заявление *IBM* о предоставлении неисключительной бесплатной лицензии на алгоритм, а также предложило присылать комментарии по поводу данного алгоритма.

Несмотря на критику, Стандарт шифрования данных DES был 23 ноября 1976 г. принят в качестве федерального стандарта и разрешен к использованию на всех несекретных правительственных

коммуникациях. Официальное описание стандарта было опубликовано 15 января 1977 г. и вступило в действие шестью месяцами позже.

Американский национальный институт стандартов (*American National Standards Institute, ANSI*) одобрил DES в качестве стандарта для частного сектора в 1981 г., назвав его Алгоритмом шифрования данных (*Data Encryption Algorithm, DEA*).

5.2. СТРУКТУРА DES

DES представляет собой блочный шифр, он шифрует данные 64-битовыми блоками. С одного конца алгоритма вводится 64-битовый блок открытого текста, а с другого конца выходит 64-битовый блок шифротекста. DES является симметричным алгоритмом: для шифрования и дешифрирования используются одинаковые алгоритм и ключ (за исключением небольших различий в использовании ключа).

Длина ключа равна 56 битам. Ключ обычно представляется 64-битовым числом, но каждый восьмой бит используется для проверки четности и игнорируется. Биты четности являются наименьшими значащими битами байтов ключа. Ключ, который может быть любым 56-битовым числом, можно изменить в любой момент времени. Ряды чисел считаются слабыми ключами, но их можно легко избежать. Безопасность полностью определяется ключом.

На простейшем уровне алгоритм не представляет ничего большего, чем комбинация двух основных методов шифрования: смещения и диффузии.

Фундаментальным строительным блоком DES является применение к тексту единичной комбинации этих методов (подстановка, а за ней — перестановка), зависящей от ключа. Такой блок называется **этапом**.

DES состоит из 16 этапов, одинаковая комбинация методов применяется к открытому тексту 16 раз.

Алгоритм использует только стандартную арифметику 64-битовых чисел и логические операции, поэтому он легко реализовывался в аппаратуре второй половины 70-х. Изобилие повторений в алгоритме делает его идеальным для реализации в специализированной микросхеме. Первоначальные программные реализации были довольно неуклюжи, но сегодняшние программы намного лучше.

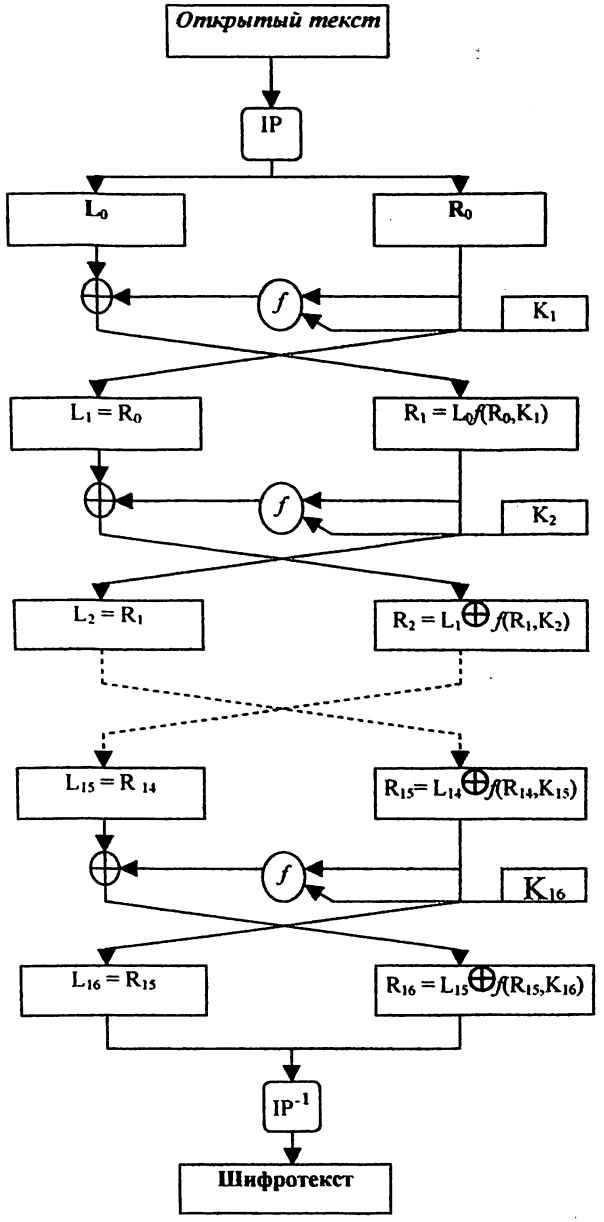


Рис. 5.1. Алгоритм DES [41]

5.2.1. СХЕМА АЛГОРИТМА

DES работает с 64-битовым блоком открытого текста:

– после первоначальной перестановки;

– блок разбивается на правую и левую половины длиной по 32 бита;

– затем выполняется 16 этапов одинаковых действий, называемых функцией *f*, в которых данные объединяются с ключом;

– после шестнадцатого этапа правая и левая половины объединяются;

– алгоритм завершается заключительной перестановкой (обратной по отношению к первоначальной).

На каждом этапе биты ключа сдвигаются, затем из 56 битов ключа выбирается 48 битов.

Функцией *f* выполняются четыре операции:

– правая половина данных увеличивается до 48 битов с помощью перестановки с расширением;

– объединяется посредством XOR («исключающее ИЛИ», суммирование по модулю 2) с 48 битами смещенного и переставленного ключа;

– проходит через восемь *S*-блоков, образуя 32 новых бита;

– переставляется снова.

Затем результат функции *f* объединяется с левой половиной с помощью XOR. В итоге этих действий появляется новая правая половина, а старая правая половина становится новой левой. Эти действия повторяются 16 раз, образуя 16 этапов DES.

Если L_i и R_i — левая и правая половины кода – результата i-той итерации, P_i — 48-битовый ключ для этапа i, а f — это функция, выполняющая все подстановки, перестановки и XOR с ключом, то этап можно представить так:

$$L_i = R_{i-1},$$

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

5.2.2. НАЧАЛЬНАЯ ПЕРЕСТАНОВКА

Начальная перестановка выполняется еще до этапа 1, при этом входной блок переставляется, как показано в табл. 5.1. Эту и все другие таблицы надо читать слева направо и сверху вниз. Например, начальная перестановка перемещает бит *58* в битовую позицию 1, бит *50* — в битовую позицию 2, бит *42* — в битовую позицию 3 и так далее.

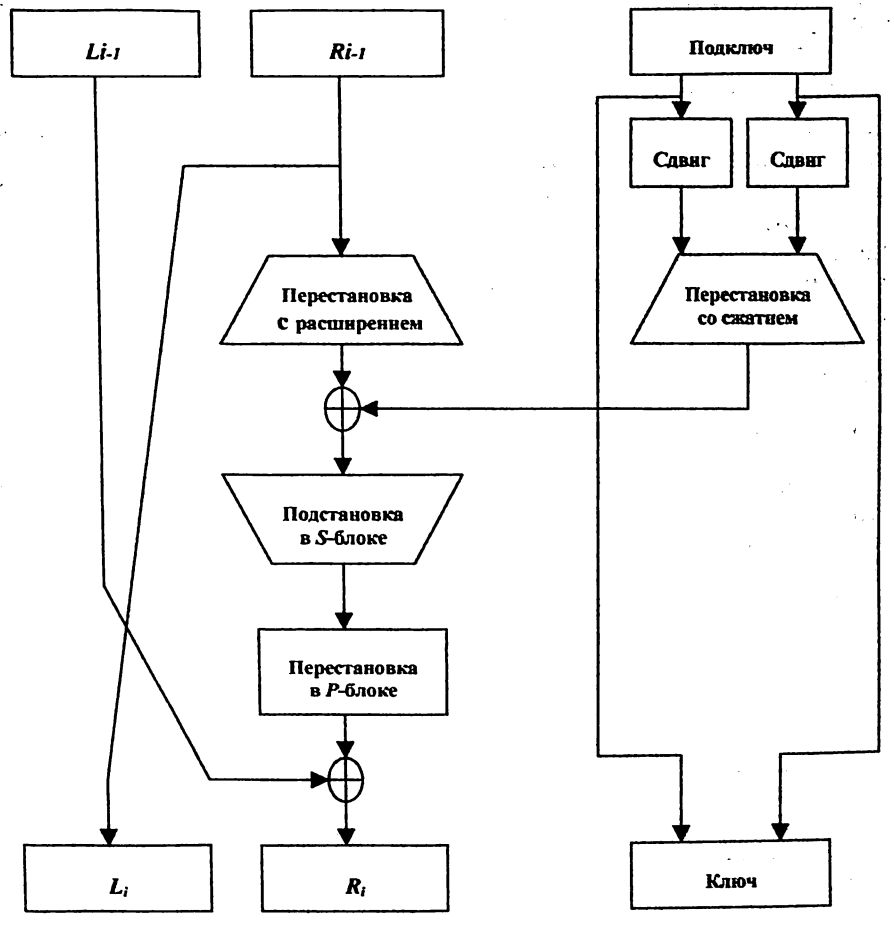


Рис. 5.2. Один этап DES

Таблица 5.1

Начальная перестановка

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

Начальная перестановка и соответствующая заключительная перестановка не влияют на безопасность DES. Как можно легко заметить, эта перестановка в первую очередь служит для облегчения побайтной загрузки данных открытого текста и шифротекста в микросхему DES. Поскольку программная реализация этой многобитовой перестановки нелегка (в отличие от тривиальной аппаратной), во многих программных реализациях DES начальная и заключительная перестановки не используются. Хотя такой алгоритм не менее безопасен, чем DES, он не соответствует стандарту DES и поэтому не может называться DES.

5.2.3. ПРЕОБРАЗОВАНИЯ КЛЮЧА

1. Сначала 64-битовый ключ DES уменьшается до 56-битового ключа отбрасыванием каждого восьмого бита. (Эти биты используются только для контроля четности, позволяя проверять правильность ключа).
2. После извлечения 56-битового ключа для каждого из 16 этапов DES генерируется новый 56-битовый подключ. Эти подключи K_i определяются согласно табл. 5.2.

Таблица 5.2

Перестановка ключа – формирование подключа

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

3. Далее 56-битовый подключ делится на две 28-битовых половинки.
4. Затем эти половинки циклически сдвигаются налево на один или два бита в зависимости от этапа. Этот сдвиг показан в табл. 5.3.

Таблица 5.3

Число битов сдвига ключа в зависимости от этапа

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

5. После сдвига выбирается 48 из 56 битов. Поскольку при этом не только выбирается подмножество битов, но и изменяется их порядок, эта операция называется перестановкой со сжатием. Ее результатом является набор из 48 битов. Перестановка со сжатием (также называемая переставленным выбором) представлена в табл. 5.4. Например, бит сдвинутого ключа в позиции **33** перемещается в позицию **35** результата, а 18-ый бит сдвинутого ключа отбрасывается.

Таблица 5.4

Перестановка со сжатием

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	32	37	47	55	30	40	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32

Из-за сдвига для каждого подключа используется отличающееся подмножество битов ключа. Каждый бит используется приблизительно в 14 из 16 подключей, хотя не все биты используются в точности одинаковое число раз.

5.2.4. ПЕРЕСТАНОВКА С РАСШИРЕНИЕМ

Эта операция расширяет правую половину данных R_i от 32 до 48 битов. Поскольку при этом не просто повторяются определенные биты, но и изменяется их порядок, эта операция называется перестановкой с расширением. У нее две задачи: привести размер правой половины в соответствие с ключом для операции XOR и получить более длинный результат, который можно будет сжать в ходе операции подстановки.

Однако главный криптографический смысл совсем в другом: вследствие влияния одного бита на две подстановки быстрее возрастает зависимость битов результата от битов исходных данных. Это называется лавинным эффектом. DES спроектирован так, чтобы как можно быстрее добиться наибольшей зависимости каждого бита шифротекста от каждого бита открытого текста и каждого бита ключа.

Перестановка с расширением показана на рис. 5.3. Иногда она называется *E*-блоком (от *expansion*). Для каждого 4-битового входного блока первый и четвертый биты преобразуются в два бита выходного блока, а второй и третий биты — в один бит выходного блока. В табл. 5.5 показано, какие позиции результата соответствуют каким позициям исходных данных. Например, бит входного блока в позиции **3** переместится в позицию **4** выходного блока, а бит входного блока в позиции **21** — в позиции **30** и **32** выходного блока.

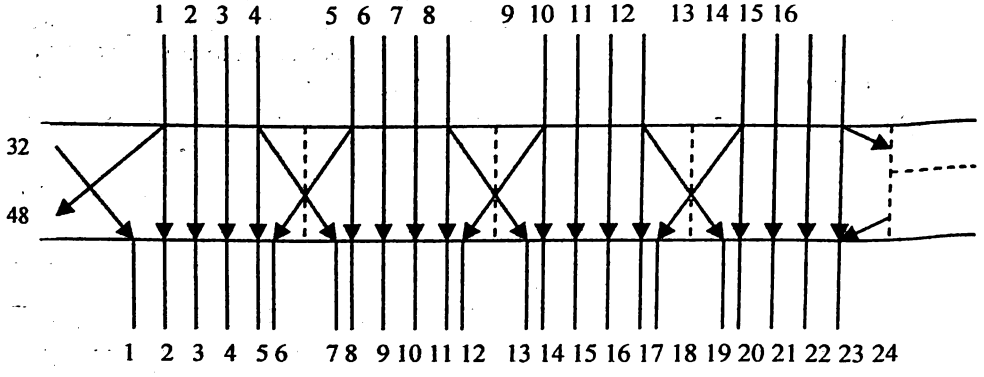


Рис. 5.3. Схема перестановки с расширением

Хотя выходной блок больше входного, каждый входной блок генерирует уникальный выходной блок.

Таблица 5.5

Перестановка с расширением

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

5.2.5. ПОДСТАНОВКА С ПОМОЩЬЮ S-БЛОКОВ

После объединения сжатого блока ключа с расширенным блоком данных с помощью XOR над 48-битовым результатом выполняется операция подстановки. Подстановки производятся в восьми блоках подстановки, или *S*-блоках (от *substitution*). У каждого *S*-блока 6-битовый вход и 4-битовый выход, всего используется восемь различных *S*-блоков. (Для восьми *S*-блоков DES потребуется 256 байтов памяти.) 48 битов делятся на восемь 6-битовых подблоков.

Каждый отдельный подблок обрабатывается отдельным *S*-блоком: первый подблок — *S*-блоком 1, второй — *S*-блоком 2 и так далее (рис. 5.4).

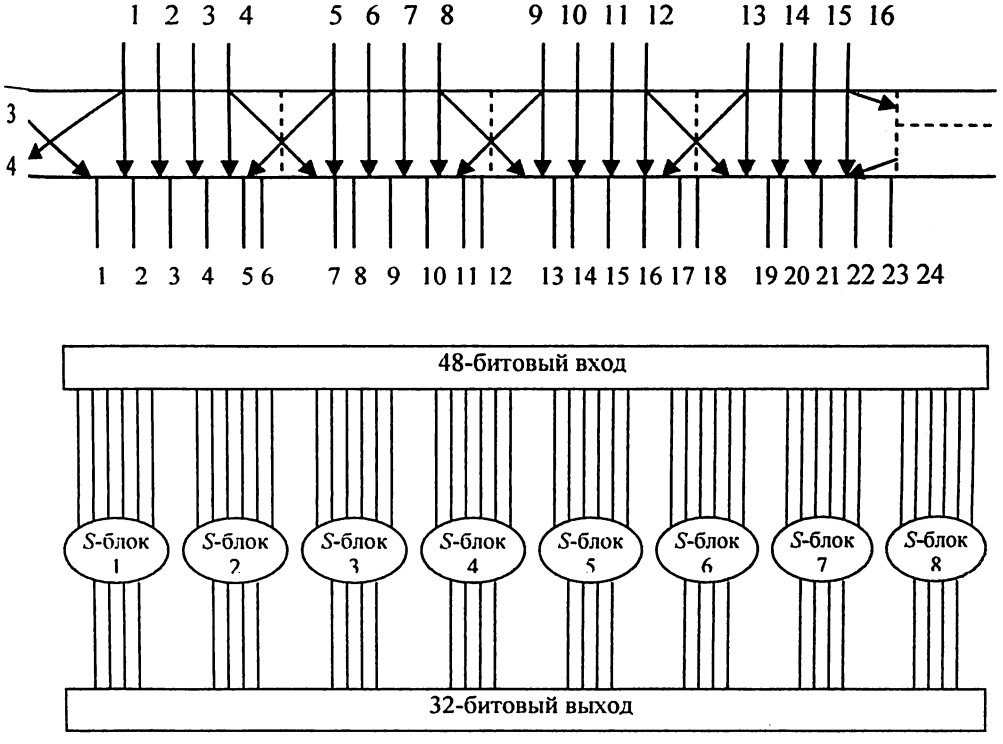


Рис. 5.4. Подстановка с помощью *S*-блоков

Каждый *S*-блок представляет собой таблицу из 4 строк и 16 столбцов. Все восемь *S*-блоков показаны в табл. 5.6.

Каждый элемент в блоке является 4-битовым числом. По шести входным битам *S*-блока определяется, под какими номерами столбцов и строк искать выходное значение: входные биты особым образом определяют элемент *S*-блока. Рассмотрим 6-битовый вход *S*-блока: *b*1, *b*2, *b*3, *b*4, *b*5, *b*6. Биты *b*1и *b*6 объединяются, образуя 2-битовое число от 0 до 3, соответствующее строке таблицы. Средние четыре бита — с *b*2 по *b*5 —объединяются, образуя 4-битовое число от 0до15, соответствующее столбцу таблицы.

Таблица 5.6

S-блоки

S-блок 1:																
14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S-блок 2:																
15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S-блок 3:																
10	0,	9	14	6	3	25	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S-блок 4:																
7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S-блок 5:																
2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S-блок 6:																
12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

Продолжение табл. 5.6

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

Например, пусть на вход шестого *S*-блока (т.е. биты с 31 по 36) попадает 110011. Первый и последний биты, объединяясь, образуют 11, что соответствует строке 3 шестого *S*-блока. Средние 4 бита образуют 1001, что соответствует столбцу 9 того же *S*-блока. Элемент шестого *S*-блока, находящийся на пересечении строки 3 и столбца 9, — это 14. Вместо 110011 подставляется 1110.

Легче всего реализовать *S*-блоки программно в виде массивов с 64 элементами. Для этого потребуется переупорядочить элементы, что не является трудной задачей.

Каждый *S*-блок можно рассматривать как функцию подстановки 4-битового элемента: *b*2 по *b*5 являются входом, а некоторое 4-битовое число — результатом. Биты *b*1 и *b*6 определяют одну из четырех функций подстановки, возможных в данном *S*-блоке.

Подстановка с помощью *S*-блоков является ключевым этапом DES. Другие действия алгоритма *линейны* и легко поддаются анализу. *S*-блоки нелинейны, и именно они в большей степени, чем все остальное, обеспечивают безопасность DES.

В результате этого этапа подстановки получаются восемь 4-битовых блоков, которые вновь объединяются в единый 32-битовый блок. Этот блок поступает на вход следующего этапа — перестановки с помощью *P*-блоков.

5.2.6. ПЕРЕСТАНОВКА СПОМОЩЬЮ *P*-БЛОКОВ

32-битовый выход подстановки с помощью *S*-блоков перетасовывается в соответствии с *P*-блоком. Эта перестановка перемещает каждый входной бит в другую позицию, ни один бит не используется дважды и ни один бит не игнорируется. Такой процесс называется

прямой перестановкой, или просто перестановкой. Позиции, в которые перемещаются биты, показаны в табл. 5.7. Например, бит 21 перемещается в позицию 4, а бит 4 — в позицию 31.

Таблица 5.7

Перестановка с помощью P-блоков

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

Наконец, результат перестановки с помощью *P*-блока объединяется посредством XOR с левой половиной первоначального 64-битового блока. Затем левая и правая половины меняются местами, и начинается следующий этап.

5.2.7. ЗАКЛЮЧИТЕЛЬНАЯ ПЕРЕСТАНОВКА

Заключительная перестановка является обратной по отношению к начальной. Обратите внимание, что левая и правая половины не меняются местами после последнего этапа DES, вместо этого объединенный блок $R_{16}L_{16}$ используется как вход заключительной перестановки. В этом нет ничего особенного: перестановка половинок с последующим циклическим сдвигом привела бы к точно такому же результату. Это сделано для того, чтобы алгоритм можно было использовать как для шифрования, так и для дешифрования.

Таблица 5.8

Заключительная перестановка

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

5.3. ДЕШИФРОВАНИЕ DES

После всех подстановок, перестановок, операций XOR и циклических сдвигов можно подумать, что алгоритм дешифрования, резко отличаясь от алгоритма шифрования, точно также запутан. Напротив, различные компоненты DES были подобраны так, чтобы выполнялось очень полезное свойство: для шифрования и дешифрования используется один и тот же алгоритм.

DES позволяет использовать для шифрования и дешифрования блока одну и ту же функцию. Единственное отличие состоит в том, что ключи должны использоваться в обратном порядке, т.е. если на этапах шифрования использовались ключи K1, K2, K3 ,..., K16, то ключами дешифрирования будут K16, K15, K14, ..., K1. Алгоритм, который создает ключ для каждого этапа, также цикличен. Ключ сдвигается вправо, а число позиций сдвига равно 0, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1.

5.4. РЕЖИМЫ DES

Как правило, определяются четыре режима работы: ECB, CBC, OFB и CFB. Банковские стандарты ANSI определяют для шифрования ECB и CBC, а для проверки подлинности — CBC и n-битовый CFB.

В мире программного обеспечения сертификация обычно не важна. Из-за своей простоты в большинстве существующих коммерческих программ используется ECB, хотя этот режим наиболее чувствителен к вскрытию. CBC используется редко, несмотря на то что он лишь незначительно сложнее, чем ECB, и обеспечивает бóльшую безопасность.

5.5. АППАРАТНЫЕ И ПРОГРАММНЫЕ РЕАЛИЗАЦИИ DES

В настоящее время утверждается, что самой быстрой является микросхема DES, разработанная в Digital Equipment Corporation. Она поддерживает режимы ECB и CBC и основана на вентильной матрице GaAs, состоящей из 50 000 транзисторов. Данные могут эффективно зашифровываться и дешифрироваться со скоростью 1 гигабит в секунду, обрабатывая 16,8 млн блоков в 1 с. Параметры ряда коммерческих микросхем DES приведены в табл. 5.9. Кажущиеся противоречия между тактовой частотой и скоростью обработки данных обусловлены конвейеризацией внутри микросхем, в которой может быть реализовано несколько работающих параллельно DES-механизмов.

Наиболее выдающейся микросхемой DES является 6868 VLSI (ранее называвшаяся «Gatekeeper»). Она не только может выполнять шифрование DES за 8 тактов (лабораторные прототипы могут делать это за 4 такта), но также выполнять троекратный DES в режиме ECB за 25 тактов, а троекратный DES в режимах OFB или CBC — за 35 тактов.

Программная реализация DES на мэйнфрейме IBM 3090 может выполнить 32 000 шифрований DES в секунду. На других платфор-

мах скорость ниже, но все равно достаточно велика. В табл. 5.9 приведены действительные результаты и оценки для различных микропроцессоров Intel и Motorola.

Таблица 5.9

Коммерческие микросхемы DES

Производитель	*Микросхема*	*Год*	*Тактовая частота, МГц*	*Скорость данных, Мбайт/с*	*Доступность*
AMD	Am9518	1981	3	1,3	Н
AMD	Am9518	?	4	1,5	Н
AMD	AmZ8068	1982	4	1,7	Н
AT&T	T7000A	1985	?	1,9	Н
CE-Infosys	SuperCrypt CE99C003	1992	20	12,5	Д
CE-Infosys	SuperCrypt CE99C003A	1994	30	20,0	Д
Cryptech	Cry12C102	1989	20	2,8	Д
Newbrige	CA20C03A	1991	25	3,85	Д
Newbrige	CA20C03W	1992	8	0,64	Д
Newbrige	CA95C68/18/09	1993	33	14,67	Д
Pijnenburg	PCC100	?	?	2,5	Д
Semaphore Communications	Roadrunner284	?	40	35,5	Д

ГЛАВА 6. УПРАВЛЕНИЕ КЛЮЧАМИ В СИММЕТРИЧНЫХ КРИПТОСИСТЕМАХ

6.1. КРАТКАЯ ХАРАКТЕРИСТИКА КАНАЛЬНОГО И СКВОЗНОГО ШИФРОВАНИЯ

При канальном шифровании каждый уязвимый канал оборудуется на обоих концах устройствами шифрования. Таким образом, весь поток данных в канальном шифровании оказывается защищенным. Несмотря на то что для этого в большой сети потребуется немало устройств шифрования, преимущества такого подхода очевидны.

Одним из недостатков является то, что сообщение должно дешифровываться каждый раз, когда оно проходит через пакетный переключатель (свитч), поскольку переключатель должен прочитать адрес (номер виртуального канала) в заголовке пакета, чтобы направить пакет по нужному адресу. Поэтому сообщение оказывается уязвимым в каждом переключателе. При использовании общедоступных сетей с коммуникацией пакетов данных пользователь не имеет никакой возможности контролировать безопасность узлов такой сети.

По поводу использования канального шифрования следует сделать несколько замечаний. Чтобы эта стратегия оказалась эффективной, шифрование должно применяться во всех доступных для пользователя каналах на маршруте от источника к адресату. Каждая пара узлов, находящаяся на концах одного канала, должна применять свой уникальный ключ, и ключи, используемые для различных каналов, должны быть различными. Таким образом, потребуется множество ключей. Но каждый из этих ключей должен быть предоставлен только одной соответствующей паре узлов.

При сквозном шифровании процесс шифрования выполняется только в двух конечных системах. Исходные данные шифруются в ведущем узле или в терминале источника. Затем данные в шифрованном виде передаются без изменений через всю сеть к терминалу или к ведущему узлу адресата. Адресат использует тот же ключ, что и отправитель, и поэтому может дешифровать полученные данные. Эта схема кажется безопасной в отношении защиты от воздействий в канале связи или в переключателях пакетов. Поэтому сквозное шифрование освобождает конечного пользователя от забот о степени защищенности сетей и каналов, по которым осуществляется связь.

Однако у такого подхода имеется слабое место. Рассмотрим следующую ситуацию. Ведущий узел присоединяется к сети с комму-

никацией пакетов по соответствующему протоколу, открывает виртуальный канал к другому ведущему узлу и готовится передать данные этому ведущему узлу с использованием сквозного шифрования. Данные передаются по такой сети в виде пакетов, состоящих из заголовка и некоторой порции данных пользователя. Какую часть каждого пакета должен шифровать ведущий узел? Предположим, что ведущий узел шифрует весь пакет, включая заголовок. Но этого делать нельзя, так как выполнить дешифрование может только другой ведущий узел. Узел переключения пакетов получит шифрованные данные, не сможет прочитать заголовок и поэтому не сможет переслать пакет дальше. Отсюда следует, что ведущий узел должен шифровать только ту часть пакета, которая содержит данные пользователя, и оставить заголовок нетронутым.

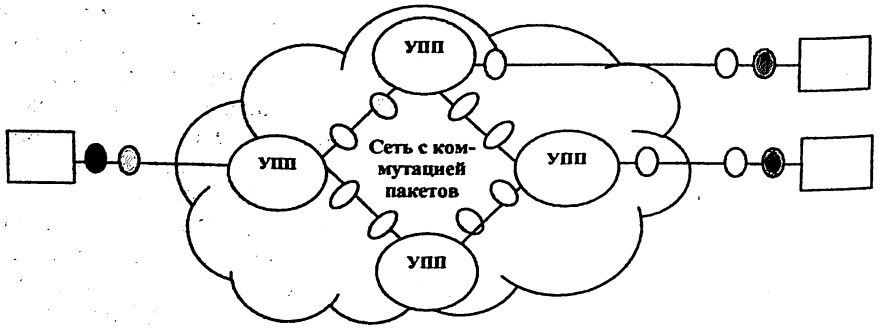


Рис. 6.1. Шифрование в сети с коммутацией пакетов

Итак, в случае сквозного шифрования данные пользователя оказываются защищенными, чего нельзя сказать о самом потоке данных, поскольку заголовки пакетов передаются в открытом виде.

В то же время сквозное шифрование в некоторой степени решает задачу аутентификации. Если конечные системы используют один общий ключ шифрования, то получатель имеет возможность убедиться, что полученное сообщение пришло от соответствующего отправителя, так как только этот отправитель использует соответствующий ключ.

При канальном шифровании такая аутентификация уже не будет внутренним свойством самой схемы шифрования.

Чтобы достичь лучшей защиты, требуется как канальное, так и сквозное шифрование (рис. 6.1). При использовании обеих этих форм шифрования ведущий узел шифрует порцию пакета данных пользователя, применяя ключ сквозного шифрования. Затем весь пакет шифруется с помощью ключа канального шифрования.

При движении пакета по сети каждый переключатель дешифрует пакет с применением ключа шифрования соответствующего канала, чтобы прочитать заголовок, а потом снова шифрует весь пакет для передачи его по следующему каналу. Теперь весь пакет оказывается защищенным почти все время, за исключением времени, когда пакет находится в памяти пакетного переключателя и заголовок пакета является открытым.

В табл. 6.1 приведены ключевые характеристики двух стратегий шифрования, о которых идет речь.

Таблица 6.1

Характеристики канального и сквозного шифрования

Канальное шифрование	*Сквозное шифрование*
Защита в конечных системах и промежуточных системах	
Сообщение уязвимо в ведущем узле источника	Сообщение зашифровано в ведущем узле источника
Сообщение уязвимо в промежуточных узлах	Сообщение зашифровано в промежуточных узлах
Роль пользователя	
Процесс отправки предполагает участие пользователя	Процесс отправки предполагает участие пользователя
Шифрование ***не зависит*** от пользователя	Шифрование выполняется пользователем
Средствами шифрования управляет ведущий узел	Алгоритм шифрования должен определить пользователь
Для всех пользователей используются одни и те же средства шифрования	Схему шифрования выбирает пользователь
Возможна ***аппаратная*** реализация функции шифрования	Предполагается ***программная*** реализация функции шифрования
Шифруются либо ***все*** сообщения, либо ***не*** шифруется ***ни одно***	Для каждого сообщения решение об использовании или об отказе от средств шифрования принимается пользователем

Окончание табл. 6.1

Канальное шифрование	*Сквозное шифрование*
Вопросы реализации	
Требуется по одному ключу на каждое из звеньев связи между узлами	Требуется по одному ключу для каждой пары пользователей
Обеспечивается идентификация узла отправителя	Обеспечивается идентификация пользователя

6.2. ПРИНЦИПЫ РАСПРЕДЕЛЕНИЯ КЛЮЧЕЙ

При традиционном шифровании обе участвующие в обмене данными стороны должны получить один и тот же ключ, к которому другие пользователи лишены доступа. При этом обычно требуется частое изменение ключей, чтобы уменьшить объем теряемых данных в случае, когда какой-нибудь из ключей становится известным противнику.

Поэтому надежность любой криптографической системы во многом зависит от используемой при этом *системы распределения ключей,* представляющей собой средства доставки ключей двум сторонам, планирующим обмен данными, не позволяющие другим увидеть эти ключи.

Для двух сторон, **A** и **B**, как указано далее, распределение ключей можно организовать различными способами:

1. Ключ может быть выбран стороной **A** и физически доставлен стороне **B**.
2. Ключ может выбрать третья сторона и физически доставить его участникам **A** и **B**.
3. Если участники обмена **A** и **B** уже используют некоторый общий ключ, одна из сторон может передать новый ключ второй стороне в шифрованном виде, используя старый ключ.
4. Если обе из сторон **A** и **B** имеют криптографически защищенные каналы связи с третьей стороной **C**, то последняя может доставить ключ участникам **A** и **B** по этим защищенным каналам.

Варианты 1 и 2 предполагают передачу ключа из рук в руки. При канальном шифровании это требование может оказаться вполне разумным, поскольку любое устройство канального шифрования предполагает обмен данными только с соответствующим устройством на другом конце канала.

Но в случае сквозного шифрования физическая доставка ключа практически неприемлема. В любой распределенной системе каждый ведущий узел или терминал может участвовать в обмене данными со многими другими ведущими узлами и терминалами. Поэтому каждому такому устройству потребуется множество ключей,

которые придется поставлять динамично. Проблема оказывается весьма трудной для решения, особенно в случае больших глобально распределенных систем.

Масштаб проблемы зависит от числа контактирующих пар, которые приходится обслуживать. Если сквозное шифрование осуществляется на сетевом уровне или на уровне IP, то потребуется по одному ключу для каждой пары ведущих узлов в сети, обменивающихся данными. Поэтому, если имеется N ведущих узлов, число необходимых ключей будет равно $[N(N-1)]/2$. Если шифрование осуществляется на уровне приложения, то свой ключ потребуется для каждой пары пользователей или процессов, выходящих на связь. При этом сеть может иметь сотни ведущих узлов и тысячи пользователей и процессов. На рис. 6.2 для случая сквозного шифрования показана зависимость сложности задачи распределения ключей от числа пар, участвующих в обмене данными. Например, в сети, насчитывающей 1000 узлов, где шифрование осуществляется на уровне узла, скорее всего придется распределять около полумиллиона ключей. А если в такой сети поддерживается около 10 000 приложений, то при шифровании на уровне приложений может потребоваться распределение около 50 миллионов ключей.

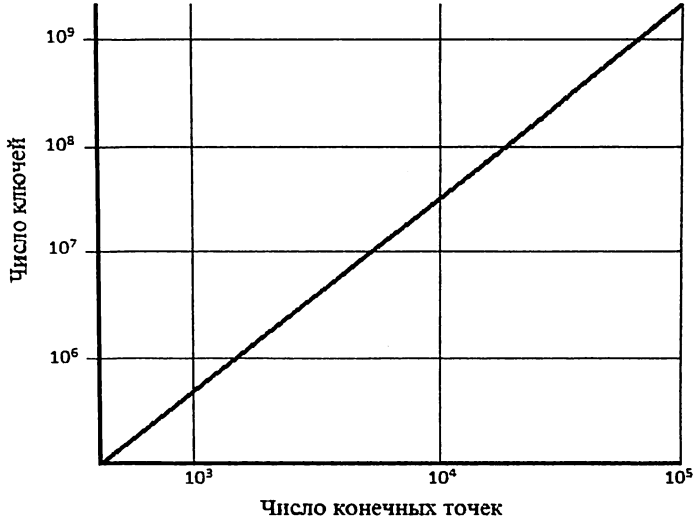


Рис. 6.2. Число ключей, необходимое для поддержки любых соединений между заданным числом конечных точек

Возвращаясь к списку способов распределения ключей, отметим, что способ 3 возможен как для случая канального шифрования, так и для сквозного, но если противнику когда-либо удастся получить доступ к одному из ключей, то он сможет получить и все последующие. К тому же начальное распределение потенциально миллионов ключей все равно должно быть выполнено.

Для сквозного шифрования широко применяется схема, являющаяся некоторой вариацией способа 4. В этой схеме за доставку ключей парам пользователей (ведущим узлам, процессам, приложениям) отвечает некоторый центр распределения ключей. Каждый пользователь при этом должен получить свой уникальный ключ, используемый им совместно с центром распределения ключей в целях организации доставки ключей.

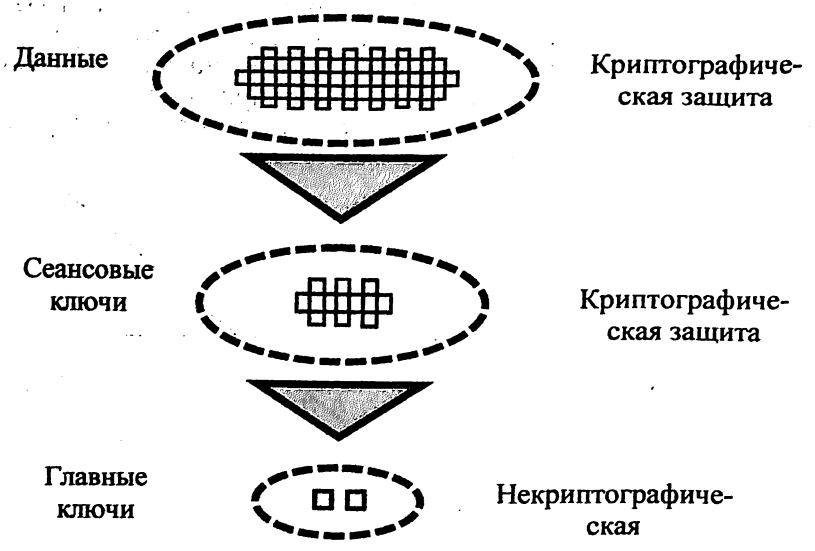


Рис. 6.3. Использование иерархии ключей

Использование центра распределения ключей предполагает организацию некоторой их иерархии. В минимальной конфигурации такая иерархия включает два уровня (рис. 6.3). Связь между конечными системами шифруется с использованием временного ключа, часто называемого сеансовым ключом (session key). Как правило, сеансовый ключ служит только для конкретного логического соединения, например, виртуального канала, или для транспортировки данных, после чего этот ключ больше не применяется. Сеансовый ключ

получают от центра распределения ключей по тем же средствам доставки данных в сети, которые служат для организации связи между конечными пользователями. Соответственно, сеансовые ключи передаются в шифрованном виде, а для шифрования используется главный ключ (master key),общий для центра распределения ключей и данной конечной системы или конкретного пользователя.

Для каждой конечной системы или конечного пользователя создается уникальный главный ключ, который применяется совместно с центром распределения ключей. Конечно, эти главные ключи тоже должны быть каким-то образом распределены. Однако эта проблема по своей сложности значительно проще. Как уже упоминалось, для N объектов, попарно обменивающихся данными, требуется [N (N — 1)] / 2 сеансовых ключей. А главных ключей требуется всего N, по одному на каждый объект. Поэтому главные ключи могут быть распределены некоторым некриптографическим образом, например, физической доставкой адресату.

Сценарий распределения ключей. Распределение ключей можно реализовать разными способами. Типичный сценарий показан на рис. 6.4. Этот сценарий предполагает, что каждый пользователь имеет уникальный главный ключ, используемый совместно с центром распределения ключей (ЦРК).

Предположим, что пользователь **A** намерен создать логическое соединение с пользователем **B** и для защиты данных, которые предполагается передать вовремя этого соединения, требуется одноразовый сеансовый ключ.

При этом пользователь **A** имеет секретный ключ K_a, известный только ему и ЦРК, и точно так же **B** использует общий с ЦРК главный ключ K_b.

Система обмена информацией выглядит следующим образом.

1. Инициатор **A** посылает запрос в ЦРК на получение сеансового ключа для защиты логического соединения с **B**. Посылаемое при этом сообщение должно включать информацию, позволяющую однозначно определить **A** и **B**, а также некоторый идентификатор N1, уникальный для данного запроса, обычно называемый оказией (nonce — данный случай, данное время (англ.)). Такими идентификаторами могут быть текущее время, некоторый счетчик или случайное число — как минимум, этот идентификатор должен быть уникальным для каждого запроса. Кроме того, чтобы предотвратить возможность фальсификации сообщения противником, последнему должно быть непросто угадать этот идентификатор. Поэтому хорошим выбором для оказии можно считать случайное число.

2. ЦРК отвечает на запрос сообщением, шифрованным с использованием ключа Ка. Единственным пользователем, кто может полу-

чить и прочитать это сообщение, является **A**, и поэтому **A** может быть уверенным, что это сообщение пришло от ЦРК. Сообщение включает два элемента, предназначенных для **A**:

– одноразовый сеансовый ключ Ks, который будет использоваться в сеансе связи;

– оригинальное сообщение запроса, включающее оказию, чтобы у пользователя **A** была возможность сопоставить ответ с соответствующим запросом.

3. Таким образом, **A** может удостовериться, что его первоначальный запрос не был изменен на пути к ЦРК, а оказия не позволит перепутать ответ на данный запрос с ответом на какой-либо из предыдущих запросов.

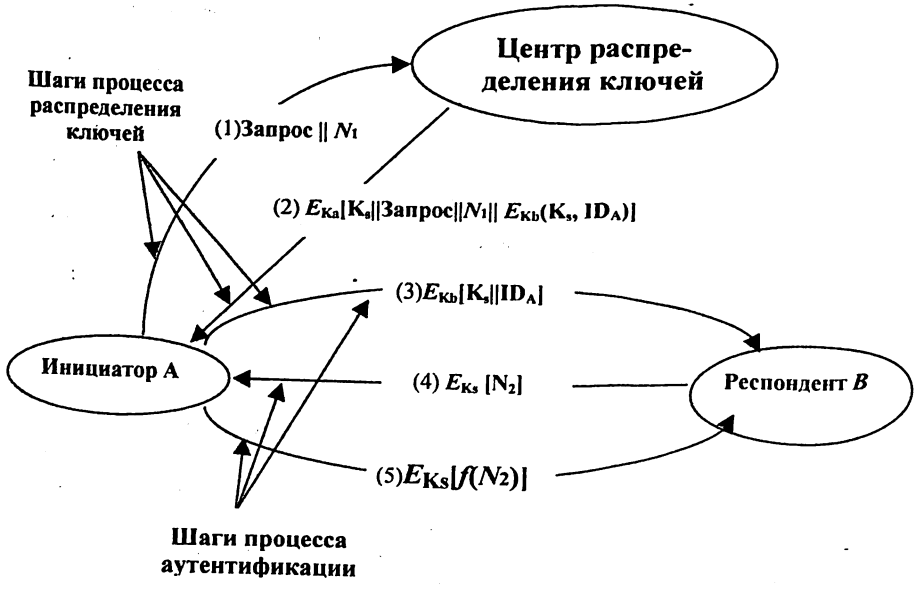


Рис. 6.4. Сценарий распределения ключей

1. Кроме того, сообщение включает и два элемента, предназначенные для **B**:

– одноразовый сеансовый ключ K_s, который будет использоваться в сеансе связи;

– идентификатор ID_A пользователя **A** (например, его сетевой адрес).

2. Оба элемента шифруются с помощью ключа K_b (главного ключа, применяемого совместно ЦРК и **B**), и предполагается, что они должны быть впоследствии отправлены **B**, чтобы установить соединение и идентифицировать **A**.

3. Сторона **A** сохраняет сеансовый ключ для предстоящего сеанса связи и пересылает стороне **B** информацию, полученную от ЦРК и предназначенную для **B** (а именно, информацию $E_{Kb}[K_s||ID_A]$). Поскольку эта информация шифрована с использованием K_b, она оказывается защищенной. Теперь получатель **B** знает сеансовый ключ (K_s) и знает, что полученная информация пришла от ЦРК (поскольку эта информация оказывается зашифрованной с использованием ключа K_b).

К этому моменту сеансовый ключ оказывается доставленным и стороне **A**, и стороне **B**, и поэтому они могут начать защищенный обмен данными. Но перед этим желательно выполнить еще две операции.

1. Используя только что полученный сеансовый ключ K_s для шифрования, сторона **B** посылает стороне **A** новую оказию N_2.

2. С помощью того же ключа K_s сторона **A** в ответ возвращает $f(N_2)$, где f является функцией, выполняющей некоторое преобразование N_2 (например, добавление единицы).

Эти действия призваны убедить адресата **B** в том, что первоначально полученное им сообщение (п. 3) не было воспроизведено.

Следует обратить внимание на то, что сам процесс передачи ключа фактически выполняется в п. 1–3, а п. 4 и 5, так же как отчасти и п. 3, призваны обеспечить функцию аутентификации.

6.3. ПРИНЦИПЫ УПРАВЛЕНИЯ КЛЮЧАМИ

6.3.1. УПРАВЛЕНИЕ ИЕРАРХИЕЙ КЛЮЧЕЙ

Совсем необязательно возлагать функцию распределения ключей на один ЦРК. На самом деле для больших сетей это вообще непрактично. Более выгодно определить некоторую иерархию центров распределения ключей. Например, можно создать локальные ЦРК, ответственные за малые домены всей сети, скажем, отдельные локальные сети или сети, размещенные в одном здании. Тогда при связи между объектами внутри одного локального домена за распределение ключей будет отвечать локальный ЦРК. Если же общий ключ потребуется для связи двух объектов из разных доменов, то соответствующие локальные ЦРК могут использовать для переговоров ЦРК глобального уровня. В этом случае любой из трех вовлеченных в переговоры ЦРК может генерировать подходящий ключ. Такая иерархия может состоять из трех или даже большего числа уровней — в зависимости от числа пользователей и от географической протяженности сети.

Иерархическая схема минимизирует усилия, необходимые для решения задачи распределения главных ключей, поскольку при этом большинство главных ключей запрашивается для совместного использования объектами, находящимися в ведении конкретного ло-

кального ЦРК. К тому же такая схема ограничивает ущерб от ошибок или повреждения одного конкретного ЦРК рамками соответствующей локальной области сети.

6.3.2. ДЕЦЕНТРАЛИЗОВАННОЕ УПРАВЛЕНИЕ КЛЮЧАМИ

Использование центра распределения ключей предполагает, что ЦРК должен внушать доверие и быть надежно защищенным от посягательств. От этих требований можно отказаться, если полностью децентрализовать распределение ключей. И хотя полная децентрализация в больших сетях, где шифрование осуществляется по традиционной схеме, практически не применяется, децентрализация может быть полезной в контексте локальных сетей.

При децентрализации требуется, чтобы каждая конечная система имела возможность обмениваться данными некоторым защищенным образом со всеми другими потенциально достижимыми конечными системами с целью распределения сеансовых ключей. При этом в сети с n конечными системами может понадобиться до $[n\,(n-1)]\,/\,2$ главных ключей.

Сеансовый ключ может быть определен в результате следующей последовательности действий (рис. 6.5).

Инициатор **A** посылает запрос участнику обмена данными **B** на получение сеансового ключа, содержащий также оказию N_1.

Респондент **B** отвечает сообщением, зашифрованным с использованием общего главного ключа K_m. Этот ответ содержит сеансовый ключ K_s, выбранный стороной **B**, идентификатор **B** (ID_B), значение $f(N_1)$ и другую оказию N_2.

Используя полученный сеансовый ключ K_s, сторона **A** возвращает стороне **B** значение $f(N_2)$.

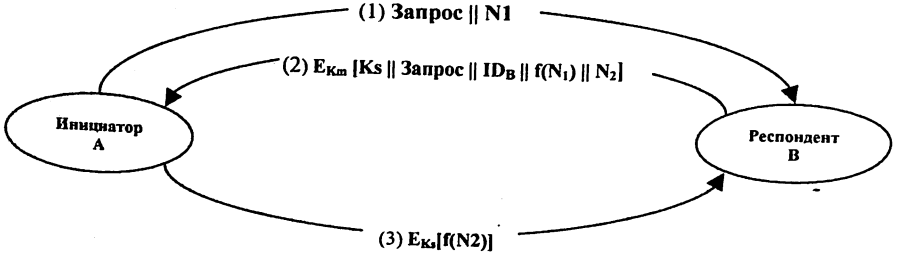


Рис. 6.5. Децентрализованное распределение ключей

Итак, каждому узлу придется поддерживать не более, чем $(n-1)$ главных ключей, но при этом можно генерировать столько сеансовых ключей, сколько потребуется. Поскольку сообщения, шифрованные с использованием главных ключей, оказываются короткими, их криптоанализ представляется трудным делом. Как и в предыдущем случае, сеансовые ключи следует использовать на протяжении ограниченного времени, чтобы обеспечить лучшую их защиту.

6.3.3. УПРАВЛЕНИЕ ИСПОЛЬЗОВАНИЕМ КЛЮЧЕЙ

Принцип иерархии ключей и использование автоматизированных методов распределения ключей значительно уменьшают число ключей, которые приходится распределять и доставлять вручную. Но может оказаться желательным иметь определенный контроль над порядком применения автоматически распределенных ключей. Например, в дополнение к разделению ключей на главные и сеансовые, можно определить различные типы сеансовых ключей в зависимости от сферы их действия:

- ключи для шифрования данных, используемые при пересылке через сеть данных общего назначения;
- ключи для шифрования личных идентификационных номеров (PIN-кодов), используемые при передаче данных в системах электронных платежей и электронной коммерции;
- ключи для шифрования файлов, используемые для шифрования файлов, хранящихся в общедоступных местах.

Чтобы понять смысл разделения ключей на типы, следует оценить риск того, что главный ключ может импортироваться в качестве ключа шифрования данных в устройство шифрования. Обычно главный ключ физически защищается размещением его в криптографическом оборудовании центра распределения ключей и конечных систем. Сеансовые ключи, шифрованные с использованием этого главного ключа, оказываются доступными для прикладных программ так же, как и данные, шифрованные с использованием сеансовых ключей. Однако если главный ключ в этом отношении не отличается от сеансового ключа, то для некоторого неавторизованного приложения может оказаться возможным получить открытые тексты сеансовых ключей, шифрованных с помощью главного ключа.

Ввиду этого желательно, чтобы в системе имелись средства управления, ограничивающие использование ключей шифрования в зависимости от характеристик, связываемых с такими ключами.

Одним из простейших подходов является связывание с каждым ключом некоторого признака. Предложенный подход применим в случае DES, где для признака используются незадействованные 8 бит 64-битового ключа DES. Иными словами, 8 неиспользуемых для ключа бит, обычно резервируемых для контроля четности, задают значение признака ключа. Эти биты имеют следующую интерпретацию:

- один бит указывает, каким является ключ — сеансовым или главным;
- один бит указывает, может ли ключ служить для шифрования;
- один бит указывает, может ли ключ применяться для дешифрования;
- остальные биты зарезервированы для использования в будущем.

Из-за того что признак оказывается частью ключа, он при распределении шифруется вместе с ключом, чем обеспечивается защита. Недостатком этой схемы является то, что:

- длина признака не превышает 8бит, что ограничивает гибкость и функциональные возможности;
- ввиду того что признак не передается в открытом виде, он может использоваться только с момента дешифрования, что ограничивает возможности контроля.

Более гибкая схема, обычно называемая схемой на основе управляющего вектора, организована следующим образом. В этой схеме каждый сеансовый ключ ассоциируется с управляющим вектором **CV**, состоящим из ряда полей, описывающих возможности применения и ограничения для данного сеансового ключа. Длина управляющего вектора может меняться.

Управляющий вектор **CV** криптографическими средствами связывается с ключом во время создания ключа в ЦРК. Процессы связывания и разделения показаны на рис. 6.6.

Сначала управляющий вектор **CV** проходит через функцию хэширования, в результате чего получается значение, длина которого равна длине ключа шифрования.

Функции хэширования подробно описаны в гл. 12. В сущности, функция хэширования отображает значения из некоторой большей области в некоторую меньшую область, обеспечивая при этом однородный в некотором смысле разброс значений. Например, если числа из диапазона от 1 до 100 ужимаются в диапазон от 1 до 10, то в каждое результирующее значение должно отображаться приблизительно 10% исходных значений.

Затем полученное в результате хэширования значение с помощью операции XOR объединяется с главным ключом, и это новое значение служит в качестве ключа для шифрования сеансового ключа.

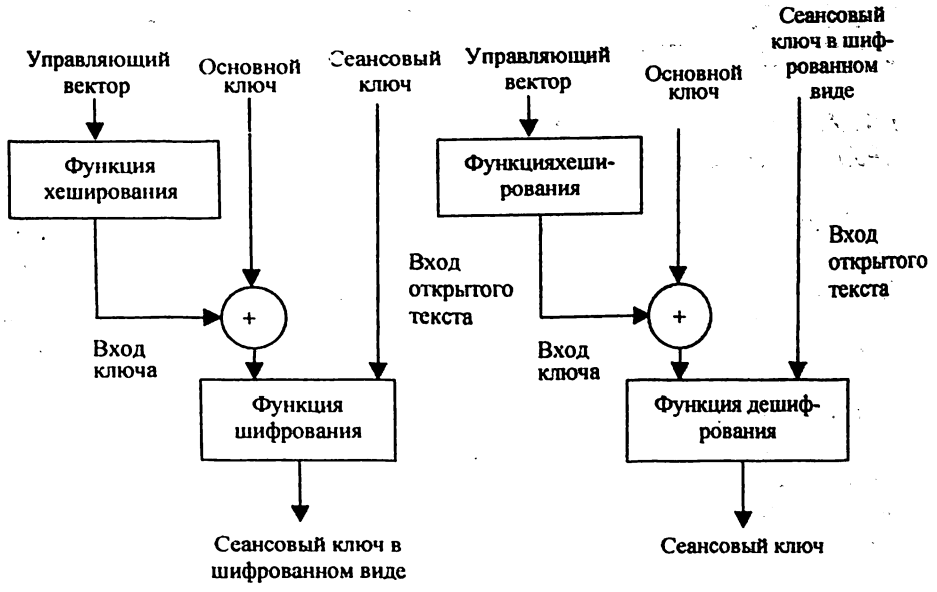
Таким образом,

$$\text{Значение хэширования} — H = \mathrm{h}(\mathbf{CV}),$$

$$\text{Значение ключа} — \mathrm{K_Г} \oplus H,$$

$$\text{Шифрованный текст} — \mathrm{EK_Г} \oplus H[\mathrm{K_s}],$$

где $\mathrm{K_Г}$ обозначает главный ключ, а K_s — сеансовый ключ.



а — Шифрование управляющего вектора *б* — Дешифрование управляющего вектора

Рис. 6.6. Шифрование и дешифрование управляющего вектора

Сеансовый ключ восстанавливается в открытый вид с помощью обратной операции:

$$\mathrm{K}_s = \mathrm{D}_{\mathrm{K_Г}} \oplus H[E_{\mathrm{K_Г}} \oplus H[\mathrm{K}_s]].$$

Наряду с сеансовым ключом пользователю от ЦРК доставляется управляющий вектор в открытой форме. Сеансовый ключ можно восстановить только при наличии и главного ключа, общего для данного пользователя и ЦРК, и управляющего вектора. Таким образом поддерживается связь между сеансовым ключом и соответствующим ему управляющим вектором.

Применение управляющего вектора имеет два преимущества по сравнению с использованием 8-битового признака.

Во-первых, нет ограничения на длину управляющего вектора, что при использовании ключей дает возможность осуществлять контроль любой сложности.

Во-вторых, управляющий вектор оказывается доступным в открытом виде на всех стадиях, вследствие чего функции контроля за использованием ключей можно разместить в самых разных точках.

ГЛАВА 7. СТОЙКОСТЬ КРИПТОГРАФИЧЕСКИХ СИСТЕМ И АЛГОРИТМОВ

7.1. ИНФОРМАЦИОННО-ТЕОРЕТИЧЕСКИЙ АНАЛИЗ КРИПТОГРАФИЧЕСКОЙ СТОЙКОСТИ

Основное назначение криптосистем — обеспечение передачи секретной информации по несекретным каналам. Этим обусловлено признание стойкости (способности противостоять попыткам противника осуществить дешифрование перехваченного шифротекста или выяснить ключи шифра) наиболее важным свойством любой криптосистемы [54].

Клод Шеннон для обеспечения стойкости криптосистем разработал информационно-теоретический подход, опирающийся на понятие информации, ее количественную оценку и анализ количества информации, известной об открытом тексте. Этот подход считается исторически первым подходом к определению стойкости криптосистем.

Обозначим через $x_1,...,x_n$ n возможных сообщений. Вероятности появления данных сообщений равны

$$P(x_1),...,P(x_n) \quad \text{причем} \quad \sum_{i=1}^{n} P(x_i) = 1.$$

Тогда энтропию сообщения x, с помощью которой формально измеряется количество информации, можно определить как

$$H(x) = -\sum_{i=1}^{n} P(x_i)\log_2 P(x_i) = \sum_{i=1}^{n} P(x_i)\log_2 \frac{1}{P(x_i)}.$$

Значение величины $H(x)$ для заданного n становится максимальным при одинаковой вероятности появления сообщений, т.е. когда

$$H(x) = \log_2 n \text{ при } \quad P(x_1) = ... = P(x_n) = \frac{1}{n}.$$

Уменьшение неопределенности (энтропии) происходит при все более отличном от равновероятного распределении вероятностей сообщений. Для некоторого сообщения x_i минимальная энтропия $\min H(X) = 0$ может быть достигнута при вероятности $P(x_i) = 1$. Энтропия позволяет измерить неопределенность сообщения числом

бит подлежащей восстановлению информации после его преобразования в шифротекст и сокрытия от криптоаналитика.

В работах Шеннона различаются практическая и теоретическая криптостойкость. Если криптоаналитик даже при наличии всех необходимых средств не имеет возможности по шифротекстам уточнять распределение вероятностей открытых текстов, криптосистему называют теоретически стойкой. При этом предполагается одноразовое (сеансовое) использование секретного ключа.

Совершенная секретность предполагает статистическую независимость всех возможных открытых текстов M и шифротекстов C, делающую невозможной добычу дополнительной информации о посланном открытом тексте при получении шифротекста.

Обозначив $P(C / M)$ условную вероятность получения шифротекста C, подвергнутого шифрованию некоторым неизвестным ключом M, получим

$$P(C/M) = \sum_{k:E_k(M)=C} P(k),$$

где $P(k)$ — вероятность использования ключа k;

E_k — преобразование зашифрования с использованием ключа k.

По большей части для данных M и C имеется хотя бы один ключ k, при котором $E_k(M) = C$. Однако бывают случаи, когда несколько различных ключей позволяют преобразовать текст M в текст C.

Необходимое и достаточное условие совершенной секретности заключается в выполнении условия равновероятности получения открытого текста M в случае перехвата конкретного шифра C для каждого C и для всех M:

$$P(M/C) = P(M).$$

Определение совершенной секретности можно выразить через энтропию следующим образом:

$$H(M/C) = H(M).$$

С учетом факта увеличения неопределенности (энтропии) при уменьшении объема известной информации при рассмотрении множества текстов и множества неизвестных ключей вместе получим соотношение

$$H(M/C) \le H(M,K/C) = H(K/C) + H(M/C,K) =$$
$$= H(K/C) \le H(K)$$

Из соотношения видно естественное отсутствие неопределенности открытого текста M при совместном наличии шифртекста C и ключа K, т.е.

$$H(M/C,K)=0$$

Иными словами, неопределенность секретного ключа не должна быть меньше неопределенности сообщения, которое будет зашифровано с помощью данного ключа. Поэтому при использовании одного и того же алфавита длина ключа k не может быть меньше длины сообщения M. Однако это жесткое условие не является практически удобным вследствие необходимости формирования больших секретных ключей, равных длинам передаваемых сообщений.

В качестве примера совершенно секретной криптосистемы можно привести систему Вернама при случайном равновероятном выборе ключа $k=k_1,...,k_n$:

$$C=M\oplus K=\{M_1\oplus K_1,...,M_n\oplus K_n\}$$

В данном случае $P(k_i)=2^{-n}$ для каждого i=1, ..., n. Поэтому можно сделать вывод о выполнении условия совершенно секретной системы, так как $P(C/M)=2^{-n}$ для всех C и M.

7.2. АНАЛИЗ КРИПТОГРАФИЧЕСКОЙ СТОЙКОСТИ НА ОСНОВЕ ТЕОРИИ СЛОЖНОСТИ

Для оценки практической стойкости криптосистемы вводится рабочая характеристика $W(n)$, которая выражает измеряемое в удобных единицах среднее количество работы для вычисления ключа k при условии наличия известных n знаков шифротекста и использования наилучшего алгоритма для криптоанализа.

Обычно оценка криптосистем производится на основе достигнутого с использованием наилучшего из известных методов дешифрования значения рабочей характеристики $W(\infty)$.

Криптосистемы, которые с использованием всех общедоступных методов сложно или невозможно вскрыть в течение приемлемого (конечного) промежутка времени ($W(\infty)\geq Const$), называют практически стойкими.

Именно это понятие стойкости криптосистем реально применяется на практике, сводя, по сути дела, задачу разработки стойких криптосистем или шифров к проблеме поиска удовлетворяющих определенным условиям наиболее сложных задач. То есть шифр разрабатывается так, чтобы его раскрытие было эквивалентно решению некоторой задачи, требующей выполнения большого объема работы, а стойкость криптосистемы связана с вычислительной сложностью применяемых для шифрования алгоритмов. Иными словами, определяющей является не возможность извлечения из анализа шифротекста информации об открытом тексте, а осуществимость данной возможности в течение приемлемого промежутка времени. В такой ситуации условие совершенной секретности криптосистемы соблюдается даже при использовании секретных ключей, значительно меньших по размерам, чем длина открытого шифруемого текста.

Вычислительная сложность алгоритма зависит от числа n входных данных и измеряется его временно́й сложностью (τ) и емкостной сложностью (S).

Временна́я сложность соответствует времени, затрачиваемому алгоритмом на решение задачи и рассматриваемому в качестве функции размера задачи или числа данных на входе.

Емкостная сложность соответствует емкости необходимой памяти вычислительного средства.

Асимптотические сложности определяются поведением костной и временно́й сложностей в пределе при изменении размера задачи в сторону увеличения. Этими сложностями определяется итоговый размер задачи, которую можно решить конкретным алгоритмом.

Сложность задачи определяется как сложность в худшем случае, когда выбирается наибольшая из сложностей на всех входах данного размера.

Сложность задачи определяется как «средняя», когда выбирается «средняя» сложность на всех входах данного размера. Часто эта сложность находится не так просто, как сложность в худшем случае.

В теории сложности по большей части работают со сложностью задач в худшем случае. Для анализа функционирования алгоритма используются модели вычислительных машин, способные достаточно точно отразить основные свойства, присущие реальным машинам, т.е. такие как:

машины, имеющие произвольный доступ к памяти;

машины, имеющие произвольный доступ к памяти и хранимую программу;

детерминированная машина Тьюринга;

недетерминированная машина Тьюринга.

В детерминированных машинах Тьюринга процесс определения нового состояния машины на следующем шаге зависит от текущего состояния машины и от символа, обозреваемого на ленте головкой. В недетерминированных (вероятностных) машинах Тьюринга на переход машины в новое состояние влияет еще случайная величина, с вероятностью ½ принимающая значения 0 или 1. Можно считать,

что в вероятностной машине Тьюринга добавлена дополнительная «случайная» лента, которую можно читать лишь в одном направлении, с записанной бесконечной двоичной случайной строкой, символы которой влияют на переход в новое состояние.

Временна́я сложность алгоритма определяется как $o(n^2)$ (произносится «сложность порядка n^2»), если входы размера n обрабатываются алгоритмом в течение времени $\tau = cn^2$, где c = const. Если говорить строго математическим языком, то неотрицательная $g(n)$ *есть* $O(f(N))$ при существовании постоянных c и n_0, для которых $g(n) \le c|f(N)|$, где $n \ge n_0$.

Если $g(n) = a_m n^m + a_{m-1} n^{m-1} + \ldots + a_1 n + a_0$, то $g(n)$ — полиномиальная функция степени m. Тогда получаем

$$g(n) = O(n^m).$$

Полиномиальным алгоритмом, или алгоритмом полиномиальной временно́й сложности называется такой алгоритм, который имеет временну́ю сложность, равную $\tau = O(P(n))$, где P(n) — некоторый полином, а n — размер задачи (входа).

Экспоненциальные алгоритмы имеют временну́ю сложность $\tau = o(t(n)P(n))$, где $t(n)$ — некоторая функция; $P(n)$ — полином.

Суперполиномиальные алгоритмы имеют временну́ю сложность $\tau = o(cP(n))$, где c = const; $P(n)$ — полином.

В качестве примера в табл. 7.1 приведено время выполнения для нескольких различных классов алгоритмов при n = 106 с учетом того, что алгоритм будет выполняться на обыкновенной последовательной ЭВМ, имеющей быстродействие 106 операций в секунду.

Таблица 7.1

Время выполнения для различных классов алгоритмов [54]

Класс алгоритма	*Сложность*	*Число операций при* $n = 10^6$	*Реальное время*
Полиномиальный	$O(1)$	1	1 мс
Линейный	$O(n)$	10^6	1 с
Квадратичный	$O(n^2)$	10^{12}	10 дней
Кубический	$O(n^3)$	10^{18}	27 397 лет
Экспоненциальный (суперполиномиальный)	$O(2^n)$	10^{30130}	10^{301016} лет

Таблица 7.1 показывает, что алгоритм сложностью $\tau = o(n^3)$ фактически невозможно выполнить на последовательной машине, в отличие от машины, имеющей 10^6 параллельно работающих процессоров, которой нужно лишь 10 дней, чтобы справиться с подобным алгоритмом. В то же время алгоритм, имеющий сложность $\tau = O(2^n)$, фактически невозможно выполнить даже на машине, имеющей один триллион процессоров, работающих параллельно.

Решаемые задачи можно решить для достаточно большой размерности n за полиномиальное время.

Нерешаемые (трудные) задачи систематически нельзя решить за полиномиальное время.

Для **алгоритмически неразрешимых задач** создание алгоритма решения невозможно. В качестве примера приведем десятую проблему Гильберта, для которой доказано, что для многочлена P с целыми коэффициентами не существует алгоритма, решающего в целых числах уравнение

$$P(x_1, \ldots, x_n) = 0.$$

Рисунок 7.1 отражает классификацию задач на основе степени их сложности, давая возможное (точно неизвестное) наглядное соотношение задач друг с другом.

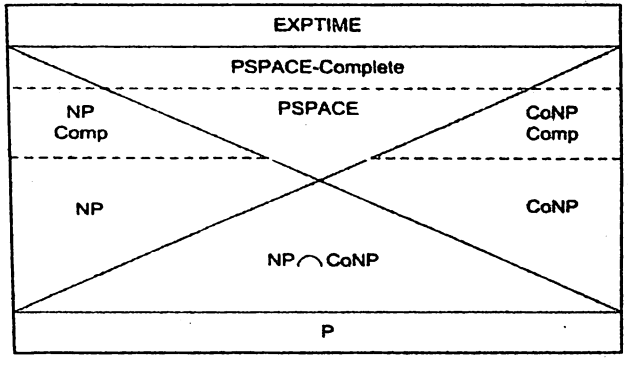


Рис. 7.1. Классификация задач по степени сложности их решения [54]

Все задачи, решение которых можно получить за полиномиальное время, объединены в класс **P** (или **P**-TIME). Все задачи, решение которых на недетерминированной машине Тьюринга, способной параллельно выполнять неограниченное число независимых вычислений, можно получить за полиномиальное время, объединены в класс **NP** (или **NP**-TIME), т.е. проверка варианта решения задачи на подобной машине Тьюринга возможна за полиномиальный проме-

жуток времени. В данном случае о решении задачи речь не идет в связи с отсутствием гарантии проверки машиной верно угаданного решения. На проверку всех вариантов решений и поиск необходимого (иными словами, на решение задачи из **NP**-класса) в некоторых случаях требуется экспоненциальное время. В качестве примера можно привести известную «задачу рюкзака», заключающуюся в поиске (определении существования) в состоящем из n целых чисел множестве $\mathbf{A} = (a_1, \ldots, a_n)$ подмножества чисел, сумма которых была бы равна числу S. Данная задача входит в класс NP-задач вследствие легкости проверки равенства S и суммы чисел любого подмножества из **A** и одновременной сложности поиска такого подмножества. Временна́я сложность проверки всех 2^n существующих подмножеств составляет $\tau = o(2^n)$.

Любую задачу, решение которой получить за полиномиальное время на детерминированной машине Тьюринга, можно решить за полиноминальное время на недетерминированной машине Тьюринга. Поэтому и вследствие отсутствия доказательств того, что $NP=P$ или $NP \neq P$, предполагается, что класс P входит в класс NP.

Сведение задач одна к другой на основе конструктивного преобразования используется в качестве основного метода для доказательства близости задач по сложности. Одним из множества известных примеров подобного преобразования является доказательство С. Куком в 1971 г. теоремы о возможности сведения за полиномиальное время любой задачи из **NP**-класса к задаче о выполнимости, заключающейся в проверке существования для набора дизъюнкций (операция «или») выполняющего набора булевых переменных $V_1, \ldots, V_n$ над этим множеством элементов.

В то же время Р. Карпом было доказано предположение о том, что сложность задачи о выполнимости равняется сложности многих хорошо известных комбинаторных задач (таких как «задача о коммивояжере» о выборе маршрута посещения различных городов так, чтобы в каждом из них побывать только однажды).

В классе *NP-Complete* (**NP**-полных) объединены наиболее трудные **NP**-задачи. Появление какой-либо **NP**-полной задачи в классе **P** докажет равенство NP= P.

Дополнительные задачи для некоторых задач из **NP** собраны в классе **CoNP**. Задачи из класса **CoNP** формулируются как «показать, что решений нет», в отличие от задач из класса **NP**, формулировка которых звучит как «определить, существует ли решение». Равенство **CoNP** = **NP** на данный момент не доказано (неизвестно, верно ли оно), однако известны задачи, которые принадлежат **CoNP**$\cap$**NP** (иными словами, пересечению классов **CoNP** и **NP**). В «задаче о разложении чисел», нашедшей в криптологии широкое применение и являющейся примером такой принадлежащей **CoNP**$\cap$**NP** задачи, необходимо по данному целому числу n, проверить существование делителей p и q, таких, что $n = pq$. Следует отметить гораздо более высокую сложность задачи поиска делителей p и q, чем сложность задачи проверки разложимости числа n, которая решается простой проверкой равенства $n = pq$ при выбранных p и q.

Требующие полиномиальных объемов машинной памяти задачи, которые не всегда возможно решить за полиномиальное время, объединены в класс **PSPACE**, включающий задачи из классов **CoNP** и **NP** (**CoNP-Complete** и **NP-Complete**) и более трудные.

Класс **PSPACE-Complete** (**PSPACE**-полных) включает задачи, обладающие свойством, что их принадлежность классу **NP** или **P** делает верным равенство **NP= PSPACE** или равенство **PSPACE= P**.

В результате применение в основе построения криптосистемы наиболее сложных задач с известными решениями приводит к созданию практически устойчивых шифров, вскрытие которых требует столько времени, что данная процедура лишается практического смысла, хотя и не является невозможной.

ГЛАВА 8. КРИПТОСИСТЕМЫ С ОТКРЫТЫМ КЛЮЧОМ

8.1. ОБЩАЯ СХЕМА ШИФРОВАНИЯ С ОТКРЫТЫМ КЛЮЧОМ

Идея применения методов криптографии с открытым ключом возникла из попыток найти решение двух из наиболее сложных проблем, возникающих при использовании традиционного шифрования.

Первой проблемой является распределение ключей. Распределение ключей при традиционном шифровании требует, чтобы обе участвующие в обмене данными стороны либо уже имели общий ключ, который каким-то образом был им доставлен, либо использовали услуги некоторого центра распределения ключей. Уитфилд Диффи, один из основателей метода шифрования с открытым ключом (вместе с Мартином Хеллманом), считал, что второе из этих требований противоречит самой сущности криптографии — возможности обеспечить полную секретность вашей собственной корреспонденции. Как он заметил: «Какой смысл имеет разработка неприступной криптосистемы, если ее пользователи должны использовать свои секретные ключи совместно с неким центром распределения ключей, который может быть скомпрометирован либо взломщиком, либо судебным решением?».

Второй проблемой, которую сформулировал Диффи и которая, очевидно, не связана с первой, является проблема «цифровых подписей». Если использование криптографии получило очень широкое распространение, и не только в области военного дела, но и в области коммерции и частных коммуникаций, то электронные сообщения и документы нуждаются в эквивалентах подписей, используемых в бумажных документах. Иными словами, можно ли разработать метод, с помощью которого обе стороны могли бы убедиться в том, что цифровое сообщение было отправлено данным конкретным лицом?

Диффи и Хеллман пришли к своему открытию в 1976 г., разработав метод, с помощью которого решались обе вышеупомянутые проблемы и который радикально отличался от всех известных ранее подходов в криптографии за всю ее четырехтысячелетнюю историю.

На рис. 8.1 показана общая схема шифрования с открытым ключом, которая выглядит следующим образом:

1. Каждая конечная система в сети генерирует пару ключей для шифрования и дешифрования получаемых сообщений.

2. Каждая из систем публикует свой ключ шифрования, размещая его в открытом для всех реестре или файле. Это и есть открытый ключ. Второй ключ, соответствующий открытому, остается в личном владении.

3. Если пользователь **A** собирается послать сообщение пользователю **B**, то он шифрует сообщение, используя открытый ключ пользователя **B**.

4. Когда пользователь **B** получит сообщение, он дешифрует его с помощью своего личного ключа. Другой получатель не сможет дешифровать сообщение, поскольку личный ключ **B** знает только **B**.

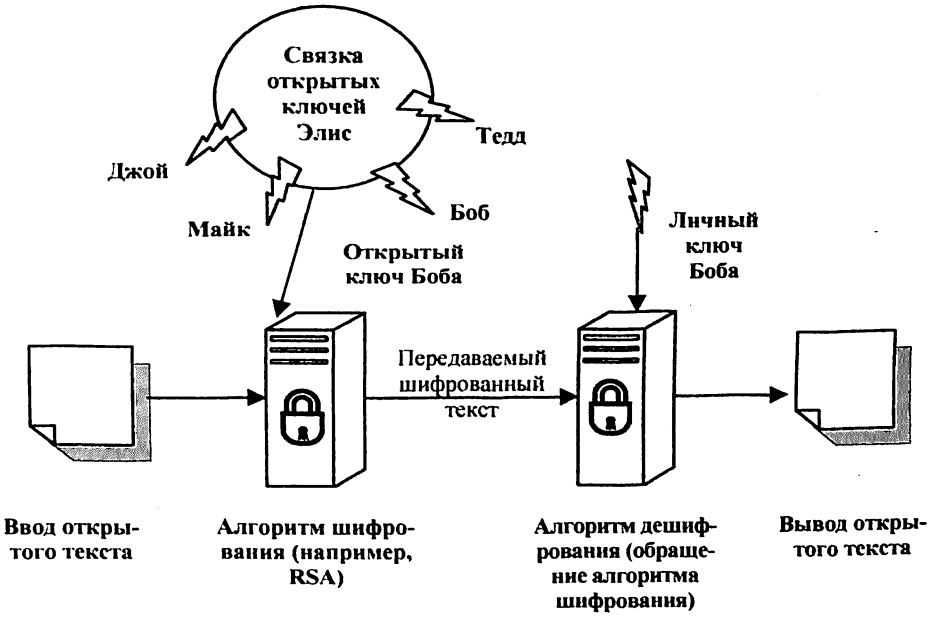


Рис. 8.1. Шифрование с открытым ключом [34]

В рамках этого подхода все участники имеют доступ к открытым ключам, а личные ключи генерируются на месте каждым участником для себя, и поэтому их никогда не приходится распределять. До тех пор пока системе удается хранить свой личный ключ в секрете, поступающие сообщения остаются защищенными. В любой момент система может изменить свой личный ключ и создать соответствующий ему открытый ключ, заменяющий старый открытый ключ.

Таким образом, алгоритмы шифрования с открытым ключом зависят от одного ключа шифрования и другого, связанного с первым ключа дешифрования. Эти алгоритмы имеют следующую важную

особенность: с точки зрения вычислений нереально определить ключ дешифрования, зная только используемый криптографический алгоритм и ключ шифрования. (Кроме того, некоторые алгоритмы, например, RSA, имеют еще одну особенность: любой из этих двух связанных ключей может служить для шифрования, и тогда другой может применяться для дешифрования).

В табл. 8.1 сравниваются некоторые важные характеристики традиционного шифрования и шифрования с открытым ключом. Чтобы различать эти два подхода, далее будем называть ключи, предназначенные для традиционного шифрования, секретными ключами. Два ключа, с помощью которых осуществляется шифрование с открытым ключом, будут называться соответственно открытым ключом и личным ключом. Личный ключ, конечно, тоже должен храниться в секрете, но называется личным, а не секретным, во избежание путаницы с ключом схемы традиционного шифрования.

Таблица 8.1

Традиционное шифрование и шифрование с открытым ключом [34]

Традиционное шифрование	*Шифрование с открытым ключом*
Необходимо для **работы**:	Необходимо для **работы**:
1. Один алгоритм с одним и тем же ключом служит и для шифрования, и для дешифрования.	1. И для шифрования, и для дешифрования используется один алгоритм, но два ключа: один ключ для шифрования, другой — для дешифрования.
2. Отправитель и получатель должны использовать одинаковые алгоритм и ключ.	2. Отправитель и получатель должны иметь по одному из пары соответствующих ключей (не один и тот же).
Необходимо для **защиты**:	Необходимо для **защиты**:
1. Ключ должен сохраняться в *секрете*.	1. Один из двух ключей должен храниться в секрете.
2. Должно быть невозможно или, по крайней мере, практически невозможно расшифровать сообщение при отсутствии дополнительной информации.	2. Должно быть невозможно или, по крайней мере, практически невозможно, расшифровать сообщение при отсутствии дополнительной информации.
3. Знания алгоритма и наличия образцов шифрованного текста должно быть недостаточно для того, чтобы восстановить ключ	3. Знания алгоритма, одного из ключей и наличия образцов шифрованного текста должно быть недостаточно для того, чтобы восстановить второй ключ

Рассмотрим основные элементы схемы шифрования с открытым ключом более подробно, используя рис. 8.2.

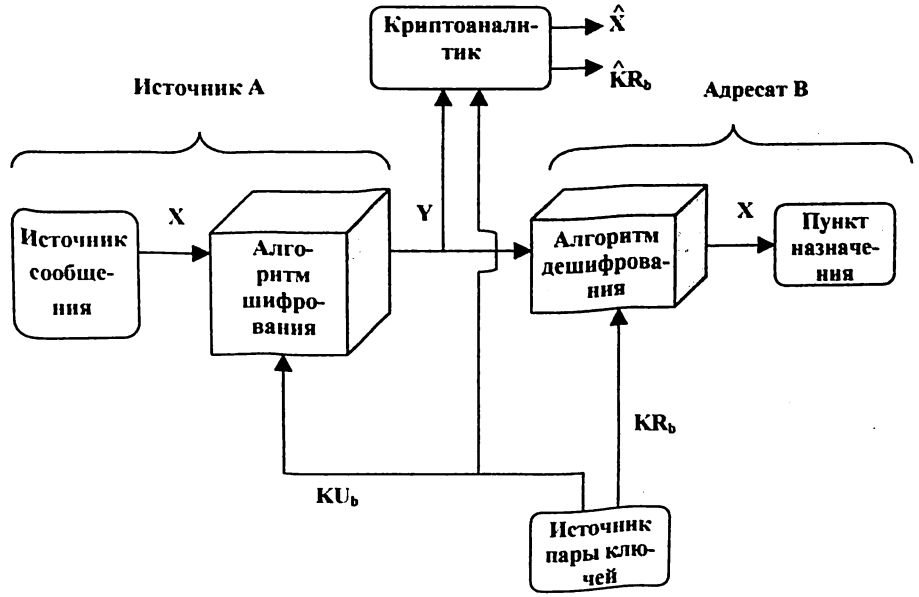


Рис. 8.2. Криптосистема с открытым ключом: защита [34]

Итак, имеется некоторый источник сообщений **A**, создающий сообщение $\mathbf{X} = [X_1, X_2, \ldots, X_M]$ в виде открытого текста. Все M элементов **X** являются буквами некоторого конкретного алфавита. Сообщение адресовано получателю **B**. Адресат **B** генерирует указанную пару ключей: открытый ключ $\mathbf{KU_b}$ и личный ключ $\mathbf{KR_b}$. Ключ $\mathbf{KR_b}$ известен только пользователю **B**, тогда как ключ $\mathbf{KU_b}$ публично доступен и, таким образом, оказывается доступным отправителю **A**.

Имея сообщение **X** и ключ шифрования $\mathbf{KU_b}$ в качестве входных данных, отправитель **A** формирует шифрованный текст $\mathbf{Y} = [Y_1, Y_2, \ldots, Y_N]$ следующим образом:

$$\mathbf{Y}=\mathbf{E}_{\mathbf{KU_b}}(\mathbf{X}).$$

Предполагаемый получатель, владея соответствующим личным ключом, может обратить это преобразование:

$$\mathbf{X}=\mathbf{D}_{\mathbf{KR_b}}(\mathbf{Y}).$$

Противник, наблюдая **Y** и имея доступ к $\mathbf{KU_b}$, но не к $\mathbf{KR_b}$ или **X**, должен будет пытаться восстановить **X** и / или $\mathbf{KR_b}$. Предполагается, что противник знает алгоритмы шифрования (**E**) и дешифрования (**D**). Если противника интересует только данное конкретное сообщение, то он сосредоточит свои усилия на восстановлении **X** с помощью получения оценок для открытого текста. Но часто противник бывает заинтересован в возможности чтения и последующих сообщений. В таком случае будет предпринята попытка восстановить ключ $\mathbf{KR_b}$ путем генерирования оценки и для него.

8.2. ЭЛЕКТРОННАЯ ЦИФРОВАЯ ПОДПИСЬ И АУТЕНТИФИКАЦИЯ В КРИПТОСИСТЕМАХ С ОТКРЫТЫМ КЛЮЧОМ

Уже упоминалось, что любой из двух связанных ключей может применяться для шифрования, при этом предполагается, что другой будет служить для дешифрования. Это дает возможность реализовать различные криптографические схемы. Если схема на рис. 8.2 обеспечивает конфиденциальность, то рис. 8.3 и 8.4 показывают, как использовать шифрование с открытым ключом для аутентификации:

$$Y=E_{KR_a}(X); \qquad X=D_{KU_a}(Y).$$

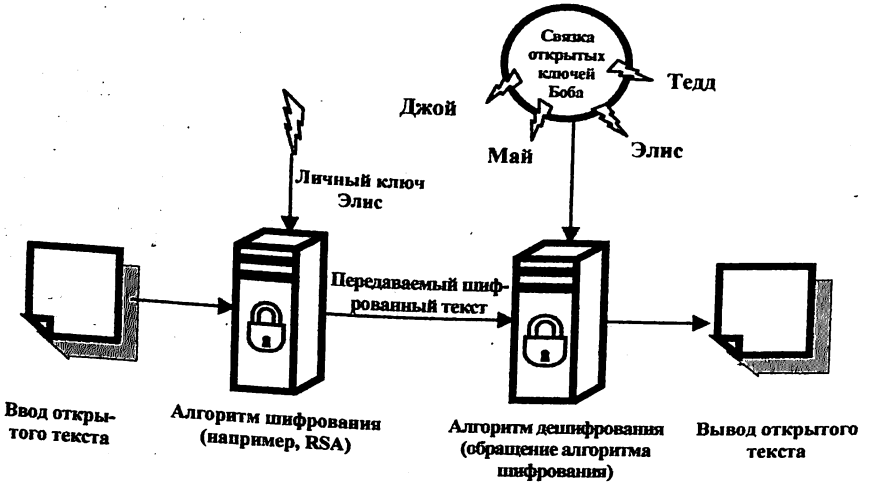


Рис. 8.3. Аутентификация с использованием открытого ключа [34]

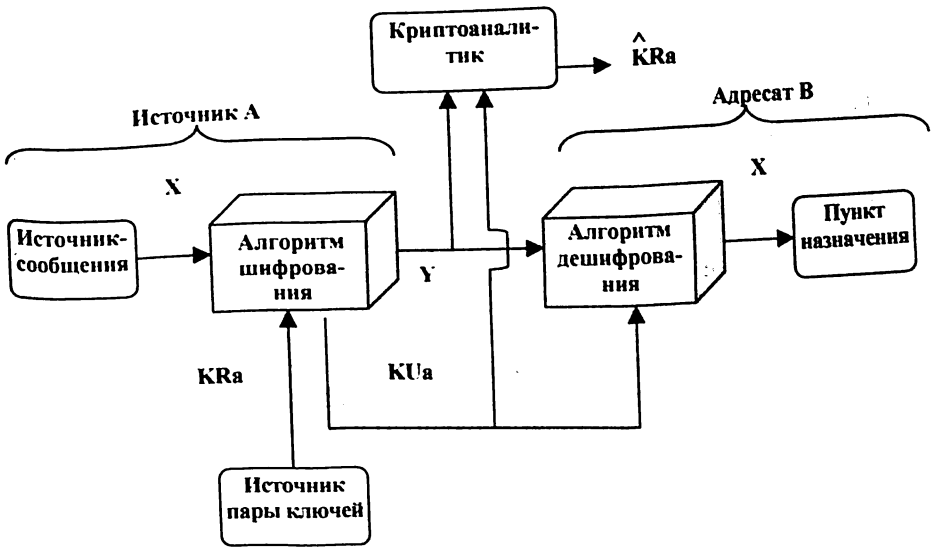


Рис. 8.4. Криптосистема с открытым ключом: аутентификация [34]

В этом случае отправитель **A** готовит сообщение адресату **B** и перед отправлением шифрует это сообщение с помощью личного ключа пользователя **A**. Получатель **B** может дешифровать это сообщение, используя открытый ключ **A**. Ввиду того что сообщение было зашифровано личным ключом отправителя **A**, только он мог подготовить это сообщение. Поэтому в данном случае все шифрованное сообщение выступает в качестве цифровой подписи. Кроме того, невозможно изменить сообщение без доступа к личному ключу пользователя **A**, поэтому сообщение решает и задачу идентификации отправителя, и задачу подтверждения целостности данных.

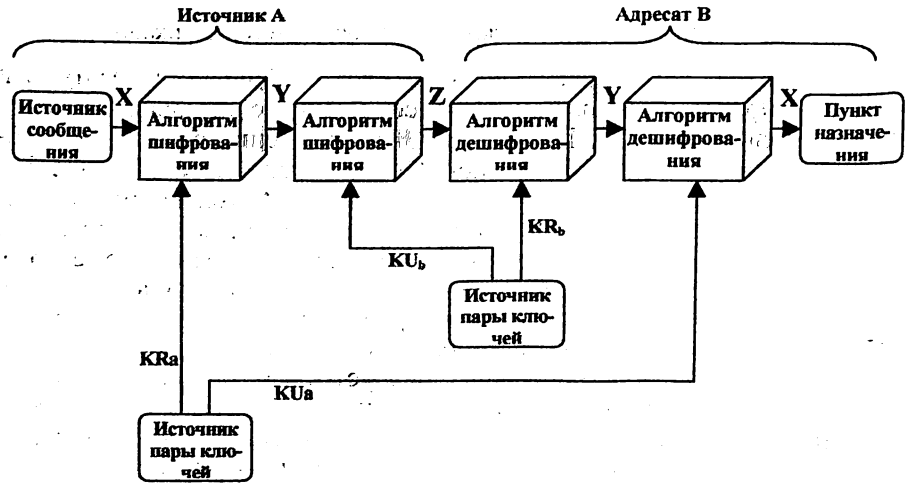
В схеме, о которой идет речь, шифруется все сообщение, которое хотя и идентифицирует отправителя и подтверждает целостность содержимого, требует достаточно много памяти. Каждый документ, чтобы его можно было использовать, должен храниться в виде открытого текста. Еще один экземпляр документа должен сохраняться в виде шифрованного текста, чтобы в случае спора можно было восстановить источник и содержимое оригинала. Более эффективным способом достижения того же результата является шифрование небольшого блока битов, являющегося функцией документа. Такой блок, называемый аутентификатором, должен иметь такое свойство, чтобы нельзя было изменить документ без изменения жестко связанного с ним аутентификатора. Если аутентификатор будет зашиф-

рован с использованием личного ключа отправителя, он будет служить подписью, удостоверяющей источник, содержимое и порядок отправки сообщения.

Важно подчеркнуть, что только что описанный процесс шифрования не обеспечивает конфиденциальность. Это означает, что пересылаемому таким образом сообщению гарантирована защита от изменения, но не от перехвата. Это очевидно в том случае, когда для формирования подписи используется только часть сообщения, поскольку тогда остаток сообщения передается открытым текстом. Но даже в случае полностью зашифрованного сообщения, как показано на рис. 8.4, конфиденциальность не обеспечивается ввиду того что любой сторонний наблюдатель может расшифровать передаваемое сообщение с помощью открытого ключа отправителя.

Однако можно обеспечить как аутентификацию, так и конфиденциальность повторным использованием схемы шифрования с открытым ключом (рис. 8.5).

$$Z=E_{KU_b}[E_{KR_a}(X)]$$



$$Z=D_{KU_a}[D_{KR_b}(Z)]$$

Рис. 8.5. Криптосистема с открытым ключом: защита и аутентификация [34]

В данном случае процедура начинается, как и прежде, с шифрования сообщения с помощью личного ключа отправителя. Это обеспечивает цифровую подпись. Затем результат шифруем снова, но уже используя открытый ключ получателя. Конечный шифрованный текст сможет дешифровать только предполагаемый адресат, поскольку только он один имеет соответствующий личный ключ. Таким образом, конфиденциальность обеспечивается. Неудобство этого подхода заключается в том, что алгоритм шифрования с открытым ключом, который оказывается весьма сложным, должен при каждой передаче данных применяться четыре раза, а не два.

8.3. ОСОБЕННОСТИ ПРИМЕНЕНИЯ КРИПТОСИСТЕМ С ОТКРЫТЫМ КЛЮЧОМ

Далее необходимо разъяснить один аспект криптосистем с открытым ключом, который в противном случае может породить недоразумения. Криптосистемы с открытым ключом характеризуются использованием криптографического алгоритма с двумя ключами, один из которых остается в личном пользовании, а второй открыт для всех. В зависимости от ситуации отправитель использует либо свой личный ключ, либо открытый ключ получателя, либо же оба, если требуется выполнить какую-то специальную криптографическую функцию. В самых широких пределах использование криптосистем с открытым ключом можно отнести к трем категориям:

Шифрование / дешифрование. Отправитель шифрует сообщение с использованием открытого ключа получателя.

Цифровая подпись. Отправитель «подписывает» сообщение с помощью своего личного ключа. Подпись получается в результате применения криптографического алгоритма к сообщению или к небольшому блоку данных, являющихся функцией сообщения.

Обмен ключами. Две стороны взаимодействуют, чтобы создать сеансовый ключ. При этом возможно несколько различных подходов, предполагающих применение личных ключей одной или обеих сторон.

Одни алгоритмы подходят для всех трех типов применения, тогда как другие предназначены только для одной или двух из этих категорий. Возможности применения алгоритмов приведены в табл. 8.2.

Таблица 8.2

Применение криптосистем с открытым ключом [34]

Алгоритм	*Шифрование/ дешифрование*	*Цифровая подпись*	*Обмен ключами*
RSA	Да	Да	Да
Диффи-Хеллмана	Нет	Нет	Да
DSS	Нет	Да	Нет

Криптосистема, варианты которой показаны на рис. 8.1–8.5, зависит от криптографического алгоритма, предполагающего применения двух связанных ключей. Диффи и Хеллман предположили без доказательства, что такие алгоритмы существуют. Однако они указали условия, которым такие алгоритмы должны удовлетворять:

Для стороны **B** процесс генерации пары ключей (открытый ключ $\mathbf{KU_b}$ и личный ключ $\mathbf{KR_b}$) не должен порождать вычислительные трудности.

Для отправителя **A** не должен порождать вычислительные трудности процесс создания шифрованного текста при наличии открытого ключа и сообщения **M**, которое требуется зашифровать:

$$\mathbf{C} = \mathbf{E}_{\mathbf{KU_b}}(\mathbf{M}).$$

Для получателя **B** не должен порождать вычислительные трудности процесс дешифрования полученного шифрованного текста с помощью личного ключа с целью восстановления оригинального сообщения:

$$\mathbf{M} = \mathbf{D}_{\mathbf{KR_b}}(\mathbf{C}) = \mathbf{D}_{\mathbf{KR_b}}\left[\mathbf{E}_{\mathbf{KU_b}}(\mathbf{M})\right].$$

Для противника должно быть невозможным с точки зрения вычислений восстановление личного ключа $\mathbf{KR_b}$ из имеющегося открытого ключа $\mathbf{KU_b}$.

Для противника должно быть невозможным с точки зрения вычислений восстановление оригинального сообщения **M** из имеющихся открытого ключа $\mathbf{KU_b}$ и шифрованного текста **C**.

К этим требованиям можно добавить еще одно, которое, хотя и представляется полезным, не является необходимым для всех приложений, реализующих криптосистемы с открытым ключом:

Функции шифрования и дешифрования могут применяться в любом порядке:

$$\mathbf{M} = \mathbf{E}_{\mathbf{KU_b}}[\mathbf{D}_{\mathbf{KR_b}}(\mathbf{M})].$$

Перечисленные требования в своей совокупности весьма сложны для выполнения, что подтверждается тем, что за несколько десятилетий, прошедших со времени открытия метода криптографии с открытым ключом, только один такой алгоритм получил широкое признание.

Перед тем как приступить к разъяснению причин, по которым эти требования оказываются сложными для выполнения, попытаемся представить их в некотором более удобном виде. Эти требования сводятся к необходимости нахождения некоторой односторонней функции с лазейкой.

Односторонней функцией называется функция, отображающая свои аргументы в некоторый диапазон значений так, что каждое значение функции вычислить легко, а обратное — практически невозможно:

$\mathbf{Y}=f(\mathbf{X})$ вычисляется легко,

$\mathbf{X}=f^{-1}(\mathbf{Y})$ практически не поддается вычислению.

В общем случае термин «легко вычислимый» означает, что проблема решается за так называемое полиномиальное время, рассматриваемое как функция длины вводимого значения. Так, если длина вводимого значения равна n битов, то полиномиальное время, требуемое для вычисления функции, пропорционально n^a, где a является фиксированной константой. Такие алгоритмы называются алгоритмами, принадлежащими к классу **P** (более подробно этот вопрос рассмотрен ранее в гл. 7). Термин «практически невычислимый» обозначает более размытое понятие. В общем случае можно сказать, что функция является практически невычислимой, если усилия по ее вычислению возрастают быстрее, чем полиномиальная функция от длины вводимого значения. Например, если вводимое значение имеет длину nбитов, и время, требуемое для вычисления функции, пропорционально 2^n, то такая функция считается практически невычислимой. К сожалению, для каждого конкретного алгоритма очень непросто выяснить, отражает ли этот алгоритм указанную степень сложности. К тому же в определении обычного понятия вычислительной сложности алгоритма рассматриваются оценки сложности либо в наихудшем случае, либо в среднем. Для криптографии эти оценки бесполезны, криптография требует практической невычислимости обратной функции почти для любого вводимого значения, а не для наихудшего или среднего случая.

Вернемся к определению односторонней функции с лазейкой — такая функция предполагается легко вычислимой в одном направлении и практически невычислимой в другом в отсутствие дополнительной информации. При наличии дополнительной информации

обратная функция может быть вычислена за полиномиальное время. Подводя итог, можно сказать так: односторонние функции с лазейкой являются такими обратимыми функциями f_k, для которых:

$\mathbf{Y}=f_k(\mathbf{X})$ вычисляется легко, если известны k и $\mathbf{X}$;

$\mathbf{X}=f_k^{-1}(\mathbf{Y})$ вычисляется легко, если известны k и $\mathbf{Y}$;

$\mathbf{X}=f_k^{-1}(\mathbf{Y})$ практически не поддается вычислению, если $\mathbf{Y}$ известно, а k — нет.

Таким образом, для разработки практически применимой схемы шифрования с открытым ключом требуется найти подходящую одностороннюю функцию с лазейкой.

8.4. КРИПТОАНАЛИЗ СИСТЕМ С ОТКРЫТЫМ КЛЮЧОМ

Как и в случае традиционного шифрования, схема шифрования с открытым ключом уязвима с позиции анализа с перебором всех ключей. И контрмеры здесь те же — использование длинных ключей. Однако в данном случае имеются и другие варианты защиты. Криптосистемы с открытым ключом зависят от некоторой обратимой математической функции со специальными свойствами. Сложность вычисления такого рода функций может зависеть не линейно от числа битов в ключе, а расти быстро. Поэтому длина ключа должна быть достаточно большой для того, чтобы сделать анализ с перебором всех возможных ключей практически невыполнимым, но достаточно малой для того, чтобы на практике можно было использовать операции шифрования и дешифрования. Предлагаемые для использования на практике длины ключей, конечно же, обеспечивают практическую неэффективность анализа с перебором всех ключей, но при этом скорость шифрования/дешифрования оказывается слишком медленной, чтобы соответствующие алгоритмы можно было рекомендовать для универсального применения. Поэтому шифрование с открытым ключом в настоящее время ограничивается областями управления ключами и приложениями цифровой подписи.

Другой формой атаки является попытка найти способ вычисления личного ключа по известному открытому ключу. На сегодняшний день нет математического доказательства невозможности такой формы атаки ни для одного алгоритма шифрования с открытым ключом. В этом отношении любой конкретный алгоритм данного типа, включая широко распространенный алгоритм RSA, оказывается не внушающим доверия. А история криптографии показывает, что проблема, которая выглядит неразрешимой, может оказаться вполне разрешимой, если посмотреть на нее с некоторой другой, совершенно новой точки зрения.

Наконец, существует форма атаки, применимая только к системам с открытым ключом. Эта форма, по существу, является формой анализа с вероятным текстом сообщения. Предположим, например, что должно быть отправлено сообщение, состоящее исключительно из 56-битового ключа DES. Противник может зашифровать все возможные ключи, используя открытый ключ, что позволит ему дешифровать подобное сообщение с помощью простого сравнения образцов с передаваемым шифрованным текстом. В данном случае не имеет значения, насколько велика длина ключа схемы шифрования с открытым ключом — анализ сводится к простому перебору 56-битовых ключей. Возможности такого рода анализа можно воспрепятствовать присоединением к указанным простым сообщениям случайных битов.

ГЛАВА 9. ОСНОВНЫЕ ПОНЯТИЯ ТЕОРИИ ЧИСЕЛ

9.1. ДЕЛИТЕЛИ И ПРОСТЫЕ ЧИСЛА

Последующее изложение будет опираться на перечисляемые далее обозначения и понятия:

- **N** (множество натуральных чисел, представленное целыми положительными числами вида 1, 2, ...);
- **Z** (множество целых чисел, включающее числа вида n, $-n$ и 0, где n — натуральное число);
- **Q** (множество рациональных чисел, включающее множество целых чисел **Z**, которое, в свою очередь, включает множество **N**, и представленное числами числа вида p/q, где p и q — целые числа и $q \neq 0$) [43].

Рассмотрим множество целых чисел, обозначаемое через **Z**. На множестве целых чисел рассмотрим операции сложения «+» и умножения «×». Операция деления, обратная операции умножения, выполняется не для всех пар чисел из **Z**. (Напомним, **Z** — целые числа).

Число a делится на число $b \neq 0$, если существует число q такое, что

$$\frac{a}{b} = q \quad или \quad a = bq$$

В этом случае говорят, что число b делит число a.

При этом число b — делитель числа a, число a — кратное числа b, число q — частное от деления a на b. Число 0 делится на любое целое $b \neq 0$. Если $a \neq 0$, то очевидно, что множество всех делителей числа a конечно.

Утверждение о том, что b делит a, обозначают символом b/a. Если b не делит a, то этот факт обозначают $b \nmid a$.

Лемма 1. Если c/b и b/a, то c/a.

Доказательство: по определению из c/b и b/a имеем, что $b = q_1c$, $a = q_2b$, откуда $a = q_1q_2c = qc$, и снова по определению c/a.

Лемма 2. Если $m = a + b$, d/m и d/a, то d/a.

Доказательство: по определению: $m=q_1d$ и $a=q_2d$. Поэтому из равенства $m= a+b$ получим $b= q_1d-q_2d= (q_1 - q_2)d= qd$, откуда следует d/b. Отсюда следует, что если $m= a_1+a_2+\ldots+a_{n-1}+a_n$ и d делит числа $m, a_1,\ldots,a_{n-1}$, то d/a_n.

Общим делителем двух или нескольких чисел называется число, являющееся делителем каждого из этих чисел.

Если $a_1,\ldots,a_n$ — числа из **Z** и хотя бы одно из них не равно нулю, то множество их общих делителей конечно и среди этих делителей существует наибольшее число.

Наибольшим общим делителем (НОД) чисел $a_1,\ldots,a_n$ называется наибольший из их общих делителей. Он обозначается как $(a_1,\ldots,a_n)$.

Число $n > 1$ называется простым, если оно не имеет других делителей, кроме 1 и n. Например, числа 2, 3, 5, 7 являются простыми, так как они делятся только на 1 и на самих себя.

Число n называется составным, если оно имеет делитель, отличный от 1 и n. Например, числа 4, 6, 8 – составные числа.

Если НОД$(a_1,\ldots,a_n)=1$, то числа $a_1,\ldots,a_n$ называют взаимно простыми.

Например, (2, 5) = 1; (10, 21) = 1.

Лемма 3. Если целое число b взаимно просто с каждым из целых чисел $a_1,\ldots,a_n$, то b взаимно просто с их произведением $a_1 \times a_2 \times a_n$.

Когда a не делится на b, то принято говорить о делении a на b с остатком r.

Теорема *о делении с остатком*. Если a и b целые числа, и $b > 0$, то существуют единственные целые числа q и r такие, что $a = b \times q + r$, $0 \le r < b$.

Число q называют неполным частным при делении a на b, число r называют *остатком* от деления a на b.

Очевидно, что если $r=0$, то b/a.

Пример. Пусть $b = 12$. Тогда для чисел $a = 110$, $a = -53$, $a = 156$ имеем

$a = b \times q + r$:

$110 = 12 \times 9 + 2$	$0 < r = 2 < b = 12$,
$-53 = 12 \times (-5) + 7$	$0 < r = 7 < b = 12$,
$156 = 12 \times 13 + 0$	$r = 0$.

Любое натуральное составное число n представляется в виде $n = ab$, $1 < a < n$, $1 < b < n$.

Лемма 4. Наименьший отличный от единицы делитель натурального числа $n > 1$ есть простое число.

Доказательство. Число $n > 1$ имеет хотя бы два различных делителя (1 и n) и, следовательно, имеет делитель, отличный от единицы. Среди всех делителей, отличных от 1, имеется наименьший. Пусть это будет q. Число q должно быть простым, так как в противном случае оно было бы составным и по определению имело бы делитель q_1 такой, что $1 < q_1 < q$. Но q_1/q и q/n, а тогда по лемме 1 имеем q_1/n. Это противоречит тому, что q наименьший делитель n, значит q — простое число.

Следствие. Каждое натуральное число $n>1$ имеет хотя бы один простой делитель.

Лемма 5. Если p — простое число, то любое целое число a либо взаимно простое с p, либо делится наp, т.е. p/a.

Доказательство. Наибольшим общим делителем (a, p) чисел a и p может быть только делитель простого числа p, который, в свою очередь, может быть равен 1 или p. Тогда в первом случае $(a, p) = 1$ и числа a и p взаимно простые, а во втором случае, когда $(a, p) = p$, a делится на p, т.е. p / a.

Основная теорема арифметики. Любое натуральное число $n > 1$ представляется в виде произведения простых чисел, причем — единственным образом.

Доказательство. По лемме 4 число $n > 1$ имеет наименьший *простой* делитель p_1,тогда $n = p_1a_1$. Если $a_1 > 1$, то аналогично получим $a_1 = p_2a_2$, где p_2 — наименьший простой делитель числа $a_1 > 1$. Поскольку числа a_1, a_2,... убывают, то на некотором r–м шаге будем иметь $a_{r-1} = p_ra_r$, где $a_r = 1$, т.е. $a_{r-1} = p_r$.

Перемножая все полученные таким образом равенства, после сокращения на a_1, a_2,..., a_{r-1} получим разложение числа n на простые сомножители:

$n = p_1 \times p_2 \times \dots p_r$.

Пусть $n > 1$ и оно представлено в виде разложения на простые сомножители:

$n = p_1 \times p_2 \times \dots p_r$.

Среди чисел $p_1,\dots,p_r$ могут быть одинаковые. Пусть $p_1,\dots,p_k$ — различные из чисел $p_1,\dots,p_r (r > k)$, а $\alpha_1,\dots,\alpha_k$ — кратности, с которыми они входят в исходное разложение. Тогда это разложение можно записать в виде

$$n = p_1^{\alpha 1} \cdot p_2^{\alpha 2} \cdot \dots \cdot p_k^{\alpha k}, \begin{cases} p_i - \text{простое} \\ \alpha_i - \text{целые числа} \end{cases},$$

и называется оно каноническим разложением числа n на простые множители.

Пример. $261360 = 2^4 \cdot 3^3 \cdot 5 \cdot 11^2$.

Согласно теореме Евклида, множество простых чисел бесконечно.

Для того чтобы составить таблицу всех простых чисел, можно использовать метод, называемый ***решетом Эратосфена*** (древнегреческий математик):

напишем одно за другим числа 2, 3, ..., N;

число 2 является простым – оставляем, и зачеркиваем после него все числа, кратные 2, т.е. все четные числа;

следующим за числом 2 является число 3, которое является простым. Оставляем 3, зачеркиваем все числа, кратные 3;

продолжая этот процесс, находим все простые числа, не превышающие заданного числа N.

Например, $N = 40$: 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40.

Простые числа: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37.

Построение в настоящее время таблицы простых чисел показывает, что с ростом их величин они встречаются все реже и реже. Например, в первой сотне чисел (N = 100) их 25, во второй — 21, в третьей — 16 и т.д. В первой 1000 их 168, во второй тысяче — 135, в третьей — 120 и т.д.

Особый класс простых чисел составляют числа вида $2^n - 1$. Такие числа называют ***простыми числами Мерсенна,*** и они являются наибольшими по своему размеру среди других известных простых чисел. Во времена Эйлера наибольшим простым числом было пятое число Мерсенна $2^{31} - 1$. Через сто лет было найдено шестое число Мерсенна $2^{61} - 1$. В 1992 г. найдено 32-е число Мерсенна (его запись содержит 227 832 цифры и требует около ста страниц текста).

Исследователи из проекта распределенных вычислений GIMPS (Great Internet Mersenne Prime Search) работают над обнаружением самых больших на сегодняшний день простых чисел. Разработчики программного обеспечения GIMPS и участники проекта уже поделили приз в 100 тысяч долларов за простое число, длина десятичной записи которого превысит 10 миллионов символов, от фонда Electronic Frontier Foundatio при известном предыдущем открытом числе с 9808358 разрядов. Проверка проходила на двух компьютерах с процессором Pentium IV 2,4 гигагерца.

Следующий приз – $150 000 за число с как минимум 100 миллионами десятичных разрядов

Простые числа распределены в натуральном ряде чисел очень нерегулярно. Согласно основной теореме арифметики, простые числа являются теми простейшими объектами, из которых с помощью умножения строятся все натуральные числа, большие единицы. Поэтому вопросы, связанные с распределением простых чисел, являются важнейшими в теории чисел. Для их изучения была введена функция $\Pi(x)$, выражающая число простых чисел, меньших x. Известно, например, что $\Pi(1) = 0$, $\Pi(2) = 1$, $\Pi(10) = 4$, $\Pi(40) = 12$. Однако для функции $\Pi(x)$ неизвестна никакая простая формула, позволяющая изучать ее поведение. Существуют лишь определенные оценки для функции $\Pi(x)$ (оценки Чебышева, теоремы об асимптотическом законе распределения простых чисел Римана и др.). Наряду с этим в теории чисел уделяется большое внимание решению аддитивных задач, связанных с возможностью представления натуральных чисел в виде суммы простых чисел. Наиболее известная из таких задач —

проблема Гольдбаха (1742 г.). Гольдбах в письме к Л. Эйлеру высказал предположение, что всякое нечётное число, большее 6, можно представить в виде суммы *трех* простых чисел. Эйлер, в свою очередь, высказал более сильную гипотезу: всякое число, начинающееся с 4, можно представить в виде суммы *двух* простых чисел.

Первым результатом, связанным с этой проблемой, была теорема о том, что существует постоянная k такая, что каждое натуральное число, большее единицы, есть сумма не более чем k простых чисел.

Далее была доказана теорема (И.М. Виноградов), утверждающая, что всякое достаточно большое нечетное число представимо в виде суммы трех простых чисел.

Утверждение о представлении четных чисел в виде суммы двух простых чисел до сегодняшнего дня не доказано.

9.2. АРИФМЕТИКА В КЛАССАХ ВЫЧЕТОВ

Далее продолжается описание основных определений и теорем, содержащихся в теории чисел [43].

Целые числа a и b ($a, b \in \mathbf{Z}$) называют сравнимыми по модулю $n \in \mathbf{N}$ при условии делимости разности a и b на n и записывают

$$A = b \ (\mathrm{mod}\ n).$$

Числа a и b сравнимы по модулю n тогда и только тогда, когда существует целое $t (t \in \mathbf{Z})$ такое, что $a=b+nt$; также и только тогда, когда они имеют одинаковые остатки r при делении на n, т.е. $a = nq_1 + r$, $b = nq_2 + r$.

Отсюда следует, что все целые числа $\mathbf{Z}$ по модулю n разбиваются по n непересекающихся классов сравнимых между собой чисел (т.е. имеющих одинаковые остатки при делении на n).

Каждое число r, входящее в какой-либо из классов, называется вычетом числа a по модулю n, а каждый класс — классом вычетов по модулю n. Число разных классов вычетов равно n.

Полной системой вычетов (по модулю n или m) называют множество из m (или n) целых чисел $\{r_1,...,r_m\}$, если для любого целого числа a существует точно одно число r_i $(i=1,...,m)$ из множества $\{r_1,...,r_m\}$ такое, что $a = r_i (\mathrm{mod}\, m)$.

Пример: множество $\{0, 1, 2, \ldots, m-1\}$.

Свойства сравнений по модулю m (или n) и свойства обыкновенных числовых равенств похожи друг на друга:

1. Пару чисел, сравнимых с третьим, можно сравнить между собой.

2. Для сравнения по одному и тому же модулю справедлива следующая формула почленного сложения:

$$[a_1 + a_2](\mathrm{mod}\, m) = [a_1(\mathrm{mod}\, m) + a_2(\mathrm{mod}\, m)](\mathrm{mod}\, m).$$

Для сравнения по одному и тому же модулю справедлива следующая формула почленного умножения:

$$[a_1 \cdot a_2](\mathrm{mod}\, m) = [a_1(\mathrm{mod}\, m) \cdot a_2(\mathrm{mod}\, m)](\mathrm{mod}\, m).$$

Существует некоторое отличие между правилами сокращения сравнений и правилами сокращения равенств.

Если $aU \equiv aV (\mathrm{mod}\ m)$ и $(a, m) = 1$, то $U \equiv V (\mathrm{mod}\ m)$ (9.1)

Если $aU \equiv aV (\mathrm{mod}\, am)$, *то* $U \equiv V (\mathrm{mod}\, m)$.

Для лучшего понимания, приведем пример, когда условие не выполняется:

$$6 \times 3 = 18 \equiv 2 \ (\mathrm{mod}\ 8),$$
$$6 \times 7 = 42 \equiv 2 \ (\mathrm{mod}\ 8),$$
$$\text{однако } 3 \neq 7 \ (\mathrm{mod}\ 8).$$

К такому странному результату привел тот факт, что для произвольно выбранного модуля сравнения m (или n) после перемножения чисел от 0 до $(m-1)$ и множителя a не формируется полный набор всех вычетов при условии наличия у a и m общих множителей.

Алгоритм Евклида. Одной из основных процедур теории чисел является алгоритм Евклида, представляющий собой простую процедуру определения наибольшего общего делителя двух положительных целых чисел.

Алгоритм Евклида опирается на следующую Теорему:

для любого неотрицательного целого числа a и любого положительного целого числа b справедливо следующее:

$$(a, b) = (b, a(\mathrm{mod} b)). \qquad (9.2)$$

Чтобы убедиться в этом, положим $d = (a, b)$. Тогда по определению НОД (,), имеем $d|a$ и $d|b$. Для любого положительного целого числа b значение a можно представить в форме:

$$a = kb + r \equiv r \ (\mathrm{mod}\ b),$$
$$a(\mathrm{mod} b) \equiv r.$$

Поэтому $a \bmod b = a - kb$ для некоторого целого числа k. Но поскольку $d|b$, d делит и kb. Из условия $d|a$ следует также $d|(a \bmod b)$. Это доказывает, что d является общим делителем для b и $a \bmod b$. И на-

оборот, если d является общим делителем для b и $a \bmod b$, то $d \mid kb$ и, таким образом, $d \mid [kb + (a \bmod b))]$, что эквивалентно $d \mid a$. Итак, множество общих делителей a и b совпадает с множеством общих делителей b и $(a \bmod b)$. Поэтому и наибольший общий делитель первой пары чисел будет равен наибольшему общему делителю второй, что завершает доказательство.

Чтобы определить наибольший общий делитель, равенство (9.2) можно использовать повторно.

Пример. $(18, 12) = (12, 6) = (6, 0) = 6$,
$(11, 10) = (10, 1) = (1, 0) = 1$.

В алгоритме Евклида многократно применяется равенство (9.2) для определения наибольшего общего делителя, например, следующим образом.

Пример. Чтобы найти (1970, 1066), выполним следующие действия:

$1970 = 1 \times 1066 + 904$	(1066,904)
$1066 = 1 \times 904 + 162$	(904,162)
$904 = 5 \times 162 + 94$	(162,94)
$162 = 1 \times 94 + 68$	(94,68)
$94 = 1 \times 68 + 26$	(68,26)
$68 = 2 \times 26 + 16$	(26,16)
$26 = 1 \times 16 + 10$	(16,10)
$16 = 1 \times 10 + 6$	(10,6)
$10 = 1 \times 6 + 4$	(6,4)
$6 = 1 \times 4 + 2$	(4,2)
$4 = 2 \times 2 + 0$	(2,0)

Следовательно, $(1970, 1066) = 2$.

В настоящее время в криптографии получил широкое применение обобщенный алгоритм Евклида, который определяет:

1. Наибольший общий делитель двух положительных целых чисел и, если эти числа оказываются взаимно простыми.
2. Мультипликативное обратное одного из них по модулю другого.

9.3. ТЕОРЕМА ЭЙЛЕРА

Функция Эйлера. Прежде чем представить теорему Эйлера, необходимо определить одну важную функцию, используемую в теории чисел и называемую функцией Эйлера.

Она обозначается символом $\varphi(n)$ и представляет собой для каждого целого числа n, $n > 1$, число положительных целых значений, которые меньше n и являются взаимно простыми с n.

В табл. 9.1 представлено 30 первых значений $\varphi(n)$. Значение $\varphi(1)$ оказывается при этом неопределенным, но считается, что оно равно 1.

Таблица 9.1

Некоторые значения функции Эйлера $\varphi(n)$

n	$\varphi(n)$	n	$\varphi(n)$	n	$\varphi(n)$
1	1	11	10	21	12
2	1	12	4	22	10
3	2	13	12	23	22
4	2	14	6	24	8
5	4	15	8	25	20
6	2	16	8	26	12
7	6	17	16	27	18
8	4	18	6	28	12
9	6	19	18	29	28
10	4	20	8	30	8

Ясно, что для простого p

$$\varphi(p) = p - 1.$$

Теперь предположим, что имеется два простых числа p и q. Тогда для $n = pq$ получим

$$\varphi(n) = \varphi(pq) = \varphi(p) \times \varphi(q) = (p - 1) \times (q - 1).$$

Чтобы увидеть это, рассмотрим множество значений $\mathbf{Z}_n$, представляющее собой

$\{0, 1, \ldots, (pq - 1)\}$.

Значениями, не являющимися *взаимно простыми* с n, будут значения множеств

$\{p, 2p, \ldots, (q-1)p\}$ и $\{q, 2q, \ldots, (p-1)q\}$, а также 0.

Соответственно имеем

$$\varphi(n) = pq - [(q - 1) + (p - 1) + 1] = pq - (p + q) + 1 = (p - 1) \times (q - 1) = \varphi(p) \times \varphi(q).$$

Теорема Эйлера. Название функции $\varphi(n)$ происходит от следующей теоремы Эйлера:

для любых взаимно простых чисел a и n (т.е. таких, что $(a, n) = 1$)

$$a^{\varphi(n)} \equiv 1 (\mathrm{mod}\ n). \qquad (9.3)$$

Доказательство. Напомним, что $\varphi(n)$ равно числу положительных целых значений, меньших n и взаимно простых с n. Рассмотрим множество таких целых чисел, пронумеровав их следующим образом:

$$R=\{x_1, x_2, \ldots, x_{\varphi(n)}\}.$$

Теперь умножим каждый элемент этого множества на a по модулю n:

$S = \{(ax_1 \mathrm{m}(\mathrm{mod}\ n)), (ax_2(\mathrm{mod}\ n)), \ldots, (ax_{\varphi(n)}(\mathrm{mod}\ n))\}$.

Приведенное выше множество представляет перестановку элементов множества **R** по следующим причинам:

Поскольку a, как и x_i, является взаимно простым с n, ax_i тоже должно быть взаимно простым с n. Таким образом, все элементы S являются целыми числами, меньшими n и взаимно простыми с n.

Во множестве S нет повторений. Если $ax_i(\mathrm{mod}\ n) = ax_j(\mathrm{mod}\ n)$, то $x_i = x_j$ (см.(9.1).

Поэтому

$$\prod_{i=1}^{\varphi(n)} (ax_i \mathrm{mod} n) = \prod_{i=1}^{\varphi(n)} x_i,$$

$$\prod_{i=1}^{\varphi(n)} ax_i \equiv \prod_{i=1}^{\varphi(n)} x_i\ (\mathrm{mod} n),$$

$$a^{\varphi(n)} \times \prod_{i=1}^{\varphi(n)} x_i \equiv \prod_{i=1}^{\varphi(n)} x_i\ (\mathrm{mod} n),$$

$$a^{\varphi(n)} \equiv 1\ (mod\ n).$$

Полезной оказывается и следующая альтернативная формулировка теоремы:

$$a^{\varphi(n)+1} \equiv a(\mathrm{mod}\ n). \qquad (9.4)$$

Из теоремы Эйлера можно вывести следствие, с помощью которого демонстрируется эффективность алгоритма RSA. Для любых двух простых чисел p и q и чисел $n = pq$ и m (здесь $0 < m < n$) выполняется следующее условие:

$$m^{\varphi(n)+1} = m^{(p-1)(q-1)+1} \equiv m \bmod n. \qquad (9.5)$$

Если $(m, n) = 1$, т.е. если m и n являются взаимно простыми, то указанное условие выполняется в силу теоремы Эйлера (см. (9.3)). Предположим, что $(m, n) \neq 1$. Что это означает? Поскольку $n = pq$, равенство $(m, n) = 1$ эквивалентно логическому выражению [(m не кратно p) **И** (m не кратно q)]. Если m кратно p, то m и n имеют общий простой множитель p и поэтому не будут взаимно простыми, а если m кратно q, то m и n имеют общий простой множитель q и поэтому не будут взаимно простыми. Таким образом, выражение $(m, n) \neq 1$ оказывается эквивалентным отрицанию вышеуказанного логического выражения. Поэтому $(m, n) \neq 1$ эквивалентно логическому выражению

[(m кратно p) **ИЛИ** (m кратно q)].

Рассмотрим случай, когда m кратно p, так что $m = cp$ для некоторого положительного целого числа c. В этом случае имеем $(m, q) = 1$, поскольку иначе m кратно p, m кратно q и $m < pq$. Если $(m, q) = 1$, то применима теорема *Эйлера*, и поэтому

$$m^{\varphi(q)} \equiv 1(\mathrm{mod}\ q).$$

Но по правилам арифметики в классах вычетов

$$[m^{\varphi(q)}]^{\varphi(p)} \equiv 1(\mathrm{mod}\ q), m^{\varphi(n)} \equiv 1(\mathrm{mod}\ q).$$

Поэтому найдется такое целое число k, что

$$m^{\varphi(n)} \equiv 1 + kq.$$

Умножив обе части этого равенства на $m = cp$, получим

$$m^{\varphi(n)+1} = m + kcpq = m + kcn,$$

$$m^{\varphi(n)+1} \equiv m(\mathrm{mod}\ n).$$

Подобный ряд аргументов используется и для случая, когда m является кратным q.

Полезной будет и следующая альтернативная форма того же следствия:

$$[m^{\varphi(n)}]^k \equiv 1\ (\mathrm{mod}\ n),$$
$$m^{k\varphi(n)} \equiv 1 \times (\mathrm{mod}\ n),$$
$$m^{k\varphi(n)+1} = m^{k(p-1)(q-1)+1} \equiv m(\mathrm{mod}\ n).$$

Из теоремы Эйлера следует *малая теорема Ферма*:

Если p — простое числои $p \nmid a$ (p – число, не делящееся на a), то

$$a^{p-1} \equiv 1(\mathrm{mod} p).$$

Это утверждение очевидно, если вспомнить, что для простого числа p

$$\varphi(p) = p - 1.$$

Результаты приведенных выше теорем могут быть использованы и для определения простых чисел, что, как следует из рассмотрения п. 9.1, является весьма непростой задачей.

Так, воспользовавшись малой теоремой Ферма для заданного N, можно перебрать все возможные значения a, и если какое-то из них нарушит условие теоремы Ферма, то N следует признать составным. Но, к сожалению, теорема Ферма дает лишь необходимое, но не достаточное условие простоты: имеются составные числа, удовлетворяющие указанному условию, они носят имя Кармайкла (*Carmichael*).

Для проверки чисел на простоту существует ряд тестов, среди которых наиболее популярным является тест Рабина.

Тест Рабина заключается в следующем.

Пусть имеется число $N = 2^S t + 1$, где t — нечетно, и требуется установить, простое оно или составное. Выбирается случайное число $1 \le a \le N$ и проверяются два условия:

1) N не делится на a;

2) $a^t = 1 \times (\text{mod } N)$ или существует целое $k (0 \le k \le s)$, для которого

$$a^{2t} = -1 \ (\text{mod } N).$$

Если оба условия выполняются, то, как показал Рабин, вероятность того, что число N окажется все-таки составным, равна ¼. Если же провести не один, а n аналогичных тестов, выбирая случайное a, то можно установить простоту числа с вероятностью 4^{-n}.

Несмотря на вероятностный характер предложенного алгоритма, за счет выбора большого n можно добиться того, что вероятность ошибочного выбора простого числа для создания криптосистемы окажется ниже, чем вероятность ее взлома существующими методами (например, пробой на ключ).

Показатель числа a по модулю n. Если a и n являются взаимно простыми ($(a, n) = 1$), то существует по крайней мере одно целое число γ, удовлетворяющее соотношению $a^\gamma \equiv 1 (\text{mod } n)$, – это число $\gamma = \varphi(n)$, существование которого обеспечивается теоремой Эйлера. Наименьшее из чисел γ называется показателем числа a по модулю n. Показатель числа a по модулю n является делителем числа $\varphi(n)$.

В частном случае, когда показатель числа a по модулю n равен $\varphi(n)$ (наивысший из показателей числа a по модулю n), то a называется первообразным (примитивным) корнем по модулю n.

Для наименьшего из положительных γ, при которых выполняется условие $a^\gamma \equiv 1 \ (\text{mod } n)$, используются также еще следующие названия:

порядок числа a по модулю n;

показатель, которому принадлежит a по модулю n;

длина периода последовательности, генерируемой степенями a.

Чтобы убедиться в истинности последнего пункта, рассмотрим степени числа 7 по модулю 19:

$7^1 = 7 \text{ mod } 19,$
$7^2 = 49 = 2 \times 19 + 11 = 11 \text{ mod } 19,$
$7^3 = 343 = 18 \times 19 + 1 = 1 \text{ mod } 19,$
$7^4 = 2401 = 126 \times 19 + 7 = 7 \text{ mod } 19,$
$7^5 = 16807 = 884 \times 19 + 11 = 11 \text{ mod } 19.$

Нет смысла продолжать вычисления дальше, поскольку последовательность повторяется. В этом можно убедиться, заметив, что $7^3 = 1 \ (\text{mod } 19)$ и, таким образом, $7^{3+j} = 7^3 \times 7^j = 7^j \ (\text{mod } 19)$. Поэтому любые две степени числа 7, показатели которых отличаются на 3 (или на число, кратное 3), сравнимы по модулю19. Другими словами, эта последовательность является периодической с периодом, равным наименьшему положительному показателю n, при котором $7^n = 1 \ (\text{mod } 19)$.

В табл. 9.2 представлены степени числа a по модулю 19 для всех положительных a. Длина последовательности для каждого из значений a указывается с помощью затенения ячеек.

Необходимо обратить внимание на следующие моменты:

А) Все последовательности заканчиваются числом 1.

Б) Длина последовательности является делителем $\varphi(19) = 18$. Из этого следует, что в каждой строке таблицы умещается целое число периодов соответствующих последовательностей.

В) Некоторые последовательности имеют длину 18. В таком случае говорят, что целое число a генерирует (своими степенями) множество всех ненулевых вычетов по модулю 19. Любое из таких целых чисел называют первообразным корнем по модулю 19 (см. определение выше).

Важность понятия первообразного корня по модулю n определяется тем фактом, что если a является первообразным корнем n, его степени $a, a^2, \ldots, a^{\varphi(n)}$ оказываются различными по модулю n и взаимно простыми с n.

В частности, для простого числа p, если a является первообразным корнем p, то $a, a^2, \ldots, a^{p-1}$ оказываются различными по модулю p.

Для простого числа 19 его первообразными корнями являются числа 2, 3, 10, 13, 14 и 15.

Не все целые числа имеют первообразные корни: целыми числами с первообразными корнями будут только числа 2, 4, p^a и $2p^a$, где p — любое нечетное простое число.

Таблица 9.2

Степени целых чисел по модулю 19

a	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}	a^{15}	a^{16}	a^{17}	a^{18}
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1
3	9	8	5	15	7	2	6	18	16	10	11	14	4	12	17	13	1
4	16	7	9	17	11	6	5	1	4	16	7	9	17	11	6	5	1
5	6	11	17	9	7	16	4	1	5	6	11	17	9	7	16	4	1
6	17	7	4	5	11	9	16	1	6	17	7	4	5	11	9	16	1
7	11	1	7	11	1	7	11	1	7	11	1	7	11	1	7	11	1
8	7	18	11	12	1	8	7	18	11	12	1	8	7	18	11	12	1
9	5	7	6	16	11	4	17	1	9	5	7	6	16	11	4	17	1
10	5	12	6	3	11	15	17	18	9	14	7	13	16	8	4	2	1
11	7	1	11	7	1	11	7	1	11	7	1	11	7	1	11	7	1
12	11	18	7	8	1	12	11	18	7	8	1	12	11	18	7	8	1
13	17	12	4	14	11	10	16	18	6	2	7	15	5	8	9	3	1
14	6	8	17	10	7	3	4	18	5	13	11	2	9	12	16	15	1
15	16	12	9	2	11	13	5	18	4	3	7	10	17	8	6	14	1
16	9	11	5	4	7	17	6	1	16	9	11	5	4	7	17	6	1
17	4	11	16	6	7	5	9	1	17	4	11	16	6	7	5	9	1
18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1

9.4. ДИСКРЕТНЫЕ ЛОГАРИФМЫ

Для обычных положительных действительных чисел логарифм является функцией, обратной возведению в степень. Аналогичная функция существует и в арифметике классов вычетов.

Свойства обычного логарифмирования. Логарифм числа определяется как степень, в которую нужно возвести значение положительного основания (не равного 1), чтобы получить данное число, т.е. для заданного основания x и произвольного y: $y = x^{\log_x(y)}$.

Далее перечислены основные свойства логарифмов:

$\log_x(1)=0$,

$\log_x(x)=1$,

$\log_x(yz)=\log_x(y)+\log_x(z)$,

$\log_x(y^r)=r\log_x(y)$.

Определение и свойства дискретного логарифмирования. Рассмотрим первообразный корень некоторого простого числа p (подобные аргументы могут быть использованы и в случае с числами, не являющимися простыми). В этом случае степени числа a с показателями от 1 до $(p-1)$ порождают каждое целое число от 1 до $(p-1)$ в точности по одному разу. Также известно, что любое целое число b можно представить в форме

$$b \equiv r \pmod p, \text{ где } 1 \le r \le (p-1)$$

в классах вычетов. Отсюда вытекает, что для любого целого числа b и любого первообразного корня a простого числа p можно найти ровно один показатель степени i, для которого

$$b \equiv a^i \pmod p, \text{ где } 1 \le i \le (p-1).$$

Значение этого показателя называют **индексом числа b по модулю p при основании a.** Записывается это значение как $\text{ind}_{a,p}(b)$.

Необходимо обратить внимание на следующий момент:

$\text{ind}_{a,p}(1)=0$, поскольку $a^0 \pmod p = 1 \pmod p = 1$,

$\text{ind}_{a,p}(a)=1$, поскольку $a^1 \pmod p = a$.

Теперь рассмотрим:

$$x = a^{ind_{a,p}(x)} \bmod p$$

$$y = a^{ind_{a,p}(y)} \bmod p$$

$$xy = a^{ind_{a,p}(xy)} \bmod p$$

Воспользовавшись правилами умножения по модулю сравнения, получим

$$a^{ind_{a,p}(xy)} \bmod p = (a^{ind_{a,p}(x)} \bmod p)(a^{ind_{a,p}(y)} \bmod p) =$$
$$= a^{ind_{a,p}(x)+ind_{a,p}(y)} \bmod p$$

Теперь применим теорему Эйлера, которая утверждает, что для любых взаимно простых каждого a и n имеет место формула

$$a^{\varphi(n)} \equiv 1 \pmod n.$$

Любое положительное целое число z может быть представлено в виде $z = q + k\varphi(n)$. Поэтому по теореме Эйлера имеем

$$a^z \equiv a^q \pmod n, \text{ если } z = q \pmod{\varphi(n)}.$$

Используя это соотношение совместно с предыдущим, получим равенство

$$\mathrm{ind}_{a,p}(xy) = [\mathrm{ind}_{a,p}(x) + \mathrm{ind}_{a,p}(y)] \bmod \varphi(p),$$

обобщив которое, получаем

$$\mathrm{ind}_{a,p}(y^r) = [r\,\mathrm{ind}_{a,p}(y)] \bmod \varphi(p).$$

Это указывает на аналогию между обычными логарифмами и индексами — по этой причине последние часто называют дискретными логарифмами.

Следует иметь в виду, что однозначно дискретные логарифмы по модулю n при основании a определяются, только когда a является первообразным корнем n.

Таблица 9.3

Таблицы дискретных логарифмов по модулю 19

(а) Дискретные логарифмы по модулю 19 при основании2

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\mathrm{ind}_{2,19}(a)$	18	1	13	2	16	14	6	3	8	17	12	15	5	7	11	4	10	9

(б) Дискретные логарифмы по модулю 19 при основании 3

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\mathrm{ind}_{3,19}(a)$	18	7	1	14	4	8	6	3	2	11	12	15	17	13	5	10	16	9

(в) Дискретные логарифмы по модулю 19 при основании 10

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\mathrm{ind}_{10,19}(a)$	18	17	5	16	2	4	12	15	10	1	6	3	13	11	7	14	8	9

(г) Дискретные логарифмы по модулю 19 при основании 13

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\mathrm{ind}_{13,19}(a)$	18	11	17	4	14	10	12	15	16	7	6	3	1	5	13	8	2	9

(д) Дискретные логарифмы по модулю 19 при основании 14

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\mathrm{ind}_{14,19}(a)$	18	13	7	8	10	2	6	3	14	5	12	15	11	1	17	16	14	9

(е) Дискретные логарифмы по модулю 19 при основании 15

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
$\mathrm{ind}_{15,19}(a)$	18	5	11	10	8	16	12	15	4	13	6	3	7	17	1	2	12	9

В табл. 9.3, данные которой получены из табл. 9.2, показано множество дискретных логарифмов, которые могут быть определены по модулю 19.

Вычисление дискретных логарифмов. Рассмотрим уравнение

$$y \equiv g^x (\bmod p).$$

Вычисление y при заданных g, x и p является простым делом. В самом худшем случае придется выполнить x повторных умножений, для чего имеются достаточно эффективные алгоритмы. Но если заданы y, g и p, то вычисление x из указанного выше соотношения — т.е. дискретное логарифмирование — является, вообще говоря, очень непростой задачей. По сложности эта задача сравнима с задачей разложения больших чисел на простые множители (что требуется для RSA) и имеет экспоненциальную сложность.

В настоящее время сложность наиболее быстрого из известных алгоритмов вычисления дискретных логарифмов по модулю простого числа оценивается величиной порядка $e^{((\ln p)^{1/3}\ln(\ln p))^{2/3}}$, что для больших простых чисел оказывается за пределами практических возможностей современных вычислительных средств.

ГЛАВА 10. АЛГОРИТМ ШИФРОВАНИЯ RSA

10.1. СТРУКТУРА АЛГОРИТМА RSA

Схема Райвеста – Шамира – Адлемана (RSA) в настоящее время является единственной, получившей широкое признание и практически применяемой схемой шифрования с открытым ключом. [34]

Схема RSA представляет собой блочный шифр, в котором и открытый текст, и шифрованный текст представляются целыми числами из диапазона от 0 до $n-1$ для некоторого n.

Открытый текст шифруется блоками, каждый из которых содержит двоичное значение, меньшее некоторого заданного числа n. Это значит, что длина блока должна быть не больше $\log_2(n)$. На практике длина блока выбирается равной 2^k битам, где $2^k < n < 2^{k+1}$. Схема, разработанная Райвестом, Шамиром и Адлеманом, основана на выражениях со степенями. Шифрование и дешифрование для блока открытого текста **M** и блока шифрованного текста **C** можно представить в виде следующих формул:

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

Как отправитель, так и получатель должны знать значение n. Отправитель знает значение e, и только получателю известно значение d. Таким образом, данная схема является алгоритмом шифрования с открытым ключом $\mathbf{KU} = \{e, n\}$, и личным ключом $\mathbf{KR} = \{d, n\}$.

Чтобы этот алгоритм мог использоваться для шифрования с открытым ключом, должны быть выполнены следующие требования:

Должны существовать такие значения e, d и n, что $\mathbf{M}^{ed} = \mathbf{M} \pmod n$ для всех $\mathbf{M} < n$.

Должны относительно легко вычисляться $\mathbf{M}^e$ и $\mathbf{C}^d$ для всех значений $\mathbf{M} < n$.

Должно быть практически невозможно определить d по имеющимся e и n.

Проанализируем сначала первое требование, а остальные рассмотрим позже. Необходимо найти соотношение вида

$\mathbf{M}^{ed} = \mathbf{M} \pmod n$.

Здесь как нельзя лучше подойдет следствие из теоремы Эйлера: для таких любых двух простых чисел p и q и таких любых двух целых чисел n и m, что $n = pq$ и $0 < m < n$, и произвольного целого числа k выполняются следующие соотношения:

$$m^{k\varphi(n)+1} = m^{k(p-1)(q-1)+1} \equiv m \pmod n,$$

где $\varphi(n)$ является функцией Эйлера, значение которой равно числу положительных целых чисел, меньших n и взаимно простых с n.

В случае простых p и q имеем $\varphi(pq) = (p-1)(q-1)$. Поэтому требуемое соотношение получается при условии

$$ed = k \times \varphi(n) + 1.$$

Это эквивалентно следующим соотношениям:

$$ed \equiv 1 \pmod{\varphi(n)},$$
$$d \equiv e^{-1} \pmod{\varphi(n)},$$

т.е. e и d являются взаимно обратными по модулю $\varphi(n)$. Обратите внимание, что в соответствии с правилами арифметики в классах вычетов это может иметь место только тогда, когда d (а следовательно и e) является взаимно простым с $\varphi(n)$. В эквивалентной записи $(\varphi(n), d) = 1$.

Теперь у нас есть все, чтобы представить схему RSA. Компонентами схемы являются:

p и q — два простых числа (секретные, выбираются);

$n = pq$ (открытое, вычисляется);

такое e, что $(\varphi(n), e) = 1, 1 < e < \varphi(n)$ (открытое, выбирается);

$d \equiv e^{-1} \pmod{\varphi(n)}$ (секретное, вычисляется).

Личный ключ складывается из $\{d, n\}$, а открытый — из $\{e, n\}$. Предположим, что пользователь **A** опубликовал свой открытый ключ, и теперь пользователь **B** собирается переслать ему сообщение **M**.

Тогда пользователь **B** вычисляет шифрованное сообщение

$$\mathbf{C} = \mathbf{M}^e \pmod n$$

и пересылает шифр **C**.

Получив этот шифрованный текст, пользователь **A** дешифрует его, вычисляя

$$\mathbf{M} = \mathbf{C}^d \pmod n.$$

Имеет смысл привести здесь обоснование этого алгоритма. Были выбраны e и d такие, что

$d \equiv e^{-1} \pmod{\varphi(n)}$. Таким образом, $ed \equiv 1 \pmod{\varphi(n)}$.

Значит, ed имеет вид $k\varphi(n)+1$. Но по следствию теоремы Эйлера, для таких любых двух простых чисел p и q и целых чисел $n = pq$ и **M**, что $0 < \mathbf{M} < n$, выполняются соотношения

$$\mathbf{M}^{k\varphi(n)+1} = \mathbf{M}^{k(p-1)(q-1)+1} \equiv \mathbf{M} \pmod n.$$

Поэтому

$$\mathbf{M}^{ed} = \mathbf{M} \pmod n.$$

Теперь имеем

$$\mathbf{C}=\mathbf{M}^e(\text{mod } n),$$
$$\mathbf{M}=\mathbf{C}^d(\text{mod } n)\equiv (\mathbf{M}^e)^d (\text{mod } n)\equiv \mathbf{M}^{ed}(\text{mod } n)\equiv \mathbf{M} (\text{mod } n).$$

Таблица 10.1 резюмирует алгоритм RSA, а на рис. 10.1 показан пример его применения. В этом примере ключи вычисляются следующим образом:

1. Выбираются два простых числа: $p = 7$ и $q = 17$.
2. Вычисляется $n = pq = 7 \times 17 = 119$.
3. Вычисляется $\varphi(n) = (p - 1)(q - 1) = 96$.
4. Выбирается e, взаимно простое с $\varphi(n) = 96$ и меньшее, чем $\varphi(n)$; в данном случае $e = 5$.
5. Определяется такое d, что $de = 1 \pmod{96}$ и $d < 96$. Соответствующим значением будет $d = 77$, так как $77 \times 5 = 385 = 4 \times 96 + 1$.
6. В результате получаются открытый ключ **KU**= {5, 119} и личный ключ **KR** = {77, 119}.

В данном примере показано использование этих ключей с вводимым открытым текстом **M** = 19. При шифровании 19 возводится в пятую степень, что в результате дает 2 476 099. В результате деления на 119 определяется остаток, равный 66. Следовательно, $19^5 = 66 (\text{mod } 119)$, и поэтому шифрованным текстом будет 66. После дешифрования выясняется, что

$$66^{77} \equiv 19 \pmod{119}.$$

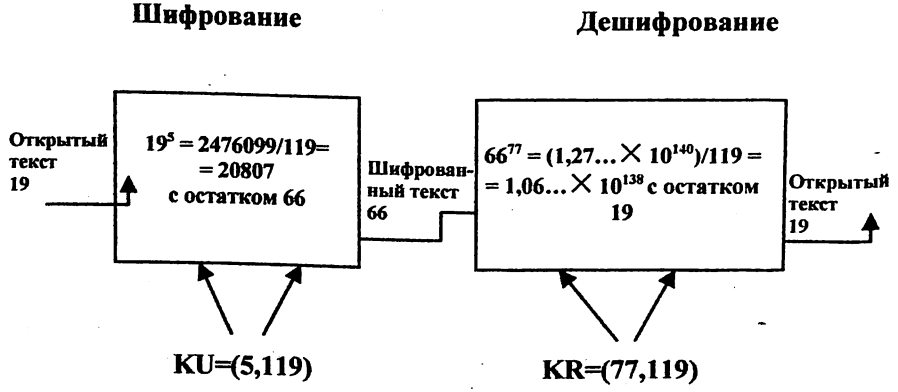


Рис. 10.1. Пример реализации алгоритма RSA [34]

Таблица 10.1

Алгоритм RSA[34]

Вычисление ключей	
Выбор $p.q$	p и q должны быть простыми
Вычисление $n=p \times q$	
Вычисление $\varphi(n) = (p - 1)(q - 1)$	
Выбор целого e	$(\varphi(n),e)= 1,\ 1< e <\varphi(n)$
Вычисление d	$d\equiv e^{-1} \bmod\varphi(n)$
Открытый ключ	**KU** = $\{e, n\}$
Личный ключ	**KR** =$\{d, n\}$
Шифрование	
Открытый текст:	**M**<n
Шифрованный текст:	**C** =$\mathbf{M}^e$ (mod n)
Дешифрование	
Открытый текст:	C
Дешифрованный текст:	**M** =$\mathbf{C}^d$(mod n)

10.2. ВЫЧИСЛИТЕЛЬНАЯ РЕАЛИЗАЦИЯ АЛГОРИТМА RSA

Теперь рассмотрим проблему сложности вычислений, необходимых при использовании RSA.

Здесь следует рассмотреть два вопроса: вычисление ключей и шифрование / дешифрование. Сначала исследуем процесс шифрования и дешифрования, а потом обратимся к вычислению ключей.

10.2.1. ШИФРОВАНИЕ И ДЕШИФРОВАНИЕ

Как шифрование, так и дешифрование в RSA предполагают использование операции возведения целого числа в целую степень по модулю n. Если возведение в степень выполнять непосредственно с целыми числами и только потом проводить сравнение по модулю, то промежуточные значения окажутся просто огромными. К счастью, здесь можно воспользоваться свойствами арифметики в классах вычетов:

$$[(a \bmod n)(b \bmod n)] \bmod n = (a \times b) \bmod n.$$

Таким образом, можно рассматривать промежуточные результаты по модулю n. Это делает вычисления практически возможными.

Другой проблемой является эффективная реализация операции возведения в степень, так как в случае применения RSA имеют дело с потенциально большими показателями. Чтобы продемонстрировать, насколько здесь можно увеличить эффективность вычислений, предположим, что необходимо вычислить x^{16}. Прямолинейный подход потребует 15 умножений, как показано ниже:

$$x^{16} = x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x.$$

Однако такой же конечный результат можно получить и с помощью всего четырех умножений, если повторно возводить в квадрат промежуточные результаты, получая при этом x^2, x^4, x^8, x^{16}.

В общем случае значение $(a^x \bmod n)$, где a и x являются положительными целыми числами, вычисляется достаточно эффективно с помощью известной схемы Горнера «слева направо»

$$a^x (\bmod n) = (((a^{x_{k-1}})^2 \cdot a^{x_{k-2}})^2 \cdot \ldots \cdot a^{x_1})^2 \cdot a^{x_0} (\bmod n),$$

или «справа налево»

$$a^{x_0} \cdot (a^2)^{x_1} \cdot \ldots \cdot (a^{2^{k-1}})^{x_{k-1}} \bmod (n)$$

при двоичном разложении числа x в виде

$$x = \sum_{i=0}^{k-1} x_i 2^i$$

Учитывая, что ряд значений x_i при этом равен 0, даже при больших числах x из интервала (1, n) вычисление

$f(x) = a^x \pmod n$

всегда можно осуществить не более чем за $0[\log_2 n]$ операций.

10.2.2. ВЫЧИСЛЕНИЕ КЛЮЧЕЙ

Перед тем как обратиться к криптосистеме с открытым ключом, каждая сторона должна генерировать пару ключей. Это означает выполнение следующих задач:

- определение двух простых чисел p и q;
- выбор одного из чисел e или d и вычисление второго.

Сначала рассмотрим процедуру выбора p и q. Ввиду того что значение $n = pq$ будет известно любому потенциальному противнику, для того чтобы не допустить возможности нахождения p и q с помощью простого перебора вариантов, простые числа должны быть выбраны из достаточно большого множества (т.е. p и q должны быть большими числами). В то же время метод нахождения больших простых чисел должен быть практически эффективным.

На сегодняшний день нет хороших методов вычисления произвольно больших простых чисел, поэтому для решения этой проблемы приходится прибегать к различным хитростям. Чаще всего процедура заключается в выборе случайного нечетного числа приблизительно желаемой величины и выяснения, является ли это число простым. Если окажется, что число простым не является, выбирается следующее случайное число, пока не будет найдено простое.

Для проверки того что числа простые, существует целый ряд тестов. Почти все такие тесты носят вероятностный характер. Это значит, что тест определит только, что данное целое число, вероятно, простое. Несмотря на отсутствие полной уверенности, такие тесты могут выполняться так, чтобы обеспечить уверенность с вероятностью, как угодно близкой к 1. Одним из наиболее эффективных и популярных таких алгоритмов является алгоритм Миллера – Рабина (*Miller – Rabin*), рассмотренный в гл. 9. Для него и для большинства других подобных алгоритмов процедура проверки простоты целого числа n состоит в выполнении ряда вычислений, в которых используется также и некоторое случайно выбранное целое число a. При этом:

– если n не выдерживает тестирования, то n простым не является;

– если n выдерживает одно тестирование, то n может оказаться простым, а может и не быть простым;

– если же n успешно проходит целый ряд таких «испытаний» различными случайно выбранными значениями a, это дает большую степень уверенности в том, что n на самом деле является простым числом.

Обобщенно процедуру выбора простого числа можно представить в следующем виде:

1. Выберите нечетное целое число n некоторым случайным образом (например, используя генератор псевдослучайных чисел).
2. Выберите целое число $a < n$ некоторым случайным образом.
3. Выполните вероятностный тест на простоту, например, тест Рабина. Если n не выдерживает тестирования, отбросьте данное значение и перейдите к п. 1.
4. Если n выдерживает достаточное число повторных тестов, примите данное значение n как подходящее, в противном случае перейдите к п. 1.

Это процедура несколько утомительна. Однако не забывайте о том, что этот процесс выполняется относительно нечасто — только тогда, когда требуется новая пара (KU, KR).

При этом неплохо знать, как много чисел, скорее всего, окажутся отброшенными прежде, чем обнаружится простое число. Одна из теорем теории чисел, известная под названием теоремы о простых числах, утверждает, что простые числа около N распределяются в среднем по одному на каждые $\ln(N)$ целых чисел. Таким образом, в среднем придется протестировать порядка $\ln(N)$ целых чисел, прежде чем найдется простое число. В действительности из-за того, что все четные целые числа можно отбросить сразу же, истинный порядок задается число $\ln(N) / 2$. Например, если искать простые числа в области величин порядка 2^{200}, то для нахождения простого числа потребуется около $\ln(2^{200}) / 2 = 70$ попыток.

После определения простых чисел p и q процесс вычисления ключей завершается выбором значения e и вычислением d или, наоборот, выбором значения d и вычислением e.

В первом случае необходимо сначала выбрать такое e, чтобы $(\varphi(n), e) = 1$, а потом вычислить $d \equiv e^{-1} \bmod \varphi(n)$. В настоящее время есть алгоритм, который в одно и то же время вычисляет наибольший общий делитель двух целых чисел, и если наибольший общий делитель оказывается равным 1, определяет обратное для одного из целых чисел по модулю другого (мультипликативное обратное). Этот алгоритм называется обобщенным алгоритмом Евклида (см. гл. 9). В процедуру алгоритма входит генерирование случайных чисел и сравнение их с $\varphi(n)$ до тех пор, пока не будет найдено число, взаимно простое с $\varphi(n)$. При этом оказывается, что вероятность того, что два выбранных случайно числа окажутся взаимно простыми, равна примерно 0,6 — это значит, что для нахождения подходящего целого значения понадобится всего несколько проверок.

10.3. КРИПТОАНАЛИЗ АЛГОРИТМА RSA

Тремя возможными подходами к криптоанализу алгоритма RSA являются следующие:

1. *Простой перебор.* Предполагает проверку всех возможных личных ключей.

2. *Математический анализ.* Существует несколько подходов такого рода, но все они по сути эквивалентны нахождению множителей произведения двух простых чисел.

3. *Анализ временных затрат.* Опирается на анализ времени выполнения алгоритма дешифрования.

Защита против простого перебора в случае RSA остается той же, что и для всех других криптосистем, — использование большого пространства ключей. С этой точки зрения, чем больше битов e и d, тем лучше. Однако из-за сложности вычислений как при генерировании ключей, так и при шифровании / дешифровании чем большим оказывается размер ключа, тем медленнее работает система.

Рассмотрим возможности криптоанализа алгоритма RSA при использовании математического подхода и подхода на основе анализа временных затрат.

Проблема разложения на множители. Можно выделить следующие три математически различных подхода к криптоанализу RSA.

1. Разложение n на два его простых множителя. Это позволит вычислить $\varphi(n) = (p - 1)(q - 1)$, на основании чего можно будет определить

$$d = e^{-1} \pmod{\varphi(n)}.$$

2. Определение непосредственно $\varphi(n)$ без того чтобы сначала определять p и q. Это также позволит определить $d = e^{-1} \pmod{\varphi(n)}$.

3. Определение непосредственно d без того чтобы сначала определять $\varphi(n)$.

В большинстве случаев обсуждение вопросов криптоанализа шифра RSA касается задачи разложения значения n на два его простых множителя.

Задача определения $\varphi(n)$ по данному n оказывается эквивалентной задаче разложения n на множители.

Для известных сегодня алгоритмов проблема определения d по данным e и n оказывается требующей, по крайней мере, таких же затрат времени, как и проблема разложения на множители.

Следовательно, затраты на решение задачи разложения на множители можно использовать в качестве эталона при оценке степени защищенности RSA.

Для больших n с большими простыми множителями разложение на множители является серьезной проблемой, но не настолько, как требуется для обеспечения высокой криптостойкости алгоритма RSA.

Поразительной иллюстрацией этого может служить следующая история. В 1977 г. три изобретателя алгоритма RSA отважились предложить читателям популярного журнала «Scientific American» раскрыть шифрованное сообщение, которое они разместили в разделе «Математические игры». За расшифровку этого сообщения они предложили награду в 100 долл. По их оценкам, задача не могла быть решена ранее чем приблизительно через 40 квадриллионов лет. Но в апреле 1994 г. группа пользователей Internet потребовала выплаты призовой суммы после всего восьми месяцев совместной работы в Internet. В предложенной задаче использовался открытый ключ длиной в 129 десятичных знаков (длина n), что равно примерно 428 битам. И точно так, как и для DES, RSA Laboratories объяви-

ли конкурсы для шифров с длинами ключей в 100, 110, 120 и больше знаков. Последней из решенных задач является задача для RSA-130, в которой длина ключа равна 130 десятичным знакам. Имеющиеся на сегодня результаты показаны в табл. 10.2. Единицей меры сложности задачи в данном случае является MIPS-год — объем работы, выполняемой в течение года процессором, осуществляющим обработку одного миллиона команд в секунду. Так, машина на базе Pentium с частотой 200 МГц показывает скорость около 50 MIPS.

Таблица 10.2

Прогресс в решении проблемы разложения на множители [34]

Число десятичных знаков	*Приблизительное-число битов*	*Дата решения*	*Требуемое число MIPS-лет*	*Использованный алгоритм*
100	332	Апрель 1991 г.	7	Квадратичное решето
110	365	Апрель 1992 г.	75	Квадратичное решето
120	398	Июнь 1993 г.	830	Квадратичное решето
129	428	Апрель 1994 г.	5000	Квадратичное решето
130	431	Апрель 1996 г.	500	Решето в поле чисел общего вида

Из табл. 10.2 вытекает удивительный факт, касающийся важности используемого метода. Вплоть до недавнего времени для разложения на множители применялся подход, известный под названием метода квадратичного решета. Для криптоанализа RSA-130 использовался новый алгоритм, называемый методом решета в поле чисел общего вида, что позволило сократить объем вычислительных затрат почти на 10% по сравнению с теми, что требовались ранее для анализа RSA-129.

Угроза ключам большой длины здесь двойная: непрерывный рост вычислительной мощи современных компьютеров и непрерывное усовершенствование алгоритмов разложения на множители. Так, применение совершенно нового алгоритма привело к существенному сокращению времени на решение задачи. Можно ожидать дальнейшего усовершенствования метода решета в поле обобщенных чисел, не исключается также возможность появления более совершенных алгоритмов. Например, имеющийся уже сегодня родственный алгоритм, использующий метод решета в поле чисел специального вида, позволяет раскладывать на множители числа специального вида значительно быстрее, чем метод решета в поле чисел общего вида. Логично ожидать появления результатов, которые дадут возможность увеличить скорость выполнения разложения на множители в общем случае до показателей, которые будут не ниже показателей метода решета в поле чисел специального вида.

Поэтому приходится быть весьма осторожным в выборе длины ключа RSA. В перспективе кажется разумным выбор длины ключа в диапазоне от 1024 до 2048 битов.

В дополнение к проблеме выбора размера n исследователями был обнаружен ряд других ограничений. Чтобы избежать выбора значений n, которые могут быть разложены на множители с меньшими усилиями, изобретатели алгоритма предлагают следующие ограничения относительно p и q.

1. Значения p и q должны различаться по длине всего на несколько разрядов. Например, и p, и q должны попадать в диапазон от 10^{75} до 10^{100}.
2. Как $(p-1)$, так и $(q-1)$ должны содержать в своих разложениях достаточно большой простой множитель.
3. $((p-1), (q-1))$ должен быть достаточно малым.
4. Показано, что если $e < n$ и $d < \frac{1}{4} n$, то d можно определить достаточно легко.

Анализ временных затрат. Для доказательства того, что защита криптографического алгоритма оказывается непростой задачей, как нельзя лучше подойдет описание метода анализа временных затрат.

В данном случае противник получает возможность определить личный ключ, анализируя затраты времени, которые требуются компьютеру для расшифровки сообщений. Анализ временных затрат можно использовать не только в случае RSA, но и с другими системами шифрования с открытым ключом. Атаки такого рода оказываются опасными по двум причинам:

– они могут вестись с самых неожиданных направлений и
– они могут предусматривать анализ только шифрованного текста.

Анализ временных затрат отчасти аналогичен подходу взломщика, отгадывающего комбинацию замка сейфа, наблюдая за тем, сколько времени требуется владельцу для того, чтобы повернуть устройство замка от цифры к цифре.

Суть этого подхода можно объяснить на примере алгоритма возведения в степень в арифметике классов вычетов при двоичном разложении показателя степени x (см. выше схему Горнера) – в данном алгоритме возведение в степень выполняется бит за битом, с одним

умножением на каждой итерации и дополнительным умножением, выполняемым для каждого бита, равного 1.

Предположим, что система шифрования использует функцию умножения в классах вычетов, которая выполняется очень быстро почти во всех случаях, кроме небольшого их числа, когда умножение занимает значительно больше времени. Анализ проводится бит за битом, начиная с первого слева бита. Предположим, что первые j битов уже известны. (Чтобы получить значение показателя степени, следует начать с $j = 0$ и повторять процедуру анализа до тех пор, пока не станут известными все биты показателя). Для имеющегося шифрованного текста противник может выполнить первые j итераций. На следующем этапе выполнение операции зависит от значения неизвестного бита показателя степени. Если этот бит равен 1, то выполняется оператор умножения уже имеющегося произведения d на a по mod n. Для нескольких значений a и d умножение выполняется чрезвычайно медленно, и противник знает, для каких именно. Поэтому если время выполнения алгоритма дешифрования всегда заметно увеличивается, когда эта конкретная итерация выполняется при значении бита, равном 1, то, выявив соответствующую задержку, можно сделать вывод о том, что данный бит равен 1. Если же для целой серии наблюдений алгоритм всегда выполняется достаточно быстро, то этот бит должен быть равным 0.

На практике реализации возведения в степень в классах вычетов не имеют столь резко выраженных вариаций во времени, чтобы продолжительность выполнения одной итерации могла превысить среднее время выполнения всего алгоритма в целом. Но имеющие место в реальности вариации все же достаточны для того, чтобы их анализ оказался практически полезным.

Хотя анализ временных затрат и оказывается серьезной угрозой, против него используются простые контрмеры, например, следующие:

1. *Постоянное время выполнения операции возведения в степень.* Изменение алгоритма таким образом, чтобы все возведения в степень занимали одно и то же время от начала выполнения до возврата результата. Сделать это просто, но при этом увеличивается общее время выполнения алгоритма.

2. *Случайные задержки.* Меньшее влияние на общее время выполнения вызывает добавление в алгоритм возведения в степень случайных задержек, что уменьшает пользу от анализа временных затрат. Но при этом если защищающаяся сторона добавит недостаточно помех, противник будет все еще в состоянии с помощью дополнительных измерений провести криптоанализ и в условиях влияния случайных задержек.

3. *Маскировка.* Умножение шифрованного текста на случайное число, перед тем как выполнять возведение в степень. Это не позволит противнику узнать, какие биты шифрованного текста обрабатывались в компьютере и, таким образом, не даст ему возможности провести поразрядный анализ, который является существенной частью подхода, основанного на анализе временных затрат.

В некоторых продуктах RSA Data Security предполагается возможность использования функции маскировки. В них операция $\mathbf{M} = \mathbf{C}^d (\mathrm{mod} n)$ с личным ключом выполняется следующим образом:

1. Генерируется секретное случайное число r в диапазоне от 0 до $n - 1$.

2. Вычисляется $\mathbf{C}' = \mathbf{C}^{re} (\mathrm{mod}\ n)$, где e является открытым значением показателя степени.

3. Вычисляется $\mathbf{M}' = (\mathbf{C}')^d (\mathrm{mod}\ n)$ для обычной реализации RSA.

4. Вычисляется $\mathbf{M} = \mathbf{M}' r^{-1} (\mathrm{mod}\ n)$. В этом равенстве r^{-1} представляет собой мультипликативное обратное значение r (mod n).Можно показать, что соответствующий результат будет корректным, если заметить, что

$$r^{ed} (\mathrm{mod}\ n) = r (\mathrm{mod}\ n).$$

RSAData Security утверждает, что при использовании функции маскировки общая производительность снижается на величину от 2 до 10%.

Аппаратные реализации RSA. Шифрование RSA выполняется многими микросхемами. Частичный список доступных в настоящее время микросхем RSA приведен в табл. 10.3.

Аппаратно RSA примерно в 1000 раз медленнее DES.

Скорость работы самой быстрой СБИС-реализации RSA с 512-битовым модулем — 64 килобита в секунду. Существуют также микросхемы, которые выполняют 1024-битовое шифрование RSA. Производители также применяют RSA в интеллектуальных карточках, но эти реализации медленнее.

Программно DES примерно в 100 раз быстрее RSA.

Эти числа могут незначительно измениться при изменении технологии, но RSA никогда не достигнет скорости симметричных алгоритмов. В табл. 10.4 приведены примеры скоростей программного шифрования RSA.

Шифрование RSA выполняется намного быстрее, если правильно выбрать значение e. Тремя наиболее частыми вариантами являются 3, 17 и $65537 (2^{16}+1)$. (Двоичное представление 65 537 содержит только две единицы, поэтому для возведения в степень нужно выполнить только 17 умножений). Не существует никаких проблем безопасности, связанных с использованием в качестве e любого из этих трех значений (при условии, что сообщения дополняются случайными числами), даже если одно и то же значение e используется целой группой пользователей.

Таблица 10.3

Существующие микросхемы RSA

Компания	*Тактовая частота,МГц*	*Скорость передачи в Бодах на 512 бит*	*Тактовые циклы для шифрования 512бит, М*	*Технология*	*Битов на микросхему*	*Число транзисторов*
Alpha Techn,	25	13K	0,98	2 микрона	1024	180 000
AT&T	15	19K	0,4	1,5 микрона	298	100 000
British Telecom	10	5,1K	1	2,5 микрона	256	----
Business Sim, Ltd,	5	3,8K	0,67	Вентильная матрица	32	----
Calmos-SystInc	20	2,8K	0,36	2 микрона	593	95 000
CNET	25	5,3K	2,3	1 микрон	1024	100 000
Cryptech	14	17K,	0,4	Вентильная матрица	120	33 000
Cylink	30	6,8K	1,2	1,5 микрона	1024	150 000
GEC Marconi	25	10,2K	0,67	1,4 микрона	512	160 000
Pijnenburg	25	50K	0,256	1 микрон	1024	400 000
Sandia	8	10K	0,4	2 микрона	272	86 000
Siemens	5	8,5K	0,03	1 микрон	512	60 000

Таблица 10.4

Скорости RSA для различных длин модулей при 8-битовом открытом ключе, *с*

Операция	*512 битов*	*768 битов*	*1024 бита*
Шифрование	0,03	0,05	0,08
Дешифрирование	0,16	0,48	0,93
Подпись	0,16	0,52	0,97
Проверка	0,02	0,07	0,08

ГЛАВА 11. УПРАВЛЕНИЕ КЛЮЧАМИ В АСИММЕТРИЧНЫХ КРИПТОСИСТЕМАХ

11.1. РАСПРЕДЕЛЕНИЕ ОТКРЫТЫХ КЛЮЧЕЙ

Одной из главных сфер применения схемы шифрования с открытым ключом является решение проблемы распределения ключей. Имеются две совершенно различные области использования шифрования с открытым ключом в этой сфере:

- распределение открытых ключей;
- использование шифрования с открытым ключом для распределения секретных ключей.

Рассмотрим каждую из этих областей использования данной схемы шифрования по порядку.

Для распределения открытых ключей было предложено несколько методов. Фактически их можно сгруппировать в следующие общие классы:

- публичное объявление;
- публично доступный каталог;
- авторитетный источник открытых ключей;
- сертификаты открытых ключей.

Публичное объявление открытых ключей. Одной из основных составляющих шифрования с открытым ключом является то, что открытый ключ общедоступный. Таким образом, при наличии широко используемого алгоритма (например, RSA) любая участвующая в обмене данными сторона может предоставить свой открытый ключ любой другой стороне или передать ключ по средствам коммуникаций для всех вообще (рис. 11.1). Например, из-за возрастающей популярности PGP, в которой используется RSA, многие пользователи PGP приняли практику присоединения своих открытых ключей к сообщениям, которые они посылают в открытые форумы, например, в группы новостей USENET или списки рассылки Internet.

Этот подход удобен, но он имеет одно слабое место: такое публичное объявление может написать кто угодно. Это значит, что кто-то может представиться пользователем **A** и послать открытый ключ другому пользователю сети или предложить такой открытый ключ для всеобщего пользования. Пока пользователь **A** откроет подлог и предупредит других пользователей, фальсификатор сможет прочитать все шифрованные сообщения, пришедшие за это время для **A**, и сможет использовать фальсифицированные ключи для аутентификации.

Авторитетный источник открытых ключей. Лучшая защита распределения открытых ключей может быть достигнута путем бо-

лее строгого контроля за распределением открытых ключей из каталога. Типичный сценарий представлен схемой на рис. 11.2.

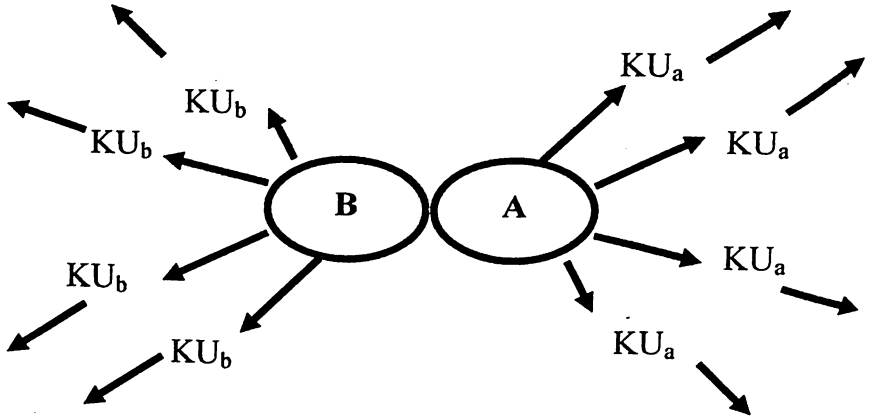


Рис. 11.1. Неконтролируемое распределение открытых ключей [34]

Сценарий предполагает наличие некоторого центрального объекта, уполномоченного поддерживать динамический каталог открытых ключей всех участников обмена данными. Кроме того, каждому из участников известен открытый ключ центра, но только центр знает соответствующий личный ключ. При этом выполняются следующие действия (их номера приведены в соответствии с номерами на рис. 11.2):

1. Инициатор **A** посылает сообщение с меткой даты/времени авторитетному источнику открытых ключей с запросом о текущем открытом ключе участника **B**.

2. Авторитетный источник отвечает сообщением, которое шифруется с использованием личного ключа авторитетного источника KR_{auth}. Это сообщение инициатор **A** может дешифровать, используя открытый ключ авторитетного источника. Поэтому отправитель **A** может быть уверенным в том, что сообщение исходит от авторитетного источника. Это сообщение должно включать следующее:

- открытый ключ KU_b участника **B**, который участник **A** может использовать для шифрования сообщений, предназначенных для получателя **B**;
- оригинальный запрос, чтобы сторона **A** имела возможность сопоставить ответ с ранее отправленным запросом и убедиться, что запрос не был изменен на пути к авторитетному источнику;
- оригинальную метку даты / времени, чтобы отправитель **A** мог удостовериться, что это сообщение не является одним из старых сообщений от авторитетного источника, содержащим ключ, отличный от текущего открытого ключа адресата **B**.

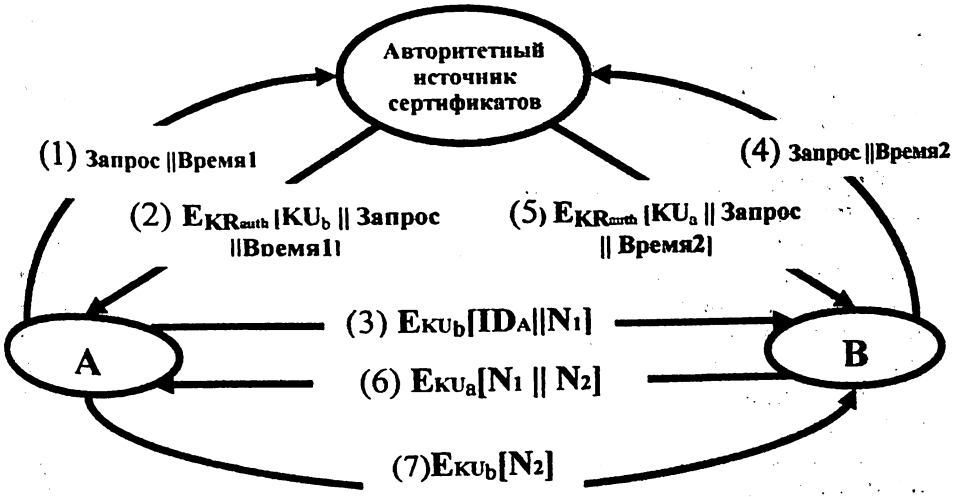


Рис. 11.2. Распределение открытых ключей с использованием авторитетного источника открытых ключей [34]

1. Инициатор **A** сохраняет открытый ключ участника **B** и использует его для шифрования сообщения, направляемого получателю **B** и содержащего идентификатор отправителя **A** (ID_A) и оказию (N_1), которая выступает в качестве уникальной метки данного сообщения.

2. Респондент **B** получает открытый ключ **A** от авторитетного источника точно таким же способом, каким отправитель **A** получил открытый ключ получателя **B**. К этому моменту открытые ключи оказываются доставленными участникам **A** и **B** вышеописанной защищенной процедурой, так что теперь они могут начать постепенный обмен данными. Но перед этим желательно выполнить два следующих дополнительных действия.

3. Респондент **B** посылает сообщение инициатору **A**, шифрованное с помощью **KUa** и содержащее оказию отправителя **A**(N_1), а также новую оказию, сформированную участником **B**(N_2). Ввиду того что только он мог дешифровать сообщение (3), присутствие N_1 в сообщении (6) убеждает участника **A** в том, что отправителем полученного сообщения был **B**.

4. Инициатор **A** возвращает N_2, шифрованное с помощью открытого ключа KU_b участника **B**, чтобы тот мог убедиться в том, что отправителем ответа является **A**.

Таким образом, в общей сумме потребуется семь сообщений. Однако отсылать первые сообщения требуется нечасто, так как обе

стороны могут сохранять открытые ключи друг друга для дальнейшего пользования, что обычно называется *кэшированием*.

Периодически пользователь должен запрашивать свежие экземпляры открытых ключей своих адресатов, чтобы иметь гарантированную стойкость обмена данными.

Сертификаты открытых ключей. Сценарий, показанный на рис. 11.2, привлекателен, но и он имеет некоторые недостатки. Авторитетный источник открытых ключей является узким местом, поскольку пользователь должен апеллировать к авторитетному источнику при необходимости получить открытый ключ для каждого нового адресата, с которым этот пользователь намерен вести переписку. Кроме того, как и в предыдущем случае, каталог имен и открытых ключей, поддерживаемый авторитетным источником, остается уязвимым по отношению к вмешательству с неблагоприятными намерениями.

Альтернативный подход основан на сертификатах, которые могут использоваться участниками обмена ключами без контакта с авторитетным источником открытых ключей и должны обеспечивать способ обмена такой же надежный, как и способ получения ключей непосредственно от авторитетного источника открытых ключей.

Каждый сертификат содержит открытый ключ и другую информацию, создается авторитетным источником сертификатов и выдается участнику вместе с соответствующим личным ключом. Один участник передает информацию о своем ключе другому с помощью передачи своего сертификата. Другие участники могут проверить, что сертификат был создан авторитетным источником. Можно сформулировать следующие требования к этой схеме.

1. Любой участник должен иметь возможность прочитать сертификат, чтобы определить имя и открытый ключ владельца сертификата.
2. Любой участник должен иметь возможность проверить, что сертификат исходит из авторитетного источника сертификатов и не является подделкой.
3. Только авторитетный источник сертификатов должен иметь возможность создавать и изменять сертификаты.
4. Любой участник должен иметь возможность проверить срок действия сертификата.

Схема использования сертификатов показана на рис. 11.3.

Каждый участник обращается к авторитетному источнику сертификатов, предоставляя открытый ключ и запрашивая для него сертификат. Запрос должен предполагать либо личное обращение, либо некоторую защищенную форму связи. Для участника **A** авторитетный источник обеспечивает сертификат вида

$$C_A = E_{KRauth}[T, ID_A, KU_a],$$

где KR_{auth} обозначает личный ключ, используемый авторитетным источником. Теперь участник **A** может переслать этот сертификат любому другому участнику, который прочитывает и проверяет сертификат:

$$D_{KUauth}[C_A] = D_{KUauth}[E_{KRauth}[T, ID_A, KU_a]] = (T, ID_A, KU_a).$$

Получатель использует открытый ключ авторитетного источника сертификатов KU_{auth}, чтобы дешифровать сертификат. Ввиду того что сертификат можно прочитать только с помощью открытого ключа авторитетного источника сертификатов, есть гарантия того, что сертификат пришел именно от авторитетного источника сертификатов.

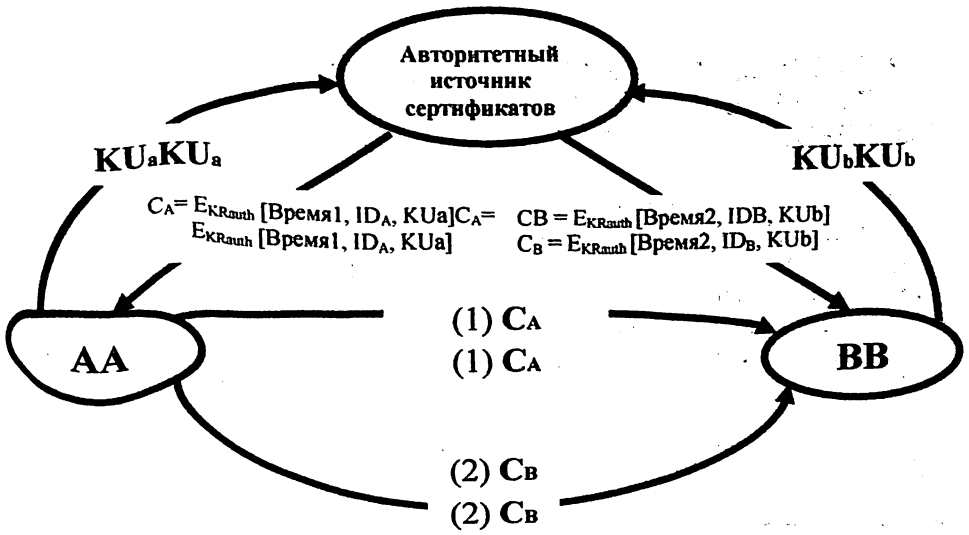


Рис. 11.3. Обмен сертификатами открытых ключей [34]

Элементы ID_A и KU_a сообщают получателю имя и открытый ключ владельца сертификата.

Наконец, метка даты / времени **T** определяет срок действия сертификата. Метка даты / времени должна быть защищена от следующей последовательности действий.

1. Противник узнает личный ключ **A**. По этой причине **A** генерирует вторую пару ключей (личный и открытый) и обращается к авторитетному источнику сертификатов за новым сертификатом.

2. Тем временем, противник воспроизводит сообщение со старым сертификатом и отсылает его **B**. Если **B** будет шифровать сообщения, используя старый скомпрометированный открытый ключ, противник сможет прочитать эти сообщения.

В этом смысле компрометация личного ключа сравнима с потерей кредитной карточки. Владелец объявляет номер кредитной карточки недействительным, но ситуация остается потенциально рискованной до тех пор, пока все возможные системы не будут информированы о том, что старая кредитная карточка уже недействительна.

Таким образом, метка даты/времени является чем-то вроде даты истечения срока действия. Если сертификат оказывается выданным достаточно давно, предполагается, что срок его действия истек.

11.2. РАСПРЕДЕЛЕНИЕ СЕКРЕТНЫХ КЛЮЧЕЙ С ИСПОЛЬЗОВАНИЕМ КРИПТОСИСТЕМЫ С ОТКРЫТЫМ КЛЮЧОМ

После того как открытые ключи были распределены и стали доступными, становится реальной организация защищенной связи, не допускающей возможность перехвата или искажения сообщений, или того и другого вместе. Однако некоторые пользователи предпочтут использовать шифрование с открытым ключом только в исключительных случаях из-за того, что в условиях применения этого шифрования скорость передачи данных оказывается относительно медленной. Поэтому шифрование с открытым ключом приходится рассматривать скорее как средство распределения секретных ключей, используемых для традиционного шифрования.

11.2.1. ПРОСТОЕ РАСПРЕДЕЛЕНИЕ СЕКРЕТНЫХ КЛЮЧЕЙ

Исключительно простая схема представлена на рис. 11.4.

Если инициатор **A** намерен обменяться данными с пользователем **B**, для этого предполагается следующая процедура.

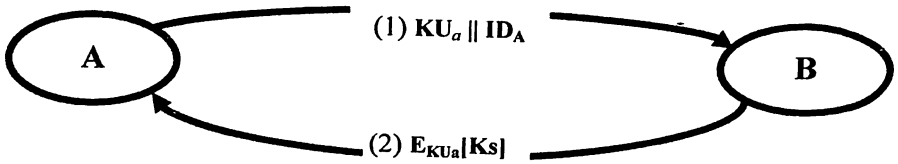


Рис. 11.4. Простое использование шифрования с открытым ключом при выборе сеансового ключа [34]

1. Сторона **A** генерирует пару открытый / личный ключи $\{KU_a, KR_a\}$ и передает стороне **B** сообщение, содержащее KU_a и идентификатор ID_A отправителя **A**.

2. Получатель **B** генерирует секретный ключ K_s и передает этот ключ инициатору сообщения **A** зашифрованным с помощью открытого ключа KU_a инициатора **A**.

3. Пользователь **A** вычисляет $D_{KRa}[E_{KUa}[K_s]]$, чтобы восстановить секретный ключ. Поскольку только пользователь **A** может дешифровать это сообщение, только участники обмена данными **A** и **B** будут знать значение K_s.

4. Участник **A** выбрасывает ключ KR_a, а участник **B** выбрасывает ключ KU_a.

Теперь обе стороны, **A** и **B**, могут использовать связь, защищенную традиционным шифрованием с сеансовым ключом K_s. Несмотря на простоту, этот протокол весьма привлекателен. Никаких ключей не существует перед началом связи и никаких ключей не остается после завершения связи. Поэтому риск компрометации ключей минимален. В то же время связь оказывается защищенной от подслушивания.

Этот протокол уязвим в отношении активных атак. Если противник **E** имеет возможность внедрения в канал связи, то он может скомпрометировать связь, без того чтобы быть обнаруженным, следующим образом.

5. Участник **A** генерирует пару открытый / личный ключи $\{KU_a, KR_a\}$ и передает стороне **B** сообщение, содержащее KU_a и идентификатор ID_A отправителя **A**.

6. Противник **E** перехватывает сообщение, создает собственную пару открытый / личный ключи $\{KU_e, KR_e\}$ и передает адресату **B** сообщение, содержащее $KU_e \| ID_A$.

7. **B** генерирует секретный ключ K_s и передает $E_{KUe}[K_s]$.

8. Противник **E** перехватывает это сообщение и узнает K_s, вычисляя $D_{KRe}[E_{KUe}[K_s]]$.

9. Противник **E** передает участнику **A** сообщение $E_{KUa}[K_s]$.

В результате оба участника, **A** и **B**, будут знать K_s, но не будут подозревать, что K_s также известен противнику **E**. Поэтому стороны **A** и **B** могут начать обмен сообщениями, используя K_s. Противник **E** больше не будет активно вмешиваться в канал связи, а просто будет перехватывать сообщения. Зная K_s, он сможет дешифровать любое сообщение, а участники **A** и **B** даже не будут подозревать о существовании проблемы. Таким образом, этот простой протокол оказывается полезным только в случае, когда единственной возможной угрозой является пассивный перехват сообщений.

11.2.2. РАСПРЕДЕЛЕНИЕ СЕКРЕТНЫХ КЛЮЧЕЙ С ОБЕСПЕЧЕНИЕМ КОНФИДЕНЦИАЛЬНОСТИ И АУТЕНТИФИКАЦИИ

Схема на рис. 11.5 обеспечивает защиту и от активной, и от пассивной форм атаки. В качестве исходных условий предположим, что **A** и **B** уже обменялись открытыми ключами по одной из схем, описанных выше. Далее надлежит выполнить следующие действия.

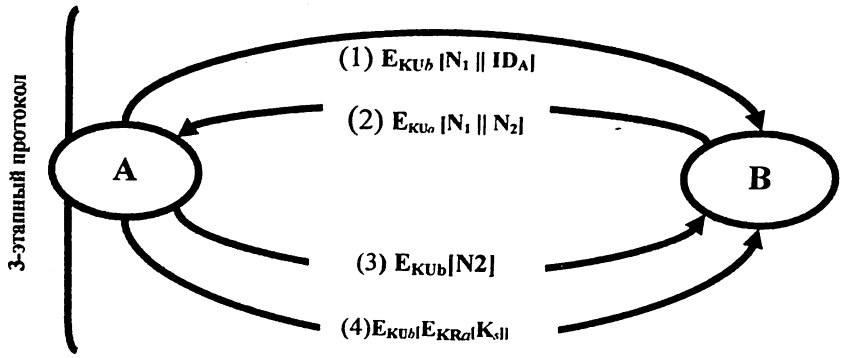


Рис. 11.5. Распределение секретных ключей с помощью шифрования с открытым ключом [34]

1. Сторона **A** использует открытый ключ стороны **B**, чтобы переслать стороне **B** шифрованное сообщение, содержащее идентификатор участника **A** (ID_A) и оказию (N_1), используемую для идентификации данной конкретной транзакции.

2. Пользователь **B** посылает сообщение пользователю **A**, зашифрованное с помощью KU_a и содержащее полученную от него оказию (N_1) и новую оказию (N_2), сгенерированную пользователем **B**. Ввиду того что только участник **B** мог дешифровать сообщение (1), присутствие N_1 в сообщении (2) убеждает участника **A** в том, что респондентом является сторона **B**.

3. Сторона **A** возвращает N_2, шифруя сообщение открытым ключом стороны **B**, чтобы гарантировать то, что его респондентом является сторона **A**.

4. Участник **A** выбирает секретный ключ K_s и посылает участнику **B** сообщение $M = E_{KUb}[E_{KRa}[K_s]]$. Шифрование этого сообщения открытым ключом стороны **B** гарантирует, что только участник **B** сможет прочитать его, а шифрование личным ключом участника **A** — что только участник **A** мог послать его.

5. Сторона **B** вычисляет $D_{KUa}[E_{KRb}[M]]$, чтобы восстановить секретный ключ.

Обратите внимание на то, что первые три действия этой схемы соответствуют последним трем действиям схемы, показанной на рис. 11.2 («трехэтапный протокол рукопожатия»). В результате при обмене секретными ключами эта схема гарантирует как конфиденциальность, так и аутентификацию.

11.2.3. ГИБРИДНАЯ СХЕМА

Еще одну схему использования шифрования с открытым ключом при распределении секретных ключей представляет гибридный подход, применяемый на мэйнфреймах IBM. Эта схема предполагает участие центра распределения ключей (ЦРК), с которым каждый пользователь использует свой главный секретный ключ, и распределение секретных сеансовых ключей, шифруемых главным ключом. Схема шифрования с открытым ключом служит для распределения главных ключей. В основе такого трехуровневого подхода лежит следующая логика.

1. *Скорость выполнения процедуры.* Существует много приложений, особенно — ориентированных на передачу транзакции, где сеансовые ключи должны меняться очень часто. Распределение сеансовых ключей с помощью схемы с открытым ключом могло бы сделать производительность системы слишком низкой из-за относительно высоких требований к вычислительным ресурсам при шифровании и дешифровании по такой схеме. В случае трехуровневой иерархии шифрование с открытым ключом применяется лишь иногда, чтобы изменить главный ключ, разделяемый пользователем и ЦРК.

2. *Обратная совместимость.* Гибридную схему можно легко реализовать в виде расширения уже имеющейся схемы, предполагающей использование ЦРК, с минимальными изменениями предусмотренной процедуры и программного обеспечения.

Добавление уровня шифрования с открытым ключом обеспечивает защищенное и эффективное средство распределения главных ключей. Это является преимуществом в конфигурации, когда один ЦРК обслуживает большое число пользователей, находящихся на значительном расстоянии друг от друга.

11.2.4.ОБМЕН КЛЮЧАМИ ПО СХЕМЕ ДИФФИ – ХЕЛЛМАНА

Первый из опубликованных алгоритмов на основе открытых ключей появился в работе Диффи и Хеллмана, в которой было определено само понятие криптографии с открытым ключом. Обычно

этот алгоритм называют «обменом ключами по схеме Диффи – Хеллмана». Такая технология обмена ключами реализована в целом ряде коммерческих продуктов.

Цель схемы — обеспечить двум пользователям защищенную возможность сообщить друг другу ключ, чтобы они могли прибегнуть к ней для шифрования последующих сообщений. Сам по себе алгоритм ограничивается процедурой обмена ключами.

Эффективность алгоритма Диффи – Хеллмана опирается на трудность вычисления дискретных логарифмов.

Формально дискретный логарифм можно определить следующим образом (см. гл. 9). Сначала определяется первообразный корень простого числа p как число, степени которого порождают все целые числа от 1 до $p-1$. Это значит, что если a является первообразным корнем простого числа p, то все числа

$$a \bmod p, (a^2) \bmod p, \ldots, (a^{p-1}) \bmod p$$

должны быть разными и представлять все целые числа от 1 до $p-1$ в некоторой перестановке.

Для любого целого числа b и любого первообразного корня a простого числа p однозначно определяется показатель степени i, при котором

$$b = (a^i) \bmod p, \text{ где } 0 \le i \le (p-1).$$

Этот показатель степени обычно называется дискретным логарифмом, или индексом b по основанию a, рассматриваемым по модулю p.

Это значение записывается в форме $\mathrm{ind}_{a,p}(b)$.

Теперь опишем обмен ключами по схеме Диффи – Хеллмана, иллюстрацией которой служит рис. 11.6.

В этой схеме имеются два открытых для всех числа: простое число q и целое число α, являющееся первообразным корнем q. Предположим, пользователи **A** и **B** намерены обменяться ключами. Пользователь **A** выбирает целое число $X_A < q$ и вычисляет

$$Y_A = \alpha^{X_A} \bmod q$$

Точно так же пользователь **B** независимо выбирает случайное целое число $X_B < q$ и вычисляет

$$Y_B = \alpha^{X_B} \bmod q$$

Каждая сторона сохраняет значение X в тайне и делает значение Y свободно доступным другой стороне. Пользователь **A** вычисляет ключ по формуле

$$K = Y_B^{X_A} \bmod q$$

а пользователь **B** — по формуле

$$K = Y_A^{X_B} \bmod q.$$

Эти две формулы вычисления дают одинаковые результаты, как показано далее.

$$K = Y_B^{X_A} \bmod q = (((\alpha^{X_B}) \bmod q)^{X_A}) \bmod q =$$
$$= ((\alpha^{X_B})^{X_A}) \bmod q = (\alpha^{X_B X_A}) \bmod q = ((\alpha^{X_A})^{X_B}) \bmod q =$$
$$= (((\alpha^{X_A}) \bmod q)^{X_B}) \bmod q = (Y_A^{X_B}) \bmod q$$

Итак, обе стороны обменялись секретным ключом. А поскольку при этом X_A и X_B были только в личном использовании и поэтому сохранились в тайне, противнику придется работать только с q, α, Y_A и Y_B. Таким образом, ему придется вычислять дискретный логарифм, чтобы определить ключ. Например, чтобы определить ключ пользователя **B**, противнику нужно вычислить

$$X_B = \mathrm{ind}_{a,q}(Y_B).$$

После этого он сможет вычислить ключ K точно так же, как это делает пользователь **B**.

Защищенность обмена ключами по схеме Диффи – Хеллмана опирается фактически на то, что в то время как степени по модулю некоторого простого числа вычисляются относительно легко, вычислять дискретные логарифмы оказывается очень трудно. Для больших простых чисел последнее считается задачей практически неразрешимой.

Пример. Обмен ключами строится на использовании простого числа $q = 97$ и его первообразного корня $\alpha = 5$. Пользователи **A** и **B** выбирают секретные ключи $X_A = 36$ и $X_B = 58$ соответственно. Каждый вычисляет свой открытый ключ:

$$Y_A = 5^{36} = 50 \bmod 97,$$
$$Y_B = 5^{58} = 44 \bmod 97.$$

После того как пользователи обменялись открытыми ключами, каждый из них может вычислить общий секретный ключ:

$$K = Y_B^{X_A} \bmod 97 = 44^{36} \bmod 97 = 75 \bmod 97,$$
$$K = Y_A^{X_B} \bmod 97 = 50^{58} \bmod 97 = 75 \bmod 97.$$

Имея {50, 44}, противнику не удастся с легкостью вычислить 75.

Глобальные открытые элементы

q – простое число

α – первообразный корень q, $\alpha < q$

Вычисление ключа пользователем A

Выбор секретного $X_A X_A < q$

Вычисление $Y_A Y_A = (\alpha^{X_A}) \bmod q$

Вычисление ключа пользователем B

Выбор секретного $X_B X_B < q$

Вычисление $Y_B Y_B = (\alpha^{X_B}) \bmod q$

Вычисление секретного ключа пользователем A

$$K = (Y_B^{X_A}) \bmod q$$

Вычисление секретного ключа пользователем B

$$K = (Y_A^{X_B}) \bmod q$$

Рис. 11.6. Алгоритм обмена ключами по схеме Диффи – Хеллмана [34]

На рис. 11.7 представлен простой протокол, в котором применяются вычисления в соответствии со схемой Диффи – Хеллмана. Предположим, что пользователь **A** собирается установить связь с пользователем **B** и использовать секретный ключ, чтобы шифровать сообщения, передаваемые с помощью такой связи. Пользователь **A** может генерировать однозначное секретное значение X_A, вычислить значение Y_A и отослать последнее пользователю **B**. В ответ пользователь **B** генерирует секретное значение X_B, вычисляет Y_B и Y_B посылает пользователю **A**. Оба пользователя могут теперь вычислить общий ключ. Необходимые открытые значения q и α должны быть известны заранее. Пользователь **A** может также выбрать эти значения на свое усмотрение и включить их в первое сообщение.

Для примера другой возможности использования алгоритма Диффи – Хеллмана рассмотрим некоторую группу пользователей (например, всех пользователей локальной сети) и предположим, что каждый из этих пользователей должен сгенерировать секретное значение X_A для долгосрочного применения и вычислить открытое значение Y_A. Все полученные таким образом открытые значения вместе с глобальными открытыми значениями q и α сохраняются централизованно в некотором каталоге.

В любой момент пользователь **B** может получить доступ к открытому значению пользователя **A**, вычислить секретный ключ и использовать его для пересылки шифрованного сообщения пользователю **A**. Если централизованно хранящийся каталог надежен, то эта форма коммуникации обеспечивает как конфиденциальность, так и определенную степень аутентификации. Поскольку только пользователи **A** и **B** могут определить ключ, другой пользователь не может прочитать сообщение (конфиденциальность). Получатель **A** знает, что только пользователь **B** мог создать сообщение, используя этот ключ (аутентификация).

Однако такая схема не защищена от атак на основе воспроизведения сообщений. Для защиты, как правило, используется 3-этапный протокол рукопожатия, рассмотренный в п. 11.2.2.

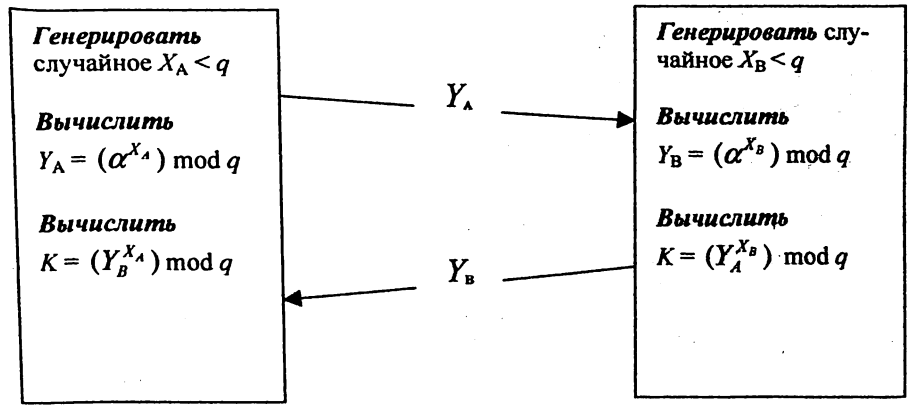


Рис. 11.7. Обмен ключами по схеме Диффи – Хеллмана [34]

ГЛАВА 12. ХЭШ-ФУНКЦИИ

12.1. ТРЕБОВАНИЯ К ХЭШ-ФУНКЦИЯМ

Хэш-функцией называется односторонняя функция, предназначенная для получения дайджеста или «отпечатков пальцев» файла, сообщения или некоторого блока данных.

Хэш-код создается функцией H:

$$h = H(\mathbf{M}),$$

где **M** является сообщением произвольной длины, а h является хэш-кодом фиксированной длины.

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).

Рассмотрим требования, которым должна соответствовать хэш-функция, для того чтобы она могла использоваться в качестве аутентификатора сообщения.

Хэш-функция H, которая используется для аутентификации сообщений, должна обладать следующими свойствами:

1. Хэш-функция H должна применяться к блоку данных любой длины.
2. Хэш-функция H создает выход фиксированной длины.
3. $H(\mathbf{M})$ относительно легко (за полиномиальное время) вычисляется для любого значения **M**.
4. Для любого данного значения хэш-кода h вычислительно невозможно найти **M** такое, что $H(\mathbf{M}) = h$.
5. Для любого данного x вычислительно невозможно найти такое $y \neq x$, что $H(y) = H(x)$.

Такое свойство иногда называют слабой сопротивляемостью коллизиям.

Коллизией называется совпадение дайджестов для различных данных.

В принципе коллизии возможны (так как мощность множества дайджестов меньше мощности множества хэшируемых данных), однако, исходя из определения хэш-функции, специально организовать коллизию за приемлемое время невозможно.

6. Вычислительно невозможно найти произвольную пару (x, y) такую, что $H(y) = H(x)$. Это свойство называют сильной сопротивляемостью коллизиям.

Первые три свойства требуют, чтобы хэш-функция создавала хэш-код для любого сообщения.

Четвертое свойство определяет требование односторонности хэш-функции: легко создать хэш-код по данному сообщению, но невозможно восстановить сообщение по данному хэш-коду.

Это свойство важно, если аутентификация с использованием хэш-функции включает секретное значение S_{AB}. Само секретное значение может не посылаться, тем не менее, если хэш-функция не является односторонней, противник может легко раскрыть секретное значение S_{AB} следующим образом.

При перехвате передачи атакующий получает сообщение **M** и хэш-код $\mathbf{C} = H(S_{AB}||\mathbf{M})$. Если атакующий может инвертировать хэш-функцию, то, следовательно, он может получить $S_{AB}||\mathbf{M} = H^{-1}(\mathbf{C})$. Так как атакующий теперь знает и **M**, и $S_{AB}||\mathbf{M}$, получить S_{AB} совсем просто.

Пятое свойство гарантирует, что не удастся найти другое сообщение, дающее в результате хэширования то же самое значение, что и данное сообщение.

Это предотвращает возможность фальсификации сообщения в том случае, когда выполняется шифрование хэш-кода. В такой ситуации противник может прочитать сообщение и вычислить соответствующий хэш-код. Однако он не имеет секретного ключа, а потому не может изменить сообщение так, чтобы это не было обнаружено. Если это свойство не выполнено, противник может действовать по следующей схеме:

– сначала перехватить сообщение вместе с присоединенным к нему шифрованным хэш-кодом,

– затем вычислить нешифрованный хэш-код сообщения,

– наконец, создать альтернативное сообщение с тем же хэш-кодом.

Шестое свойство определяет стойкость функции хэширования к конкретному классу атак, известных под названием атак, построенных на парадоксе «задачи о днях рождения» (рассматривается далее).

12.2. ПРОСТЫЕ ХЭШ-ФУНКЦИИ

Все функции хэширования построены на следующих общих принципах.

Вводимое значение (сообщение, файл и т.д.) рассматривается как последовательность n-битовых блоков. Вводимые данные обрабатываются последовательно блок за блоком, чтобы в результате получить n-битовое значение функции хэширования.

Одной из простейших функций хэширования является связывание всех блоков операцией поразрядного исключающего «ИЛИ» (XOR). Это можно записать в следующем виде:

$$\mathbf{C}_i = b_{i1} \oplus b_{i2} \oplus \ldots \oplus b_{im},$$

где $\mathbf{C}_i$ — i-й бит хэш-кода, $1 \le i \le n$;
m — число n-битовых блоков ввода;
b_{ij} — i-й бит в j-м блоке;
$\oplus$ — операция XOR.

Эта процедура показана в табл. 12.1. Она осуществляет простой побитовый контроль четности и обычно называется продольным контролем четности.

Таблица 12.1

Простая функция хэширования, выполняющая операцию XOR

Блоки	*Бит 1*	*Бит 2*	...	*Бит n*
1	b_{11}	b_{21}	...	b_{n1}
2	b_{12}	b_{22}	...	b_{n2}
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
m	b_{1m}	b_{2m}	...	b_{nm}
Хэш-код	C_1	C_2	...	C_n

Такая процедура достаточно эффективна при контроле целостности данных в случае данных произвольного вида. Тогда любое n-битовое значение функции хэширования оказывается одинаково вероятным. Значит, вероятность того, что при появлении ошибки в данных значение функции хэширования останется прежним, равна 2^{-n}. Если же речь идет о более прогнозируемых форматированных данных, такая функция менее эффективна.

Например, в текстовых файлах с английским текстом старший разряд каждого байта всегда равен нулю, поэтому если использовать 128-битовое значение функции хэширования, то вместо эффективности 2^{128} функция хэширования для этого типа данных покажет эффективность, равную 2^{112}.

Проще всего усовершенствовать такую схему, рассмотрев возможность выполнения однобитового циклического сдвига или поворота значения функции хэширования после завершения обработки каждого очередного блока. Такая процедура состоит из следующих этапов:

1. Начальная инициализация n-битового значения функции хэширования нулевым значением.

2. Последовательная обработка n-битовых блоков данных по следующему правилу:

- выполнение циклического сдвига текущего значения функции хэширования влево на один бит;
- добавление текущего блока к значению функции хэширования с помощью операции XOR.

Эта процедура демонстрирует эффект «рандомизации» вводимых данных и разрушения регулярностей, которые наблюдаются для вводимых данных.

Хотя вторая из вышеупомянутых процедур и обеспечивает хорошую возможность контроля целостности данных, она практически бесполезна для защиты, когда с открытым сообщением передается шифрованный хэш-код. Имея некоторое сообщение, совсем нетрудно создать новое сообщение, которому будет соответствовать тот же самый хэш-код: просто подготовьте любое необходимое альтернативное сообщение и присоедините к нему подходящий n-битовый блок, который вместе с новым сообщением сформирует желаемый хэш-код.

Хотя простое выполнение операции XOR или той же операции с циклическим кодом (RXOR) в случае шифрования только хэш-кода оказывается недостаточным, такая простая функция может быть полезна, когда шифруются и хэш-код, и сообщение.

Но здесь требуется последующее шифрование всего сообщения в режиме сцепления блоков (CBC).

Такая схема может быть описана следующим образом. Имея сообщение, складывающееся из последовательности 64-битовых блоков $X_1, X_2, \ldots, X_n$, сначала следует вычислить хэш-код $\mathbf{C}$, равный результату связывания всех блоков с помощью операции XOR, а затем присоединить полученный хэш-код к концу сообщения в качестве еще одного блока:

$$\mathbf{C} = X_{n+1} = X_1 \oplus X_2 \oplus \ldots \oplus X_n.$$

После этого все сообщение вместе с присоединенным хэш-кодом шифруется в режиме CBC, в результате чего получается шифрованное сообщение $Y_1, Y_2 \ldots Y_{n+1}$.

Существует несколько способов, с помощью которых шифрованный текст такого сообщения можно реорганизовать так, чтобы это не повлияло на хэш-код.

Так, например, если слагаемые уравнения сцепления связываются операцией XOR в любом порядке, хэш-код не меняется при изменения порядка следования блоков шифрованного текста.

Метод сцепления блоков. Целый ряд предложений, касающихся функций хэширования, был основан на использовании техники сцепления шифрованных блоков, но без использования секретного ключа.

Одним из первых таких предложений было предложение Рабина (*Rabin*).

Разделим сообщение **M** на блоки фиксированной длины $\mathbf{M}_1$, $\mathbf{M}_2$, ..., $\mathbf{M}_n$ и используем любую систему традиционного шифрования (например, DES), чтобы вычислить хэш-код **G** следующим образом:

H_0 = начальное значение,
$H_i = E_{Mi}[H_{i-1}]$,
$G = H_n$.

Это похоже на использование режима CBC, но в данном случае нет никакого секретного ключа. Как любой хэш-код, эта схема может подвергаться атакам, в основе которых лежит парадокс задачи о днях рождения, и если алгоритмом шифрования является DES, когда поражается только 64-битовый хэш-код, то система может стать уязвимой.

12.3. ПАРАДОКС ДНЯ РОЖДЕНИЯ И АТАКИ, НА НЕМ ОСНОВАННЫЕ

Так называемый «парадокс дня рождения» состоит в следующем.

Предположим, количество выходных значений хэш-функции **H** равно n. Каким должно быть число k, чтобы для конкретного значения **X** и значений $\mathbf{Y}_1, \ldots, \mathbf{Y}_k$ вероятность того, что хотя бы для одного $\mathbf{Y}_i$ выполнялось равенство

$$\mathbf{H(X)} = \mathbf{H(Y)},$$

была бы больше 0,5?

Для одного **Y** вероятность того, что $\mathbf{H(X)} = \mathbf{H(Y)}$, равна $1 / n$.

Соответственно, вероятность того, что $\mathbf{H(X)} \neq \mathbf{H(Y)}$, равна $1 - 1 / n$.

Если создать k значений, то вероятность того, что ни для одного из них не будет совпадений, равна произведению вероятностей, соответствующих одному значению, т.е. $(1 - 1 / n)^k$.

Следовательно, вероятность, по крайней мере, одного совпадения равна

$$1 - (1 - 1/n)^k.$$

По формуле бинома Ньютона

$$(1 - a)^k = 1 - ka + (k(k-1)/2!)a^2 - \ldots \approx 1 - ka,$$

т.е. $1 - (1 - k/n) = k/n = 0.5$, откуда $k = n / 2$.

Таким образом, для m-битового хэш-кода достаточно выбрать 2^{m-1} сообщений, чтобы вероятность совпадения хэш-кодов была больше 0,5.

Теперь рассмотрим следующую задачу: обозначим $P(n, k)$ вероятность того, что во множестве из k элементов, каждый из которых может принимать n значений, есть хотя бы два с одинаковыми значениями.

Чему должно быть равно k, чтобы $P(n, k)$ была бы больше 0,5?

Число различных способов выбора элементов таким образом, чтобы при этом не было дублей, равно

$$n(n-1) \ldots (n-k+1)n!/(n-k)!.$$

Всего число возможных способов выбора элементов равно n^k.

Вероятность того, что дублей нет, равна $n! / (n-k)!n^k$.

Вероятность того, что есть дубли, соответственно равна

$$1 - n!/(n-k)! \times n^k.$$

$$P(n, k) = 1 - n!/((n-k)! \times n^k) = 1 - (n(n-1) \times \ldots$$
$$\ldots \times (n-k-1)) / n^k = 1 - [(n-1)/n(n-2)/n \times \ldots$$
$$\ldots \times (n-k+1)/n] = 1 - [(1-1/n)(1-2/n) \times \ldots$$
$$\ldots \times (1-(k-1)/n)].$$

Известно, что $1 - x \leq e^{-x}$.

По условию задачи $P(n, k) > 1 - [e^{-1/n} \times e^{-2}/n \times \ldots \times e^{-k}/n]$, т.е.

$$P(n, k) > 1 - e^{-k(k-1)/n}.$$

Следовательно,

$$1/2 = 1 - e^{-k(k-1)/n}, \text{ и } 2 = e^{k(k-1)/n}.$$

Отсюда

$$\ln 2 = k(k-1) / 2n \text{ и } k(k-1) \approx k^2.$$

Окончательно имеем $k = (2n \times \ln 2)^{1/2} = 1{,}17\, n^{1/2} \approx n^{1/2}$.

Таким образом, если хэш-код имеет длину m бит, т.е. принимает 2^m значений, то $k = \sqrt{2m} = 2^{m/2}$

Подобный результат называется «парадоксом дня рождения», потому что в соответствии с приведенными выше рассуждениями для того чтобы вероятность совпадения дней рождения у двух человек была больше 0,5, в группе должно быть всего 23 человека. Этот результат кажется удивительным, возможно, потому, что для каждого отдельного человека в группе вероятность того, что с его днем рождения совпадет день рождения кого-то другого в группе, достаточно мала.

Атаки, основанные на парадоксе дня рождения. Вернемся к рассмотрению свойств хэш-функций. Предположим, что используется 64-битный хэш-код. Можно считать, что это вполне достаточная и, следовательно, безопасная длина для хэш-кода. Например, если зашифрованный хэш-код **C** передается с соответствующим незашифрованным сообщением **M**, то противнику необходимо будет найти **M'** такое, что $H(\mathbf{M}') = H(\mathbf{M})$, чтобы подменить сообщение и обмануть получателя. В среднем противник должен перебрать 2^{63} сообщений, чтобы найти такое, у которого хэш-код равен перехваченному сообщению.

Тем не менее, возможны различного рода атаки, основанные на «парадоксе дня рождения». Возможна следующая стратегия:

1. Противник (атакующий) создает $2^{m/2}$ вариантов сообщения, каждое из которых имеет некоторый определенный смысл. Противник подготавливает такое же количество сообщений, каждое из которых является поддельным и предназначено для замены настоящего сообщения.

Два набора сообщений сравниваются в поисках пары сообщений, имеющих одинаковый хэш-код. Вероятность успеха в соответствии с «парадоксом дня рождения» больше 0,5. Если соответствующая пара не найдена, то создаются дополнительные исходные и поддельные сообщения до тех пор, пока не будет найдена пара.

2. Атакующий предлагает отправителю исходный вариант сообщения для подписи. Эта подпись может быть затем присоединена к поддельному варианту для передачи получателю. Поскольку оба варианта имеют один и тот же хэш-код, будет создана одинаковая подпись. Противник будет уверен в успехе, даже не зная ключа шифрования.

Таким образом, если используется 64-битный хэш-код, то необходимая сложность вычислений составляет порядка 2^{32}.

Генерировать много вариантов сообщения с одинаковым смыслом совсем не трудно. Например, противник может вставить в ряд мест документа последовательности типа «пробел — пробел — возврат на одну позицию» между словами. Тогда вариации можно будет порождать, заменяя указанные последовательности на «пробел-возврат на одну позицию-пробел» в избранных местах. Противник также может просто переписать сообщение другими словами, сохранив смысл.

В заключение отметим, что длина хэш-кода должна быть достаточно большой. Длина, равная 64 битам, в настоящее время не считается безопасной. Предпочтительнее, чтобы длина составляла порядка 100 битов.

12.4. СПОСОБЫ ИСПОЛЬЗОВАНИЯ ХЭШ-ФУНКЦИЙ

Способы использования хэш-кода для аутентификации сообщений показаны на рис. 12.1, перечислены далее и обобщены в табл. 12.2:

а) Сообщение вместе с присоединенным к нему путем конкатенации хэш-кодом шифруется методами традиционного шифрования. Аргументация при этом следующая: поскольку только пользователям **A** и **B** известен секретный ключ, сообщение наверняка пришло от **A** и не могло быть изменено по пути следования. Хэш-код обеспечивает структуризацию или избыточность, требуемую для аутентификации. Поскольку шифрование выполняется по отношению ко всему сообщению вместе с добавленным хэш-кодом, то при этом обеспечивается и конфиденциальность.

б) Шифруется только хэш-код средствами традиционного шифрования. Это позволяет снизить вычислительную нагрузку на систему, если речь идет о приложениях, не требующих конфиденциальности. Обратите внимание на то, что хэширование и шифрование в комбинации фактически дают **код** аутентичности сообщения, т.е. $E_K[H(\mathbf{M})]$ представляет собой функцию сообщения **M** произвольной длины и секретного ключа ***K***, которая дает на выходе значение фиксированного размера, защищенное от противника, не знающего секретный ключ.

в) Шифруется только хэш-код средствами шифрования с открытым ключом с использованием личного ключа отправителя. При этом обеспечивается не только аутентификация, как в предыдущем случае (**б**), но и цифровая подпись, так как только отправитель может произвести соответствующим образом зашифрованный хэш-код. Фактически в этом и заключается суть техники использования цифровой подписи.

г) Если требуется обеспечение не только конфиденциальности, но и цифровой подписи, можно зашифровать сообщение вместе с хэш-кодом, шифрованным открытым ключом. Для этого используются методы традиционного шифрования с секретным ключом.

д) В целях аутентификации сообщений можно использовать функцию хэширования без шифрования. В таком случае предполагается, что две участвующие в обмене данными стороны используют известное только им секретное значение *S*. Отправитель **A** вычисляет значение функции хэширования для результата конкатенации **M** и *S* и присоединяет полученное значение функции хэширования к **M**. Получателю **B** значение *S* известно, поэтому он может тоже вычислить значение функции хэширования, чтобы сравнить последнее с пришедшим вместе с сообщением. Ввиду того что секретное значение непосредственно не посылается, противник не может модифицировать перехваченное сообщение или генерировать ложное.

е) Конфиденциальность может быть обеспечена при некоторой модификации подхода, описанного в пункте (д), если зашифровать сообщение вместе с добавленным к нему хэш-кодом.

Когда конфиденциальность не требуется, методы (б) и (в) оказываются предпочтительнее по сравнению с методами, в которых предполагается шифрование всего сообщения, поскольку требуют меньше вычислений.

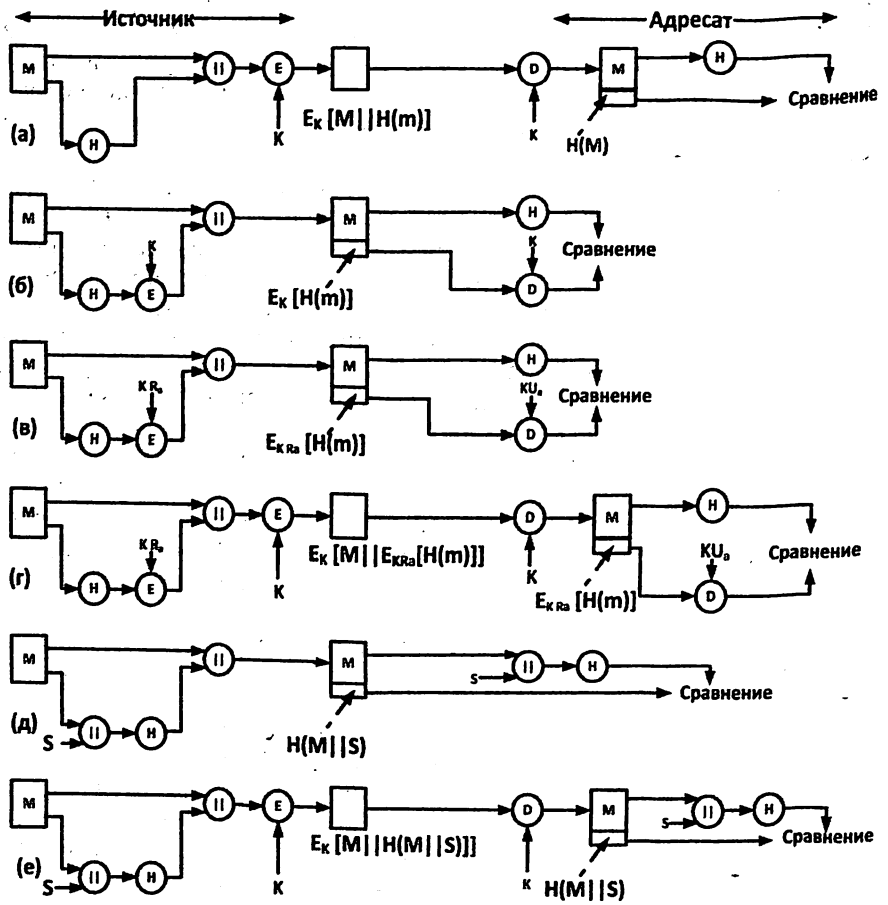


Рис. 12.1. Основные способы использования функции хэширования

Таблица 12.2

Основные возможности использования функции хэширования

(а) A→B: E_K[M‖H(M)] • Обеспечивает конфиденциальность – Только стороны **A** и **B** знают K • Обеспечивает аутентификацию – H(**M**) криптографически защищено	**(г) A→B:** E_K[**M** ‖ E_{KRa}[H(**M**)]] • Обеспечивает аутентификацию и цифровую подпись • Обеспечивает конфиденциальность – Только стороны **A** и **B** знают K
(б) A→B: **M** ‖ E_K[H(**M**)] • Обеспечивает аутентификацию - H(**M**) криптографически защищено	**(д) A→B:** **M** ‖ H(**M** ‖ S) • Обеспечивает аутентификацию – Только стороны **A** и **B** знают S
(в) A→B: **M** ‖ E_{KRa}[H(**M**)] • Обеспечивает аутентификацию и цифровую подпись – H(**M**) криптографически защищено – Только сторона **A** может создать E_{KRa}[H(**M**)]	**(е) A→B:** E_K[**M** ‖ H(**M** ‖ S)] • Обеспечивает аутентификацию – Только стороны **A** и **B** знают S • Обеспечивает конфиденциальность – Только стороны **A** и **B** знают K

И тем не менее, наблюдается постоянно возрастающий интерес к методам, которые позволяют избежать шифрования, что вызвано целым рядом следующих причин.

1. Программное обеспечение, выполняющее шифрование, работает довольно медленно. Даже если объем данных, которые шифруются при передаче сообщения, будет небольшим, поток исходящих и входящих сообщений в системе может оказаться очень интенсивным.

2. Цены на аппаратные средства шифрования довольно высокие. Хотя и имеются недорогие микросхемы, реализующие алгоритмы DES, общая их стоимость может оказаться очень высокой, если задаться целью оснастить ими все узлы сети.

3. Аппаратные средства шифрования оптимизируются для работы с большими объемами данных. При малых блоках данных значительная часть времени тратится непроизводительно на инициализацию/вызов.

4. Алгоритмы шифрования могут быть защищены патентами. Некоторые алгоритмы шифрования, например, алгоритм *RSA* шифрования с открытым ключом, запатентованы, и поэтому на их использование требуются лицензии, что тоже выливается в дополнительные расходы.

5. Алгоритмы шифрования являются одним из вопросов экспортного государственного регулирования США.

12.5. КРИПТОАНАЛИЗ ХЭШ-ФУНКЦИЙ

В последние годы были затрачены значительные усилия и достигнуты определенные успехи в деле разработки криптографических методов анализа функции хэширования. Чтобы понять их, рассмотрим структуру типичной защищенной функции хэширования, показанную на рис. 12.2.

Эту структуру, называемую итерированной функцией хэширования, предложил Меркл (Merkle), и именно такую структуру имеет большинство используемых сегодня функций хэширования, включая MD5, SHA-1 и RIPEMD-160.

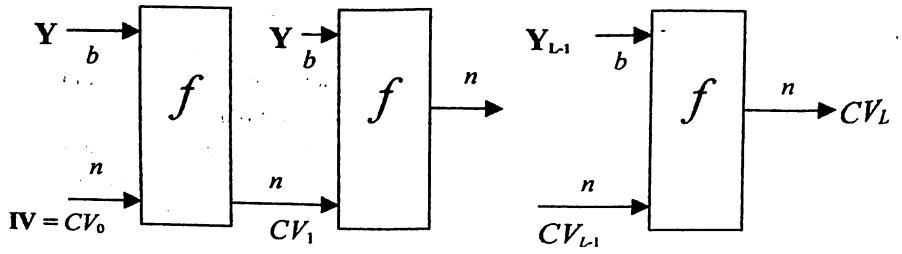


Рис. 12.2. Общая структура защищенного хэш-кода

На рис.12.2 приняты обозначения:
IV — начальное значение;
CV — переменная сцепления;
$\mathbf{Y}_i$ — *i*-й вводимый блок;
f — алгоритм сжатия;
L — число вводимых блоков;
N — длина хэш-кода;
b — длина вводимого блока.

Функция хэширования получает на вход сообщение и делит его на $L - 1$ блоков равной фиксированной длины по *b* битов каждый. Если необходимо, последний блок дополняется до *b* битов. В последний блок также включается значение суммарной длины ввода функции хэширования. Это делает задачу противника еще более сложной. Противник должен найти либо два сообщения равной длины, имеющие одинаковые значения функции хэширования, либо два сообщения разной длины, которые вместе с соответствующими им значениями длины будут иметь одинаковые значения функции хэширования.

Алгоритм хэширования предполагает многократное применение функции сжатия *f*, получающей на вход два значения:

– *n*-битовое значение, полученное на предыдущем этапе и называемое переменной сцепления;
– *b*-битовый блок сообщения,
и порождающее *n*-битовое выходное значение.

В начале хэширования переменная сцепления получает начальное значение, являющееся частью алгоритма. Конечное значение переменной сцепления и будет значением функции хэширования. Обычно $b > n$, поэтому и говорят о сжатии.

Функция хэширования может быть формально описана следующим образом:

$CV_0 = \mathrm{IV}$ = начальное *n*-битовое значение,
$CV_i = f(CV_{i-1}, Y_{i-1})$, $1 \le i \le L$,
$H(\mathbf{M}) = CV_L$, где вводимыми данными функции хэширования является сообщение **M**, складывающееся из блоков $Y_0, Y_1, \ldots, Y_{L-1}$.

Толчком к созданию такого рода итерационных структур послужили результаты Меркла и Дамгарда (*Damgard*), свидетельствующие о том, что если функция сжатия обладает сопротивляемостью коллизиям, то такой же будет и итерированная функция хэширования. Поэтому такая структура может использоваться для создания защищенной функции хэширования, работающей с сообщениями любой длины. Проблема создания защищенной функции хэширования сводится к проблеме поиска функции сжатия, обладающей сопротивляемостью коллизиям и работающей с вводимыми данными некоторой фиксированной длины.

Криптоанализ функций хэширования обычно сосредоточен на исследовании внутренней структуры *f* и опирается на попытки найти эффективные методы обнаружения коллизий при однократном выполнении f.

Если эта проблема решена, то атакующему остается рассмотреть фиксированное значение IV. Конкретный вид атаки на *f* зависит от внутренней структуры этой функции. Обычно, например когда речь идет о симметричных блочных шифрах, *f* предполагает несколько раундов обработки данных, так что лучше всего выполнять анализ изменения побитовой структуры данных от раунда к раунду.

Следует при этом иметь в виду, что коллизии должны существовать в любой функции хэширования, поскольку последняя отображает, как минимум, блок длины *b* в хэш-код длины *n*, где $b > n$.

Требуется лишь вычислительная невозможность обнаружить такие коллизии.

ГЛАВА 13. ЭЛЕКТРОННАЯ ЦИФРОВАЯ ПОДПИСЬ

13.1. ТРЕБОВАНИЯ К ЦИФРОВЫМ ПОДПИСЯМ И ИХ КЛАССИФИКАЦИЯ

13.1.1. ОБЩИЕ ПОЛОЖЕНИЯ

Постановка задачи. Передача сообщения отправителем (пользователь **A**) получателю (пользователь **B**) 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.

В ситуациях, когда нет полного доверия между отправителем и получателем, требуется нечто большее, чем простая аутентификация. Наиболее привлекательным решением этой проблемы оказывается **цифровая подпись.**

Цифровая подпись является аналогом подписи, сделанной от руки. Она должна обеспечивать следующие возможности.

1. Возможность установить автора, а также дату и время подписи.
2. Возможность установить достоверность содержимого сообщения на время подписи.
3. Возможность проверки подписи третьей стороной на случай возникновения спора.

Таким образом, функции цифровой подписи охватывают, в частности, и функции аутентификации.

Конкретизируя вышеизложенное, полагаем далее, что 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 известную только отправителю **A** закрытой информации **kA**.

2. Необходимо, чтобы правильную подпись **M**:

SIGN{kA, M, идентификатор **B}**

в сообщении для пользователя **B** нельзя было составить без **kA**.

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

4. Пользователь **B** должен иметь возможность удостовериться, что **SIGN{kA, M,** идентификатор **B}** есть правильная подпись **M** пользователем **A**.

Рассмотрим эти пункты подробнее.

1. Подпись сообщения — определенный способ шифрования **M** путем криптографического преобразования.

Закрываемым элементом **kA** в преобразовании

<Идентификатор **B, M** > →**SIGN{kA, M,** идентификатор **B}** является ключ криптопреобразования.

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

Во всех практических криптографических системах **kA** принадлежит конечному множеству ключей **K**. Исчерпывающая проверка всех ключей, задаваемых соответствующими парами

< M, идентификатор **B >** ↔ **SIGN{kA, M,** идентификатор **B}**

в общем должна привести к определению ключа **kA** злоумышленником. Если множество **K** достаточно велико и ключ **kA** определен методом случайного выбора, то полная проверка ключей невозможна.

Говоря, что составить правильную подпись без ключа невозможно, следует иметь в виду, что определение **SIGN{kA, M,** идентификатор **B}** без **kA** 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 от 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, продаже 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 цифровой обработки данных получатель мог установить подлинность отправителя, необходимо выполнение следующих требований к цифровой подписи:

1. Подпись должна быть двоичным кодом, который зависит от подписываемого сообщения.

2. Подпись должна использовать некоторую информацию, уникальную для отправителя, чтобы предотвратить возможность как фальсификации, так и отрицания авторства.

3. Цифровую подпись должно быть относительно просто произвести.

4. Цифровую подпись должно быть относительно просто распознать и проверить.

5. С точки зрения вычислений должно быть нереально фальсифицировать цифровую подпись ни с помощью создания нового сообщения, ни с помощью расшифровки созданной подписи.

Так, например, защищаемая функция хэширования, встроенная в соответствующую схему, подобную показанным на рис. 12.1 (в) или 12.1 (г), удовлетворяет этим требованиям.

Для реализации идеи цифровой подписи было предложено множество подходов, которые можно разбить на две категории: с непосредственной и арбитражной логикой.

13.1.2. НЕПОСРЕДСТВЕННАЯ ЦИФРОВАЯ ПОДПИСЬ

Непосредственная цифровая подпись подразумевает участие только обменивающихся данными сторон (источник, адресат). Предполагается, что адресат знает открытый ключ источника. Цифровая подпись может быть сформирована с помощью шифрования всего сообщения личным ключом отправителя или с помощью шифрования хэш-кода сообщения личным ключом отправителя.

После этого конфиденциальность может быть обеспечена шифрованием всего сообщения вместе с подписью — либо с помощью открытого ключа получателя (шифрование с открытым ключом), либо с помощью общего секретного ключа (традиционное шифрование) (см. рис. 12.1 (г)).

Следует обратить внимание на то, что важно сначала выполнить функцию подписи и только потом — внешнюю функцию, обеспечивающую конфиденциальность. В случае возникновения разногласий некая третья сторона должна рассмотреть сообщение и подпись. Если вычислять подпись для шифрованного сообщения, то третьей стороне, чтобы прочитать оригинальное сообщение, потребуется доступ к ключу дешифрования. Если же подпись является внутренней операцией, получатель сможет сохранить сообщение в виде открытого текста и подпись для возможного использования в дальнейшем в процессе разрешения конфликта.

Все до сих пор предлагавшиеся схемы непосредственного применения цифровой подписи имеют общее слабое место: пригодность всей схемы зависит от защищенности личного ключа отправителя. Если отправитель впоследствии решит отрицать отправку конкретного сообщения, он может заявить, что личный ключ был утерян или украден, и поэтому кто-то другой с помощью этого ключа фальсифицировал подпись.

Для того чтобы воспрепятствовать или, по крайней мере, помешать применению такого коварного приема, можно прибегнуть к административным средствам контроля, имеющим отношение к защите личных ключей, но угроза при этом все равно полностью не устраняется. Одной из возможностей здесь является требование, чтобы каждое подписанное сообщение включало метку даты / времени, а также требование немедленно сообщать о любом случае компрометации ключа в уполномоченный центр.

Другая угроза заключается в том, что личный ключ может быть действительно похищен у **X** в момент времени **T**. После этого противник получает возможность отослать сообщение с подписью **X**, помеченное временем более ранним или равным **T**.

13.1.3. АРБИТРАЖНАЯ ЦИФРОВАЯ ПОДПИСЬ

Проблемы, возникающие при использовании непосредственных цифровых подписей, могут решаться с помощью использования арбитра (третьей стороны). Как и для непосредственных цифровых подписей, имеется множество схем применения арбитражных цифровых подписей. В общем, они все строятся следующим образом. Каждое подписанное сообщение отправителя **X** адресату **Y** сначала попадает к арбитру **A**, который подвергает сообщение и подпись к нему тестированию по ряду критериев, чтобы проверить достоверность источника и содержимого сообщения. После этого сообщение датируется и посылается **Y** с указанием того, что это сообщение было проверено и удовлетворило критериям арбитра.

Наличие **A** решает проблему, возникающую в схемах использования непосредственных цифровых подписей, когда **X** может отказаться от авторства своего сообщения.

В таких схемах арбитр играет исключительно важную роль, и все участвующие в обмене данными стороны должны иметь очень высокую степень доверия к механизму арбитражного устройства.

В табл. 13.1 представлено несколько примеров схем арбитражных цифровых подписей.

В первом примере используется традиционное шифрование. Предполагается, что отправитель **X** и арбитр **A** используют общий секретный ключ K_{xa}, а **A** и **Y** — общий секретный ключ K_{ay}. Отправитель **X** создает сообщение **M** и вычисляет значение функции хэширования $H(\mathbf{M})$. Затем **X** передает сообщение с добавленной к нему подписью *арбитру* **A**. Подпись складывается из идентификатора **X** ($\mathbf{ID}_X$) и значения функции хэширования $H(\mathbf{M})$, и все это шифруется с использованием K_{xa}. Арбитр **A** дешифрует подпись и проверяет значение функции хэширования $H(\mathbf{M})$, чтобы убедиться в достоверности сообщения. Потом **A** передает сообщение адресату **Y** в зашифрованном с помощью K_{ay} виде. Это сообщение включает $\mathbf{ID}_X$, оригинальное сообщение **M** и метку даты / времени. Получатель **Y** может дешифровать его, чтобы восстановить сообщение и подпись. Метка даты / времени информирует **Y**, что это сообщение получено своевременно и не является воспроизведением. Теперь **Y** может сохранить **M** и подпись. В случае спора **Y**, заявляющий, что получил **M** от **X**, отправит **A** сообщение следующего содержания:

$$\mathbf{E}_{Kay}[\mathbf{ID}_X||\mathbf{M}||\mathbf{E}_{Kxa}[\mathbf{ID}_X||H(\mathbf{M})]].$$

Арбитр использует K_{ay}, чтобы восстановить $\mathbf{ID}_X$, **M** и подпись, а затем обращается к помощи K_{xa}, чтобы дешифровать подпись и проверить хэш-код. В этой схеме **Y** не может непосредственно проверить подпись **X** — для него подпись присутствует исключительно в качестве элемента, к которому можно будет обратиться в случае возникновения конфликта для его разрешения. Получатель **Y** считает сообщение от **X** подлинным, поскольку оно пришло от **A**. Этот сценарий предполагает, что обе стороны должны иметь очень высокую степень доверия к **A**:

- **X** должен верить, что **A** не разгласит K_{xa} и не будет генерировать фальшивые подписи вида $\mathbf{E}_{Kxa}[\mathbf{ID}_X||H(\mathbf{M})]$;
- **Y** должен верить, что **A** будет посылать $\mathbf{E}_{Kay}[\mathbf{ID}_X||\mathbf{M}||\mathbf{E}_{Kxa}[\mathbf{ID}_X||H(\mathbf{M})]||\mathbf{T}]$ только в тех случаях, когда значение функции хэширования $H(\mathbf{M})$ оказывается правильным и подпись действительно была выполнена отправителем **X**;

- обе стороны должны быть уверены, что **A** будет честно разрешать конфликты.

Если арбитр действительно будет отвечать таким ожиданиям, то **X** может быть уверен, что никто не сможет фальсифицировать его подпись, а **Y** может быть уверен, что **X** не сможет дезавуировать свою подпись.

Таблица 13.1

Варианты схем арбитражных цифровых подписей

а) Традиционное шифрование, арбитр может видеть сообщение
(1)**X**→**A**: $\mathbf{M} \,||\, \mathbf{E}_{Kxa}[\mathbf{ID}_X \,||\, H(\mathbf{M})]$; (2)**A**→**Y**: $\mathbf{E}_{Kay}[\mathbf{ID}_X||\mathbf{M}||\mathbf{E}_{Kxa}[\mathbf{ID}_X||H(\mathbf{M})]||\mathbf{T}]$.
б) Традиционное шифрование, арбитр не видит сообщения
(1)**X**→**A**: $\mathbf{ID}_X \,||\, \mathbf{E}_{Kxy}[\mathbf{M}] \,||\, \mathbf{E}_{Kxa}[\mathbf{ID}_X \,||\, H(\mathbf{E}_{Kxy}[\mathbf{M}])]$; (2)**A**→**Y**: $\mathbf{E}_{Kay}[\mathbf{ID}_X \,||\, \mathbf{E}_{Kxy}[\mathbf{M}] \,||\, \mathbf{E}_{Kxa}[\mathbf{ID}_X \,||\, H(\mathbf{E}_{Kxy}[\mathbf{M}])] \,||\, \mathbf{T}]$.
в) Шифрование с открытым ключом, арбитр не видит сообщения
(1)**X**→**A**: $\mathbf{ID}_X|| \,\mathbf{E}_{KRx}[\mathbf{ID}_X \,||\, \mathbf{E}_{KUy}(\mathbf{E}_{KRx}[\mathbf{M}])]$; (2)**A**→**Y**: $\mathbf{E}_{KRa}[\mathbf{ID}_X \,||\, \mathbf{E}_{KUy}(\mathbf{E}_{KRx}[\mathbf{M}]) \,||\, \mathbf{T}]$.

В табл. 13.1. использованы обозначения: **X** — отправитель, **Y** — получатель, **A** — арбитр, **M** — сообщение.

Предыдущий сценарий подразумевает, что **A** может прочитать сообщения от **X** к **Y**, но то же самое может сделать и любой перехватчик сообщений. В табл. 13.1(б) показан сценарий, в котором, как и в предыдущем сценарии, обеспечивается арбитраж, но, кроме того, гарантируется и конфиденциальность. В данном случае предполагается, что **X** и **Y** используют общий секретный ключ K_{xy}. Теперь **X** передает арбитру **A** идентификатор, экземпляр сообщения, зашифрованный с помощью K_{xy}, и подпись. Подпись складывается из идентификатора и значения функции хэширования шифрованного сообщения, и все это шифруется с использованием K_{xa}. Как и прежде, **A** дешифрует подпись и проверяет значение функции хэширования $H(\mathbf{M})$, чтобы проверить подлинность полученного сообщения. В данном случае **A** работает только с шифрованной версией сообщения и поэтому не имеет возможности прочитать его. После проверки **A** передает адресату **Y** то, что было получено от **X**, добавив метку даты / времени и зашифровав все это с помощью K_{ay}.

Не имея возможности прочитать сообщение, арбитр все же оказывается способным предотвратить обман и со стороны **X**, и со стороны **Y**. Проблемами, которые остаются здесь от первого сценария,

являются сговор арбитра с отправителем с целью отрицания факта отправки подписанного сообщения, или сговор арбитра с получателем с целью фальсификации подписи отправителя.

Все вышеупомянутые сложности могут быть преодолены с помощью схемы с открытым ключом, один из вариантов которой показан в табл. 13.1 (в). В этом случае **X** дважды шифрует сообщение — сначала с помощью личного ключа **X** (ключа KR_X), а потом с помощью открытого ключа **Y** (ключа KU_y). Это будет подписанная секретная версия сообщения. Данное подписанное сообщение наряду с идентификатором **X** (ID_X) шифруется снова с помощью KR_x и вместе с ID_X посылается **A**. Внутреннее, дважды шифрованное сообщение защищено от арбитра (и любого другого лица, кроме **Y**). Однако можно снять внешнее шифрование и убедиться, что сообщение наверняка пришло от **X** (поскольку только **X** имеет KR_X). Арбитр **A** должен убедиться в том, что пара личный / открытый ключи **X** еще действительна, и если это так, проверить само сообщение. Затем **A** передает сообщение **Y**, шифруя его с помощью KR_a. Сообщение включает ID_a, дважды шифрованное сообщение и метку даты / времени.

Эта схема имеет ряд преимуществ по сравнению с двумя предыдущими.

Во-первых, в совместном распоряжении сторон до начала обмена данными нет никакой информации, что предотвращает возможность сговора с целью обмана.

Во-вторых, некорректно датированное сообщение не может быть передано, даже если KR_X скомпрометирован, если только не скомпрометирован KR_a.

Наконец, содержимое сообщения от **X** к **Y** является секретом для **A**, как и для всех остальных.

13.2. ОСНОВНЫЕ АЛГОРИТМЫ ЦИФРОВЫХ ПОДПИСЕЙ

Электронная цифровая подпись (ЭЦП) Эль-Гамаля. 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 (*El Gamal*) аналогично тому, как это делается в системе шифрования, носящей его имя.

Выбирается большое простое число p и целое число g, являющееся примитивным элементом в Zp. Эти числа публикуются. Затем выбирается секретное число x и вычисляется открытый ключ для проверки подписи

$$y = g^x (\mathrm{mod}\ p).$$

Далее для подписи сообщения **M** вычисляется его хэш-функция $h = H(\mathbf{M})$.

Выбирается случайное целое k: $1 < k < (p - 1)$, взаимно простое с $p - 1$, и вычисляется

$$r = g^k (\mathrm{mod}\ p).$$

После этого с помощью расширенного алгоритма Евклида решается относительно s уравнение

$$h = (xr + ks)(\mathrm{mod}\ (p - 1)).$$

Подпись образует пара чисел (r, s).

После выработки подписи значение k уничтожается.

Получатель подписанного сообщения вычисляет хэш-функцию сообщения $h = H(\mathbf{M})$ и проверяет выполнение равенства

$$y^r r^s (\mathrm{mod}\ p) = g^h.$$

Корректность этого уравнения очевидна:

$$y^r r^s = g^{xr} g^{ks} = g^{xr+ks} = g^h (\mathrm{mod}\ p).$$

ЭЦП Шнорра. Еще одна подобная схема была предложена Шнорром (*Schnorr*). Как обычно, p — большое простое число; q — простой делитель $(p - 1)$; g — элемент порядка q в Zp; k — случайное число, x и $y = g^x (\mathrm{mod}\ p)$ — секретный и открытый ключи соответственно.

Уравнения выработки подписи выглядят следующим образом:

$$r = g^k (\mathrm{mod}\ p);$$
$$h = H(\mathbf{M}, r);$$
$$s = k + xh (\mathrm{mod}\ q).$$

Подписью является пара (r, s).

На приемной стороне вычисляется значение хэш-функции $h = H(\mathbf{M}, r)$ и проверяется выполнение равенства

$$r = g^s y^{-h} (\mathrm{mod}\ p),$$

при этом действия с показателями степени производятся по модулю q.

Другой вариант подписи Шнорра выглядит так. Для подписи сообщения **M** автор выбирает случайное k, вычисляет

$$g^k (\mathrm{mod}\ p),$$
$$h = H(g^k, \mathbf{M}),$$
$$z = k + x\,h\ (\mathrm{mod}\ q).$$

Подписью является тройка $(\mathbf{M}, h, z)$.

Проверка подписи заключается в проверке равенства

$$H(g^{z} y^{-h}, \mathbf{M}) = h.$$

В самом деле,

$$g^{z} y^{-h} = g^{k \cdot xh} g^{-xh} = g^{k}.$$

Стандарт ЭЦП DSS. Федеральный стандарт обработки информации FIPSPUB 186, известный также как DSS (Digital Signature Standard — стандарт цифровой подписи), опубликован Национальным институтом стандартов и технологии США (NIST). Стандарт DSS основан на алгоритме хэширования SHA (Secure Hash Algorithm — защищенный алгоритм хэширования). Стандарт DSS был предложен в 1991 г., а его исправленная версия — в 1993 г. в ответ на возникшие сомнения в безопасности соответствующей схемы. В 1996 г. в него были внесены незначительные изменения.

Новая редакция стандарта на выработку и верификацию цифровой подписи DSS принята в США 7 января 2000 г. (FIPSPUB 186-2).

Согласно этому стандарту, электронная цифровая подпись может вырабатываться по одному из трех алгоритмов:

– DSA (Digital Signature Algorithm) — алгоритм, основанный на проблеме логарифма в конечном поле;

– ANSI X9.31 (RSA DSA),

– ANSI X9.63 (EC DSA) — алгоритм выработки подписи, основанный на проблеме логарифма в группе точек эллиптической кривой над конечным полем.

В России вычисление дайджеста и реализацию электронной подписи регламентируют 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».

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

Структура DSS. В стандарте DSS используется алгоритм, призванный обеспечить только функцию цифровой подписи. В отличие от RSA, данный алгоритм не может служить для шифрования или обмена ключами. Однако это пример технологий криптографии с открытым ключом.

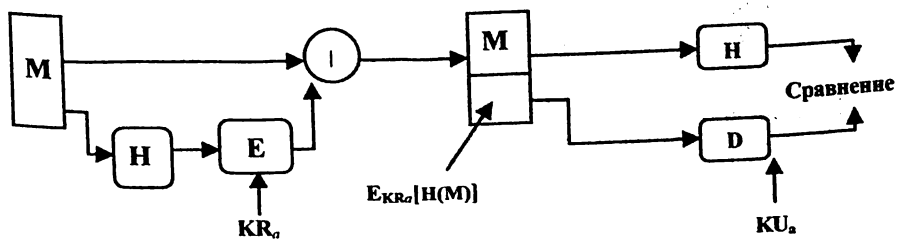
На рис. 13.1 сравниваются схемы генерирования цифровых подписей с использованием DSS и RSA. При подходе на основе RSA сообщение, которое должно быть подписано, поступает на вход функции хэширования, которая выдает защищенный хэш-код фиксированной длины. Этот хэш-код шифруется затем личным ключом отправителя, в результате чего и получается подпись. После этого передаются и сообщение, и его подпись. Получатель принимает сообщение и вычисляет хэш-код. Кроме того, он дешифрует подпись, используя открытый ключ отправителя. Если вычисленный хэш-код соответствует дешифрованной подписи, подпись считается подлинной. Поскольку только отправитель знает свой личный ключ, подлинную подпись может поставить только он.

Подход *DSS* также основан на функции хэширования. Хэш-код подается на вход функции создания подписи вместе со случайным числом k, специально генерируемым для данной подписи. Функция создания подписи зависит также от личного ключа отправителя $\mathbf{KR}_a$ и ряда параметров, известных группе сообщающихся сторон. Можно считать, что это множество составляет глобальный открытый ключ $\mathbf{KU}_G$.

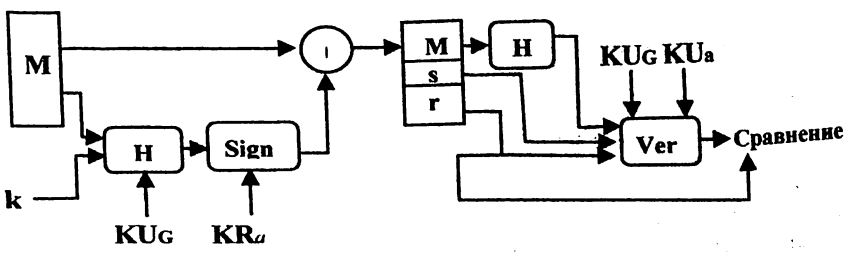
В результате получается подпись, складывающаяся из *двух* компонентов, обозначаемых s и r.

В пункте назначения генерируется хэш-код поступившего сообщения. Этот код и подпись подаются на вход функции верификации. Функция верификации зависит также от глобального открытого ключа и от открытого ключа отправителя $\mathbf{KU}_a$, соответствующего личному ключу отправителя. Если подпись является подлинной, то на выходе функции верификации получается значение, совпадающее с компонентом r подписи. Функция создания подписи такова, что только отправитель, зная свой личный ключ, может создать правильную подпись.

Обратимся теперь к рассмотрению деталей алгоритма.



(а) Подход RSA



(б) Подход DSS

Рис. 13.1. Два подхода к использованию цифровых подписей

Алгоритм цифровой подписи DSA. Алгоритм цифровой подписи *DSA* (*Digital Signature Algorithm*) создан с учетом трудностей вычисления дискретных логарифмов и опирается на схемы, предложенные Эль-Гамалем и Шнорром.

Алгоритм схематически представлен на рис. 13.2.

Глобальные компоненты открытого ключа

p —простое число, $2^{L-1} < p < 2^L$, где $512 < L < 1024$ и L является кратным 64, т.е. длиной между 512 и 1024 битами с шагом 64 бита;

q — простой делитель $(p - 1)$, где $2^{159} < q < 2^{160}$, т.е. длиной 160 битов;

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

где h является любым целым числом, таким, что $1 < h < (p - 1)$ и $h^{(p-1)/q} \bmod p > 1$.

Личный ключ пользователя

x — случайное или псевдослучайное число, $0 < x < q$

Открытый ключ пользователя

$$y = (g^x) \bmod p$$

Секретный номер сообщения пользователя

k– случайное или псевдослучайное число, $0 < k < q$

Создание **подписи**

$$r = (g^k \bmod p) \bmod q;$$

$$s = [k^{-1} (H(\mathbf{M}) + xr)] \bmod q.$$

Подпись(r, s)

Верификация

$$w = (s')^{-1} \bmod q;$$

$$u1 = H(\mathbf{M}')w \bmod q;$$

$$u2 = (r')w \bmod q;$$

$$v = [(g^{u1})(y^{u2}) \bmod p] \bmod q.$$

ПРОВЕРКА: $v = r'$

M — подписываемое сообщение;
H(**M**) — хэш-код **M** по методу SHA-1;
M', r', s'— полученные версии **M**, r и s.

Рис. 13.2. Алгоритм цифровой подписи (DSA)

В алгоритме имеется три параметра, которые являются открытыми и предполагаются известными группам пользователей.

Выбирается 160-битовое простое число q. Затем выбирается такое простое число p длиной между 512 и 1024 битами, что q делит $(p - 1)$.

Наконец, выбирается число g вида $h^{(p-1)/q} \bmod p$, где h является целым числом между 1 и $(p - 1)$ с тем ограничением, что g должно быть больше 1.

Имея эти числа, каждый пользователь выбирает личный ключ и генерирует открытый ключ.

Личный ключ x должен быть числом от 1 до $(q - 1)$;которое должно выбираться случайным или псевдослучайным образом.

Открытый ключ вычисляется на основе личного ключа по формуле $y = g^x \bmod p$. Вычислить y по имеющемуся значению x относительно просто. Однако при имеющемся значении открытого ключа y задача определения значения x по значению y считается нереальной, поскольку для этого требуется вычислить дискретный логарифм y по основанию g и по модулю p.

Чтобы создать подпись, пользователь вычисляет две величины: r и s, являющиеся функциями компонентов открытого ключа (p, q, g), личного ключа пользователя x, хэш-кода сообщения H(**M**) и некоторого целого числа k, которое должно выбираться случайным или псевдослучайным образом и быть уникальным для каждого выполнения подписи.

В пункте назначения выполняется верификация, для чего используются формулы, показанные на рис. 13.2. Получатель генерирует величину v, являющуюся функцией компонентов открытого ключа, открытого ключа отправителя и хэш-кода поступившего сообщения. Если эта величина соответствует компоненту r подписи, то подпись подтверждается.

На рис. 13.3 представлены функции выполнения подписи и верификации.

Структура алгоритма, как видно из рис. 13.3, весьма интересна. Следует обратить внимание на то, что проверка в конце осуществляется со значением r, которое не зависит от сообщения вообще. Значение r является функцией k и *mpex* компонентов глобального открытого ключа. Мультипликативное обратное значение k (mod q) передается функции, которая получает на вход также хэш-код сообщения и личный ключ пользователя. Структура этой функции такова, что получатель может восстановить значение r по поступившим сообщению и подписи, открытому ключу пользователя и глобальному открытому ключу (т.е. осуществить верификацию электронной подписи) следующим образом.

Пусть было принято сообщение **m**.

Тогда уравнение проверки имеет вид:

$$r \equiv (g^{H(\mathbf{m}) \times s^{-1}} \times y^{r \times s^{-1}} \bmod p) (\bmod q).$$

Действительно,

$$(g^{H(\mathbf{m}) \times s^{-1}} \times y^{r \times s^{-1}} \times \bmod p)(\bmod q) = (g^{H(\mathbf{m}) \times s^{-1}} \times g^{xr \times s^{-1}} \times \times \bmod p)(\bmod q) = (g^{(H(\mathbf{m}) + xr) \times s^{-1}} \times \bmod p)(\bmod q) = (g^{k^{-1} \times (H(\mathbf{m}) + xr)^{-1} \times}$$

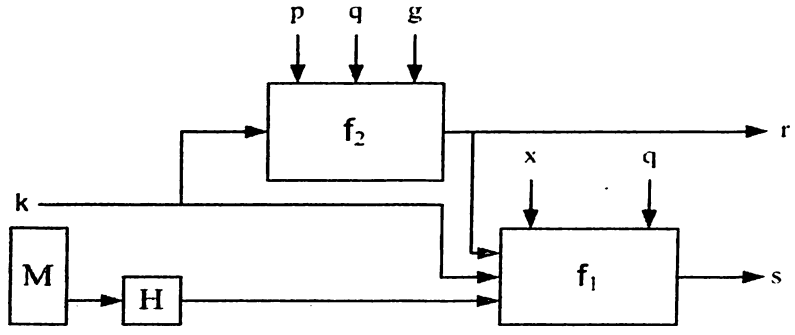
$^{\times (H(\mathbf{m})+ xr)} \bmod p)(\bmod q) = (g^{(k^{-1})^{-1} (H(\mathbf{m})+ xr)^{-1} \cdot (H(\mathbf{m})+ xr)} \bmod p)(\bmod q) =$
$= (g^k \bmod p)(\bmod q) \equiv r;$

$n = f_1(H(\mathbf{M}), k, x, r, q) = (k^{-1}(H(\mathbf{M}) + xr)) \bmod q;$

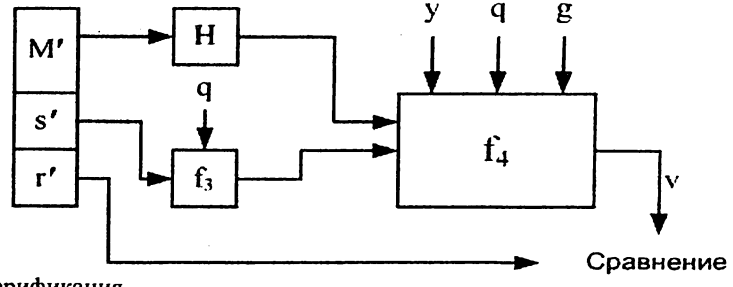
$r = f_2(k, p, q, g) = (g^k \bmod p) \bmod q;$

$w = f_3(s9, q) = (s')^{-1} \bmod q;$

$v = f_4(y, q, g, H(\mathbf{M}'), w, r') = ((g^{(H(\mathbf{M}')w) \bmod q y r' w \bmod q \bmod p) \bmod q}$



(а) Создание подписи



(б) Верификация

Рис. 13.3. Подпись и верификация DSS′

При трудности вычисления дискретных логарифмов для противника оказывается нереальным с точки зрения вычислений найти k по известному r или найти x по известному s.

В алгоритме создания подписи интенсивные вычисления потребуются только при возведении в степень для вычисления $(g^k) \bmod p$. Поскольку это значение не зависит от подписываемого сообщения, оказывается возможным вычислить значение заранее. На самом деле, пользователь может заранее вычислить целый ряд значений r, чтобы пользоваться ими при создании подписей документов по мере необходимости.

Другой, отчасти требующей некоторых усилий, задачей является определение мультипликативного обратного k^{-1}. Здесь также имеется возможность заранее вычислить целый ряд таких значений.

ГЛАВА 14. ПРОТОКОЛЫ АУТЕНТИФИКАЦИИ

14.1. ВЗАИМНАЯ АУТЕНТИФИКАЦИЯ

Важной областью приложений являются протоколы взаимной аутентификации. Такие протоколы дают возможность обеим участвующим в обмене данными сторонам убедиться в подлинности друг друга и обменяться сеансовыми ключами. Эта тема уже обсуждалась в гл. 6 (традиционные методы) и 11 (схемы с открытым ключом), где в центре внимания было распределение ключей. В этой главе будут рассмотрены возможности аутентификации в более широком контексте.

В задаче авторизованного распределения ключей главными оказываются две проблемы: конфиденциальность и своевременность. Чтобы предотвратить возможность имитации и не допустить компрометацию сеансовых ключей, значительная часть информации по аутентификации и сеансовым ключам должна передаваться в шифрованном виде. Это предполагает наличие секретных или открытых ключей, используемых с этой целью и уже существующих до начала обмена данными.

Вторая проблема — своевременность — важна ввиду угрозы возможности воспроизведения сообщения. Такое воспроизведение в самом худшем случае может позволить противнику скомпрометировать сеансовый ключ и выступать от имени другой стороны. Как минимум, успешное воспроизведение сообщения может разрушить взаимодействие сторон с помощью представления сторон сообщениями, которые кажутся подлинными, но на самом деле таковыми не являются.

Возможны следующие атаки с использованием воспроизведения сообщений:

1. *Простое воспроизведение*. Противник просто копирует сообщение и позже воспроизводит его.

2. *Воспроизведение, которое можно зарегистрировать*. Противник может воспроизвести сообщение с меткой даты / времени внутри реального времени.

3. *Воспроизведение, которое нельзя обнаружить*. Эта ситуация может иметь место ввиду того, что оригинальное сообщение можно принудительно задержать, чтобы оно не достигло адресата, тогда адресат получит только воспроизведенное сообщение.

4. *Возвратное воспроизведение без модификации*. Это воспроизведение сообщения, возвращаемое обратно отправителю. Такой вид атаки оказывается возможным, когда используется традиционное

шифрование и отправителю не слишком просто отличить по содержимому посылаемые сообщения от получаемых.

Одним из способов противодействия атакам с воспроизведением сообщений является присоединение к каждому сообщению порядкового номера, используемого в процессе аутентификации. Новое сообщение принимается только тогда, когда оно имеет соответствующий порядковый номер. Трудность здесь в том, что при этом каждой из сторон необходимо отслеживать порядковые номера для каждого из отправителей, с которыми приходится иметь дело. Из-за такой дополнительной нагрузки порядковые номера в процессах аутентификации и обмена ключами обычно не используются. Вместо этого применяется один из следующих общих подходов.

1. *Метки даты / времени.* Сторона **A** принимает сообщение как новое, только если сообщение содержит метку даты / времени, значение которой, по мнению **A**, достаточно близко к значению текущего времени в системе **A**. При этом подходе необходимо, чтобы часы участвующих в обмене данными сторон были синхронизированы.

2. *Запрос / ответ.* Сторона **A**, ожидающая новое сообщение от **B**, сначала посылает **B** оказию (запрос) и требует, что следующее сообщение, полученное от **B** (ответ), содержало соответствующее корректное значение оказии.

При этом следует иметь в виду, что метки даты / времени нельзя использовать для приложений, основанных на применении протокола с установлением логических соединений, из-за внутренних свойств этой технологии.

Во-первых, требуется какой-нибудь протокол, обеспечивающий синхронизацию часов процессоров разных систем. Этот протокол должен быть, с одной стороны, толерантным к отказам, чтобы распознавать сетевые ошибки, а с другой — защищенным, чтобы противостоять атакам враждебной стороны.

Во-вторых, шансы успешного завершения атак возрастают, когда возникают временные нарушения синхронизации из-за отказов часового устройства одной из сторон. Кроме того, из-за спонтанных и непредсказуемых задержек в сетях, что для сетей вполне естественно, нельзя ожидать, что распределенные часы могут поддерживать точную синхронизацию.

Поэтому любая процедура, использующая метки даты / времени, должна допускать для окна времени достаточно широкие рамки, чтобы учитывать возможность задержек в сети, но в то же время эти рамки должны быть достаточно узкими, чтобы минимизировать возможности для атак.

Подход с использованием запросов / ответов не годится для приложений, работающих с протоколами без установления соединений,

поскольку в таких приложениях потребуется очень большое число дополнительных подтверждений перед передачей любой порции данных, что сводит на нет главные преимущества модели обмена данными без установления соединений. Для таких приложений лучшим подходом является применение некоторого надежного сервера времени и согласование попыток каждой из сторон содержать свои часы в синхронизации с часами такого сервера.

Подходы на основе традиционного шифрования. Для того чтобы обеспечить конфиденциальность связи в распределенной среде, можно использовать двухуровневую иерархию ключей традиционного шифрования. В общем случае эта стратегия предполагает наличие и использование надежного центра распределения ключей (ЦРК). Каждая участвующая в обмене данными сторона в сети имеет свой секретный ключ, называемый главным ключом и используемый совместно с ЦРК. ЦРК несет ответственность за создание ключей, которые действуют в течение недолгого времени для соединений между двумя сторонами и который называются сеансовым ключом , а также за распределение таких ключей некоторым защищенным с помощью главных ключей способом. Такой подход является общепринятым.

На рис. 6.4 показана схема, изначально предложенная Нидхэмом (*Needham*) и Шредером (*Schroeder*) для распределения секретных ключей с помощью ЦРК, что включает элементы аутентификации. Соответствующий протокол можно представить следующим образом:

1. A →ЦРК: $ID_A \| ID_B \| N_1$
2. ЦРК →A: $E_{Ka}[K_s \| ID_B \| N_1 \| E_{Kb}[K_s \| ID_A]]$
3. A→B: $E_{Kb}[K_s \| ID_A]$
4. B→A: $E_{Ks}[N_2]$
5. A→B: $E_{Ks}[f(N_2)]$

Секретные ключи $\mathbf{K}_a$ и $\mathbf{K}_b$ используют совместно **A** и ЦРК, и **B** и ЦРК соответственно. Целью применения протокола является защищенная передача сеансового ключа $\mathbf{K}_s$ сторонам **A** и **B**. Сторона **A** получает новый сеансовый ключ в шаге 2 приведенной схемы. Сообщение, передаваемое на шаге 3 схемы, может быть дешифровано и прочитано только стороной **B**. Шаг 4 отражает знание ключа $\mathbf{K}_s$ стороной **B**, а шаг 5 убеждает сторону **B** в том, что ключ $\mathbf{K}_s$ известен и стороне **A**, и в том, что соответствующее сообщение является новым, так как в нем используется оказия $\mathbf{N}_2$.

Целью шагов 4 и 5 является предотвращение возможности воспроизведения сообщений определенного типа противником. В частности, противник сможет перехватить сообщение на шаге 3 и затем воспроизвести его, что может определенным образом нарушить действия **B**.

Несмотря на подтверждения, выполняемые в ходе шагов 4 и 5, этот протокол все еще оказывается уязвимым в отношении атак с воспроизведением сообщений некоторого специального вида. Предположим, что противник **X** каким-то образом сможет скомпрометировать старый сеансовый ключ. Скорее всего, такая возможность гораздо менее вероятна по сравнению с тем, что противник сможет просто перехватить и записать сообщение, передаваемое на шаге 3. Однако такая угроза потенциально все же существует. Противник **X** может выступить от лица **A** и обмануть **B**, заставив использовать старый ключ, для чего просто воспроизведет в соответствующем виде шаг 3. Если **B** не помнит точно все предыдущие сеансовые ключи, использовавшиеся с **A**, то **B** не сможет определить, что полученное сообщение является воспроизведением. Если некто **C** может перехватить сообщение квитирования на шаге 4, то сможет и отправить ответ (шаг 5) от лица **A**. После этого **C** получает возможность посылать поддельные сообщения **B**, которые будут казаться **B** пришедшими от **A** ввиду использования достоверного сеансового ключа.

Деннинг (*Denning*) предложил устранить этот недостаток путем модификации протокола Нидхэма – Шредера, которая заключается в том, что на шагах 2 и 3 добавляются метки даты / времени. При этом предполагается, что главные ключи $\mathbf{K}_a$ и $\mathbf{K}_b$ являются защищенными, а вся процедура складывается из следующих шагов:

1. A →ЦРК: $ID_A \| ID_B$
2. ЦРК →A: $E_{Ka} [K_s \| ID_B \| T \| E_{Kb}[K_s \| ID_A \| T]]$
3. A→B: $E_{Kb}[K_s \| ID_A \| T]$
4. B →A: $E_{Ks}[N_1]$
5. A→B: $E_{Ks}[f(N_1)]$

Здесь **T** является меткой даты / времени, использование которой убеждает стороны **B** и **A** в том, что сеансовый ключ был создан *только что*. Таким образом, как участник **A**, так и участник **B** будут знать, что данная сессия распределения ключей соответствует текущему времени. Стороны **A** и **B** могут убедиться в соответствии времени, проверив неравенство

$$| \text{Время} — T | < \Delta t_1 + \Delta t_2,$$

где Δt_1 обозначает оценку для среднего отклонения показаний локальных часов (**A** или **B**) от показаний часов ЦРК, а Δt_2 — ожидаемое время задержки в сети.

Каждый узел может настраивать свои часы, сравнивая их с часами некоторого стандартного справочного источника. Ввиду того что метка даты / времени **T** шифруется с использованием защищенных главных ключей, противник, даже зная старый сеансовый ключ, не

сможет добиться успеха из-за того, что воспроизведение сообщения, передаваемого на шаге 3, будет сразу же обнаружено **B** как не соответствующее времени получения.

В заключение отметим, что шаги 4 и 5 не входили в оригинальную схему, а были добавлены позже. Эти действия подтверждают получение сеансового ключа стороной **B**.

Протокол Деннинга, по-видимому, обеспечивает более высокий уровень безопасности по сравнению с протоколом Нидхэма – Шредера. Однако здесь возникает новая проблема: эта усовершенствованная схема требует зависимости от часов, показания которых должны быть синхронизированы в сети. При этом возникает следующий риск: распределенные часы могут стать рассинхронизированными либо в результате преднамеренных действий противника, либо в результате отказа часов или механизма синхронизации. Проблема возникает тогда, когда часы отправителя опережают часы предполагаемого адресата. В этом случае противник может перехватить сообщение, идущее от отправителя, и воспроизвести это сообщение позже, когда метка даты / времени в сообщении станет соответствовать текущим показаниям часов узла адресата. Такое воспроизведение может привести к непредсказуемым результатам. Гонг (*Gong*) называет такие атаки атаками задержки-воспроизведения.

Одним из способов противостоять подобным атакам является введение требования регулярного сравнения показаний часов сторон с показаниями часов ЦРК.

Другим вариантом, когда нет необходимости синхронизации часов, является применение протоколов подтверждения связи, подразумевающих оказии. Такой вариант оказывается неуязвимым в отношении атак задержки-воспроизведения, поскольку оказии, которые получатель выберет в будущем, не могут быть известны отправителю заранее.

Протокол Нидхэма – Шредера тоже использует оказии, но имеет другие слабые места. В связи с этим была сделана попытка учесть моменты, связанные с атаками задержки-воспроизведения, и исправить недостатки протокола Нидхэма – Шредера. Теперь протокол выглядит следующим образом.

1. A →B: $ID_A \| N_A$
2. B→ЦРК: $ID_B \| N_b \| E_{Kb}[ID_A \| N_A \| T_b]$
3. ЦРК →A: $E_{Ka} [ID_B \| N_A \| K_s \| T_b] \| E_{Kb} [ID_A \| K_s \| T_b] \| N_b$
4. A → B: $E_{Kb} [ID_A \| K_s \| T_b] \| E_{Ks} [N_b]$

Проследим шаг за шагом за теми действиями, которые необходимо выполнить при таком обмене.

1. Сторона **A** инициирует идентификационный процесс, генерируя оказию $\mathbf{N_A}$и посылая ее вместе со своим идентификатором стороне **B** в виде открытого текста. Эта оказия будет возвращена стороне **A** всообщении, включающем сеансовый ключ, чтобы сторона **A** могла убедиться в соответствии этого ключа текущему времени.

2. Сторона **B** сообщает ЦРК о том, что требуется сеансовый ключ. Соответствующее сообщение в ЦРК включает идентификатор **B** и оказию $\mathbf{N_A}$ — оказия будет возвращена **B** в шифрованном сообщении, включающем сеансовый ключ, чтобы у **B** имелась возможность убедиться в соответствии этого ключа текущему времени. Сообщение **B**, направляемое в ЦРК, также включает блок, зашифрованный с помощью секретного ключа, используемого совместно **B** и ЦРК. Этот блок служит для того, чтобы дать указание ЦРК выдать удостоверение стороне **A**, поэтому в блоке указывается предполагаемый получатель удостоверения, срок действия удостоверения, а также оказия, полученная от **A**.

3. ЦРК пересылает стороне **A** оказию **B** и блок, шифрованный с использованием секретного ключа, который **B** применяет совместно с ЦРК. Этот блок играет роль «мандата», который **A** может использовать для продолжения процесса идентификации, как будет видно далее. ЦРК также посылает стороне **A** блок, шифрованный с помощью секретного ключа, используемого совместно **A** и ЦРК. Этот блок убеждает сторону **A** в том, что сторона **B** получила начальное сообщение **A** (ввиду присутствия в блоке $\mathbf{ID_B}$) и что это сообщение соответствует времени и не является воспроизведением ($\mathbf{N_A}$), а также сообщает **A** сеансовый ключ ($\mathbf{K}_s$) и пределы времени его использования ($\mathbf{T}_b$).

4. Сторона **A** пересылает мандат стороне **B** вместе с оказией стороны **B**, шифрованной полученным сеансовым ключом. Мандат предоставляет участнику **B** секретный ключ, который служит для того, чтобы дешифровать $\mathbf{E_{Ks}[N_A]}$ и проверить значение оказии. Тот факт, что оказия участника **B** приходит в шифрованном с помощью сеансового ключа виде, убеждает его в том, что сообщение пришло от участника **A** и не является воспроизведением.

Этот протокол обеспечивает для обеих сторон (**A** и **B**) эффективный и безопасный способ начать сеанс обмена данными с использованием секретного сеансового ключа. Кроме того, протокол предоставляет стороне **A** ключ, который может служить и для последующей идентификации стороны **B** и позволит избежать необходимости устанавливать контакт с сервером аутентификации повторно.

Предположим, что участники **A** и **B** сначала устанавливают сеанс связи с помощью описанного выше протокола, а затем завершают этот сеанс. Позже, в рамках границ времени, указанных протоколом, сторона **A** намеревается начать новый сеанс связи со стороной **B**. Для этого выполняются действия в соответствии со следующим протоколом.

1. A →B: E_{Kb} [ID_A || K_s || T_b], N'_a
2. B →A: N'_b, E_{Ks} [N'_a]
3. A →B: E_{Ks} [N'_b]

Когда участник **B** получает сообщение на шаге 1, он проверяет, что мандат не просрочен. Новые оказии $\mathbf{N'}_a$ и $\mathbf{N'}_b$ убеждают обе стороны в том, данный процесс не является атакой воспроизведения.

В приведенном выше обсуждении время, представленное значением $\mathbf{T}_b$, является временем часов участника **B**. Таким образом, данная метка даты / времени не требует синхронизации часов, так как стороне **B** приходится проверять только метки даты / времени, сгенерированные в своей системе.

Подходы на основе шифрования с открытым ключом. В гл. 11 был представлен подход, позволяющий использовать схемы шифрования с открытым ключом для распределения сеансовых ключей. Соответствующий протокол предполагает, что в распоряжении каждой из двух сторон имеется текущий открытый ключ другой стороны. Но на практике требование выполнения этого условия может оказаться неудобным.

В связи с этим предлагается следующий протокол, использующий метки даты / времени, когда центральная система выступает в качестве сервера аутентификации (AS), поскольку не несет ответственности за распределение секретных ключей (скорее, AS обеспечивает сертификацию открытых ключей).

1. A →AS: $ID_A||ID_B$
2. AS →A: E_{KRas} [$ID_A||KU_A$ ||T]||E_{KRas}[$ID_B||KU_B||T$]
3. A→B: $E_{KRas}[ID_A||KU_A||T]||$ $E_{KRas}[ID_B||KU_B||T]$ || $E_{KUb}[E_{KRa}[K_s||T]]$

Сеансовый ключ выбирается и шифруется стороной **A**, поэтому нет риска его разглашения со стороны системы AS. Метки даты / времени защищают от воспроизведения скомпрометированных ключей.

Этот протокол компактен, но, как и предыдущие, требует синхронизации часов. Другой подход, который предложили By (*Woo*) и Лэм (*Lam*), предполагает применение оказий. Соответствующий протокол состоит из следующих шагов:

1. A →ЦРК: ID_A || ID_B
2. ЦРК →A: $E_{KRauth}[ID_B$ || $KU_b]$
3. A→B: $E_{KUb}[N_A$ || $ID_A]$
4. B→ЦРК: ID_B || ID_A || $E_{KUauth}[N_A]$
5. ЦРК →B: $E_{KRauth}[ID_A$ || $KU_a]$ || E_{KUb} [E_{KRauth} [N_A || K_s || ID_B]]
6. B →A: $E_{KUa}[E_{KRauth}[N_A$ || K_s || $ID_B]$ || $N_b]$
7. A→B: $E_{Ks}[N_b]$

На шаге 1 сторона **A** информирует ЦРК о своем намерении установить безопасное соединение со стороной **B**. ЦРК возвращает участнику **A** экземпляр сертификата открытого ключа стороны **B** (шаг 2). Используя открытый ключ участника обмена информацией **B**, сторона **A** информирует сторону **B** о намерении установить соединение и посылает оказию N_A (шаг 3). На шаге 4 сторона **B** спрашивает у ЦРК сертификат открытого ключа стороны **A** и сеансовый ключ. Соответствующее сообщение участника **B** включает оказию участника **A**, чтобы ЦРК мог пометить выдаваемый сеансовый ключ этой оказией. Оказия защищается шифрованием с использованием открытого ключа ЦРК. На шаге 5 ЦРК возвращает стороне **B** экземпляр сертификата открытого ключа стороны **A** и информацию $\{N_A, K_s, ID_B\}$. Эта информация, по существу, доказывает, что K_s является секретным ключом, сгенерированным ЦРК от имени стороны **B** и связанным с N_A, причем связывание K_s с N_A призвано убедить сторону **A** в том, что ключ K_s является новым. Эта тройка значений шифруется с помощью личного ключа ЦРК, чтобы сторона **B** могла проверить, что указанные значения на самом деле получены от ЦРК. Затем все это шифруется с помощью открытого ключа **B**, чтобы никто другой не мог использовать эти значения в попытке создания незаконного соединения с **A**. На шаге 6 тройка значений $\{N_A, K_s, ID_B\}$, оставаясь в виде, зашифрованном с помощью личного ключа ЦРК, передается стороне **A** вместе с оказией N_b, сгенерированной стороной **B**. Все это вместе шифруется теперь с использованием открытого ключа **A**. Сторона **A** извлекает из сообщения сеансовый ключ K_s и использует его для того, чтобы зашифровать N_b и вернуть соответствующее значение стороне **B**. Это убеждает сторону **B** в том, что сторона **A** действительно получила сеансовый ключ K_s.

Описанный протокол кажется вполне защищенным в отношении атак самого разного вида, однако следующая, исправленная, версия алгоритма оказывается еще более защищенной.

1. $A \rightarrow$ЦРК: $ID_A \| ID_B$
2. ЦРК $\rightarrow A$: $E_{KRauth}[ID_B \| KU_b]$
3. $A \rightarrow B$: $E_{KUb}[N_A \| ID_A]$
4. $B \rightarrow$ЦРК: $ID_B \| ID_A \| E_{KUauth}[N_A]$
5. ЦРК $\rightarrow B$: $E_{KRauth}[ID_A \| KU_a] \| E_{KUb}[E_{KRauth}[N_A \| K_s \| ID_A \| ID_B]]$
6. $B \rightarrow A$: $E_{KUa}[E_{KRauth}[N_A \| K_s \| ID_A \| ID_B] \| N_b]$
7. $A \rightarrow B$: $E_{Ks}[N_b]$

Здесь к множеству элементов, шифруемых личным ключом ЦРК на шагах 5 и 6, добавляется идентификатор ID_A стороны **A**. Это связывает сеансовый ключ K_s с идентификаторами обеих сторон, которые будут участвовать в сеансе обмена данными. Включение в набор

значения ID_A объясняется тем, что значение оказии N_A должно быть уникальным только для оказий, генерируемых стороной **A**, а не для всех оказий, генерируемых любой из сторон. Таким образом, именно пара $\{ID_A, N_A\}$ уникальным образом идентифицирует запрос на соединение со стороны **A**.

Как в случае этого протокола, так и в случаях протоколов, описанных выше, первоначальные версии протоколов в дальнейшем подвергались ревизии и после дополнительного анализа появлялись их исправленные версии. Это указывает на то, что в области аутентификации весьма непросто добиться приемлемого уровня надежности с первой попытки.

14.2. ОДНОСТОРОННЯЯ АУТЕНТИФИКАЦИЯ

Одним из приложений, для которых популярность использования шифрования постоянно растет, является электронная почта. Сама природа электронной почты и ее главное преимущество состоит в том, что для отправителя и получателя сообщений совсем не требуется быть в сети одновременно. Сообщение электронной почты направляется в электронный почтовый ящик адресата, где оно будет храниться до тех пор, пока у получателя не появится возможность прочесть его.

«Конверт», или заголовок сообщения электронной почты должен оставаться в незашифрованном виде, чтобы сообщение могло быть обработано протоколом передачи с промежуточным хранением электронной почты, например, протоколом *SMTP* (*Simple Mail Transfer Protocol* — простой протокол передачи почты). Однако во многих ситуациях желательно, чтобы протокол обработки почты не требовал доступа к сообщению в виде открытого текста, поскольку при этом необходимо обеспечить доверие к механизму обработки почты. Соответственно, сообщения электронной почты должны шифроваться так, чтобы у системы обработки почты не было необходимости иметь ключ дешифрования.

Вторым требованием является требование аутентификации. Обычно получатель стремится получить некоторые гарантии того, что сообщение пришло из указанного источника.

Подход на основе традиционного шифрования. При традиционном шифровании сценарий децентрализованного распределения ключей, представленный в гл. 6 на рис.6.5, оказывается непрактичным. Соответствующая схема требует, чтобы отправитель выслал запрос предполагаемому адресату, дождался ответа, который будет включать сеансовый ключ, и только потом посылал сообщение.

С некоторыми уточнениями стратегия использования ЦРК вполне может подойти в качестве схемы шифрованного обмена электронной

почтой. Ввиду стремления избежать необходимости интерактивного режима для получателя **B**, в то время, когда отправитель **A** посылает ему сообщение, шаги 4 и 5 следует исключить. Для сообщения **M** последовательность необходимых шагов будет иметь следующий вид:

1. A →ЦРК: $ID_A \| ID_B \| N_1$
2. ЦРК →A: $E_{Ka}[K_s \| ID_B \| N_1 \| E_{Kb}[K_s \| ID_A]]$
3. A →B: $E_{Kb}[K_s \| ID_A] \| E_{Ks}[M]$

Этот подход гарантирует, что только предполагаемый получатель сообщения может прочитать его. Обеспечивается также некоторая степень гарантии, что отправителем является **A**. Протокол не защищает от возможности воспроизведения сообщений. В определенной мере защита от этого могла бы быть обеспечена с помощью включения в сообщение метки даты / времени. Однако из-за потенциально возможных задержек в процессе передачи электронной почты польза от таких меток даты / времени оказывается весьма условной.

Подходы на основе шифрования с открытым ключом. Подходы на основе шифрования с открытым ключом неплохо зарекомендовали себя применительно к передаче электронной почты, включая подход, при котором предполагается непосредственное шифрование всего полного сообщения с целью обеспечения конфиденциальности, аутентификации или обеих этих задач. При таком подходе требуется либо чтобы отправитель знал открытый ключ получателя (конфиденциальность), либо чтобы получатель знал открытый ключ отправителя (аутентификация), либо чтобы было выполнено и то, и другое (конфиденциальность плюс аутентификация). Кроме того, необходимо одно- или двукратное применение алгоритма шифрования с открытым ключом к сообщению, которое может оказаться достаточно длинным.

Если главной задачей является конфиденциальность, то наиболее эффективной может быть следующая процедура:

$$\mathbf{A \rightarrow B:}\ E_{Kb}[K_S] \| E_{Ks}[M].$$

В этом случае сообщение шифруется одноразовым секретным ключом. Сторона **A** также шифрует одноразовый ключ открытым ключом стороны **B**. Только **B** сможет использовать соответствующий личный ключ, чтобы восстановить значение одноразового ключа, а затем применить этот ключ для дешифрования сообщения. Эта схема более эффективна, чем простое шифрование всего сообщения открытым ключом стороны **B**.

Если главной задачей является аутентификация, то может оказаться вполне достаточно цифровой подписи:

$$\mathbf{A \rightarrow B:}\ M \| E_{KRa}[H(M)].$$

Описанный метод гарантирует, что сторона **A** не сможет впоследствии отказаться от факта отправки сообщения. Однако этот метод не страхует от других видов обмана.

Например, Боб компонует сообщение своему боссу Элису, содержащее идею, которая позволит сэкономить деньги компании. Он добавляет к сообщению свою цифровую подпись и посылает это сообщение по электронной почте. В результате сообщение доставляется в почтовый ящик Элиса. Но предположим, что об идее Боба прослышал Макс, который имеет доступ к очереди почтовых сообщений, куда последние попадают перед доставкой. Макс находит сообщение Боба, убирает его подпись, ставит свою и снова помещает сообщение в очередь на доставку Элису. Макс получает вознаграждение за идею Боба.

Чтобы противодействовать такой схеме, как сообщение, так и подпись к нему можно зашифровать открытым ключом адресата:

$$\mathbf{A \rightarrow B:}\ E_{KUb}[M \| E_{KRa}[H(M)]].$$

Последние две схемы требуют, чтобы открытый ключ пользователя **A** был известен пользователю **B** и чтобы этот ключ не был просроченным. Эффективным способом обеспечения такой гарантии является цифровое удостоверение. Теперь имеем:

$$\mathbf{A \rightarrow B:}\ M \| E_{KRa}[H(M)] \| E_{KRas}[T \| ID_A \| KU_a]].$$

В дополнение к сообщению сторона **A** посылает стороне **B** подпись, шифрованную личным ключом **A**, и сертификат **A**, шифрованный личным ключом сервера аутентификации. Получатель сообщения сначала использует сертификат, чтобы получить открытый ключ отправителя и проверить аутентичность источника, а затем — открытый ключ, чтобы проверить само сообщение. Если требуется конфиденциальность, то все сообщение можно зашифровать с помощью открытого ключа стороны **B**.

Другим вариантом является шифрование всего сообщения с помощью одноразового секретного ключа. При этом секретный ключ тоже должен пересылаться, но в виде, зашифрованном с использованием открытого ключа **B**.

ГЛАВА 15. ИМИТОСТОЙКОСТЬ И ПОМЕХОУСТОЙЧИВОСТЬ КРИПТОСИСТЕМ

15.1. ОСНОВНЫЕ ПРИНЦИПЫ ИМИТОЗАЩИТЫ И ПОМЕХОУСТОЙЧИВОСТИ КРИПТОСИСТЕМ

В теории защиты информации рассматривается защита от двух классов воздействий — случайных и преднамеренных.

Защита информации, передаваемой по каналам связи, от случайных помех осуществляется с помощью ее помехоустойчивого кодирования. При таком кодировании в информацию вносится избыточность (добавляется контрольная сумма, вычисленная по определенному алгоритму) и на приемном конце с использованием этой избыточности производится обнаружение и / или исправление ошибок, внесенных в сообщение при его передаче.

В настоящее время разработано большое число методов помехоустойчивого кодирования, наиболее популярными среди которых являются следующие:

- кодирование с контролем двоичного сообщения на четность (числа единиц в сообщении) или на нечетность;
- кодирование с обеспечением постоянного веса двоичного сообщения (постоянного числа единиц в сообщении);
- корреляционное (парафазное) кодирование («1» и «0» кодируются парой символов, например, «10» и «01» соответственно);
- кодирование на основе линейных уравнений в полях Галуа и т.д.

Однако с помощью помехоустойчивого кодирования трудно обеспечить защиту от преднамеренных воздействий на сообщения, так как алгоритмы помехоустойчивого кодирования являются открытыми и известны злоумышленнику. В этом случае он может модифицировать сообщение, затем вновь вычислить контрольную сумму, а затем передать измененное сообщение получателю. Он также может навязывать ложную информацию, создавая собственные сообщения, кодируя их помехоустойчивым кодом и передавая их в канал связи.

Защита канала шифрованной связи от навязывания ложной информации носит название **имитозащиты**.

Для обеспечения имитозащиты необходимо, чтобы злоумышленник не имел возможности создавать правильные сообщения (т.е. те, которые на приемном конце канала будут восприняты как правильные).

Более формально — вероятность случайно выбранного сообщения пройти проверку на подлинность не должна превышать некоторую заданную величину.

Это возможно путем внесения избыточности в сообщение, подобно тому как это делается в случае защиты от случайных помех.

15.2. СТРУКТУРА ИМИТОЗАЩИЩЕННОГО ПОМЕХОУСТОЙЧИВОГО КАНАЛА СВЯЗИ

Структура простейшего помехозащищенного криптоканала представлена на рис. 15.1. Здесь Γ_1 и Γ_2 — синхронные генераторы ключей $k(t)$; **Ш** — криптографический шифратор; **К** — помехоустойчивый кодер; **ДК** — помехоустойчивый декодер; **ДШ** — криптографический дешифратор; x — открытый текст; y — шифрованное сообщение; ε — ложная информация и (или) помеха. В данной схеме ключ шифрования меняется от сообщения к сообщению. Это не дает возможности злоумышленнику расшифровывать информацию, поскольку схема работы генератора ключей (и, следовательно, последовательность ключей) является секретным параметром.

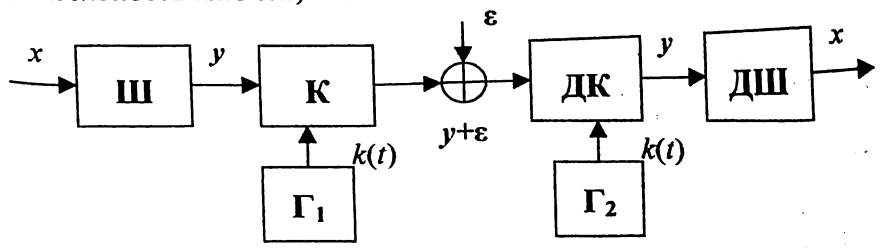


Рис. 15.1. Простейший помехозащищенный криптоканал

Однако защита, обеспечиваемая по такой схеме, не является полной, поскольку она позволяет злоумышленнику подменять сообщения. В этом случае у принимающей стороны не будет возможности определить, не пользуясь дополнительной информацией (например, избыточностью сообщения), является ли сообщение ложным или истинным. Данную проблему можно решить, введя в сообщение дополнительную избыточность перед шифрованием (рис. 15.2).

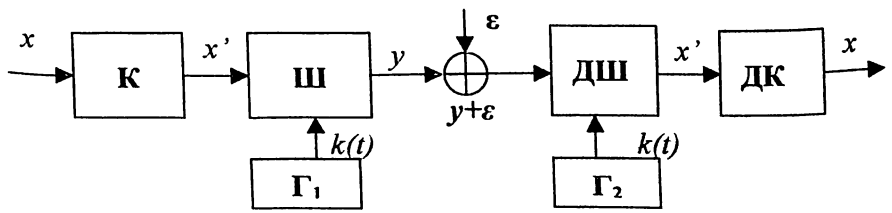


Рис. 15.2. Защита от подмены сообщений

Теперь перед шифрованием сообщение кодируется помехоустойчивым кодом, а получатель после дешифрования проверяет целостность сообщения с помощью декодера. Если злоумышленнику неизвестен алгоритм шифрования (или помехоустойчивого кодирования), он не сможет подменить сообщение, так как вероятность того, что расшифрованное ложное сообщение будет правильным кодовым словом помехоустойчивого кода, мала, и получатель сможет легко обнаружить подмену.

При этом следует иметь в виду, что если алгоритмы шифрования и помехоустойчивого кодирования не являются коммутативными по отношению к искажениям кодовых последовательностей за счет помех в канале связи, то для помехоустойчивости данной схемы необходимо введение еще одного помехоустойчивого кодера после криптографического шифратора и помехоустойчивого декодера перед криптографическим дешифратором.

В рассмотренных выше схемах нерешенной остается проблема согласования работы генераторов ключей Γ_1 и Γ_2. Если одно или несколько сообщений будут пропущены, то генераторы на передающей и приемной сторонах окажутся в разных состояниях, и прием сообщений станет невозможным. Необходимо предусмотреть средство для синхронизации работы генераторов ключей. Заметим, что передача постоянной синхронизирующей информации для этой цели не подходит, так как злоумышленник может определить ее структуру и в дальнейшем помешать процессу синхронизации.

Предлагаемая схема (рис. 15.3) дает возможность обеспечить синхронизацию генераторов ключей, не передавая никакой дополнительной информации, а также обнаруживать случайные ошибки, вносимые в сообщения в процессе их передачи.

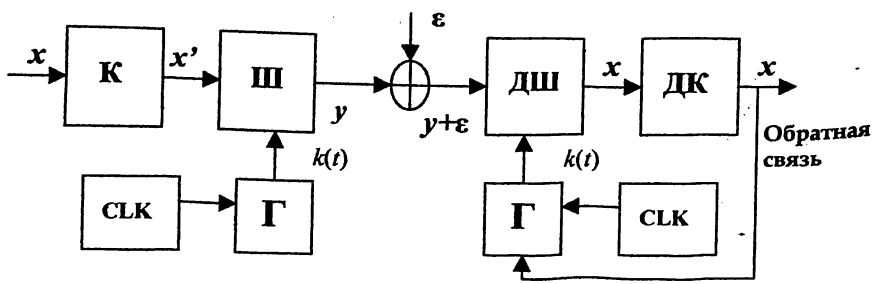


Рис. 15.3. Синхронизация генераторов ключей

Существо метода заключается в следующем: на приемной и передающей сторонах на вход генераторов ключей подаются сигналы часов **CLK**, и генерация ключей производится в зависимости от их

показаний. Поскольку часы работают автономно, возможно рассогласование их показаний на некоторую ограниченную сверху величину, определяемую конструкцией часов.

Если обозначить максимальную величину разности показаний часов через Δ, а величину дискретности показаний часов — через δ, то возможно рассогласование состояний генераторов ключей максимально на $l = [\Delta/\delta]$ позиций.

На приемной стороне после получения очередного сообщения производится его дешифрование с ключами, определяемыми состояниями генератора, отстающими от текущего не более чем на l позиций, а также производится декодирование помехоустойчивого кода. Поскольку в случае, если сообщение будет расшифровано с неверным ключом, вероятность того, что результат окажется правильным кодовым словом, мала, можно будет определить точное значение ключа и величину поправки к показаниям часов. Если в процессе передачи в сообщение были внесены искажения, то это приведет к тому, что после дешифрования не будет получено правильное кодовое слово ни при каком значении ключа из указанного выше диапазона. Данная ситуация будет свидетельствовать о наличии ошибок в сообщении, оно будет отвергнуто, а попытка синхронизации будет осуществлена при приеме следующего сообщения.

15.3. ИМИТОЗАЩИТА НА ОСНОВЕ РЕЖИМА ВЫРАБОТКИ ИМИТОВСТАВКИ

Обычно процедура имитозащиты строится на основе некоторой криптографической системы: к сообщению добавляется отрезок информации фиксированной длины, вычисленный по определенному правилу на основе открытых данных и ключа, называемый **имитовставкой**, или **кодом аутентификации сообщения** (*MAC — Message Authentication Code*).

В открытые данные, используемые для выработки имитовставки может быть включена, помимо собственно текста сообщения, и служебная информация, такая как дата и время отправки сообщения, регистрационный номер сообщения и т.п. В этом случае можно обеспечить также защиту от повторной передачи ранее переданного правильного сообщения.

Обеспечение имитозащиты по такой схеме регламентируется ГОСТ 28147–89. Стандартом предусмотрен режим выработки имитовставки, схема которого приведена на рис. 15.4. При выработке имитовставки используются 16 циклов шифрования в режиме простой замены. Выработка имитовставки производится с использованием той же ключевой информации, что применяется для шифрова-

ния исходного текста. Имитовставка вырабатывается с учетом блоков открытой информации (служебная информация, синхропосылка и т.п.), которая может не зашифровываться. При этом вся входная открытая информация разбивается на 64-битовые блоки, которые перед выработкой имитовставки делятся на 32-битовые блоки N_1, N_2. (Если последний 64-битовый блок неполон, то он дополняется нулями).

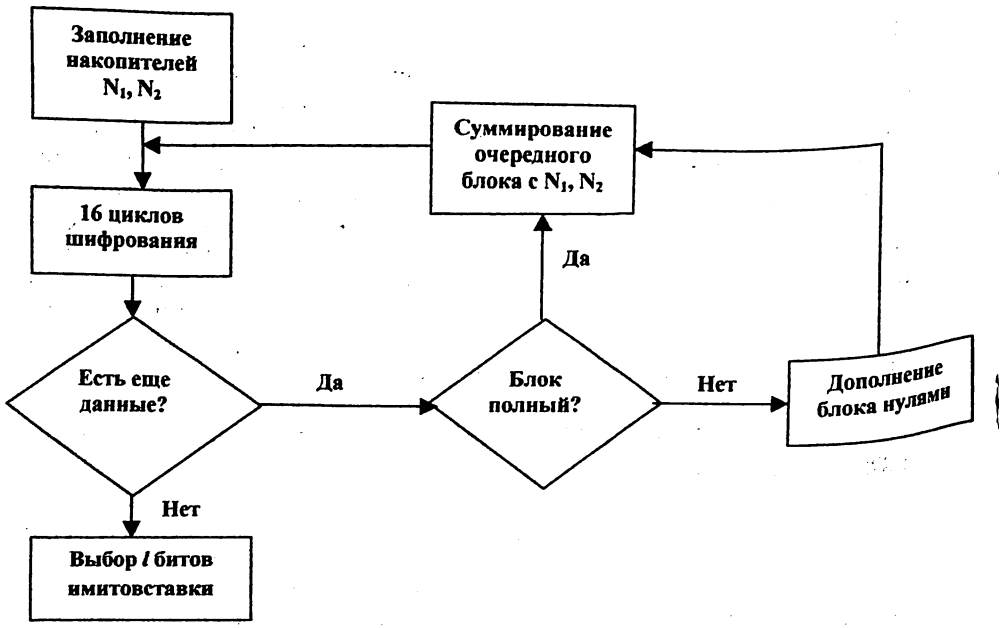


Рис. 15.4. Режим выработки имитовставки

На приемном конце имитовставка вырабатывается аналогично — после дешифрования информации, и вычисленное значение сравнивается с полученным по каналу связи. В случае совпадения сообщение считается истинным, в противном случае — ложным.

ГЛАВА 16. МЕТОДЫ ГЕНЕРАЦИИ СЛУЧАЙНЫХ ЧИСЕЛ

16.1. ТРЕБОВАНИЯ К СЛУЧАЙНЫМ ЧИСЛОВЫМ ПОСЛЕДОВАТЕЛЬНОСТЯМ. ФИЗИЧЕСКИЕ ИСТОЧНИКИ СЛУЧАЙНЫХ ЧИСЕЛ

Использование случайных чисел. Ряд алгоритмов защиты сети, основанных на средствах криптографии, предполагает использование случайных чисел. К таким алгоритмам относятся, в частности, следующие.

1. Схемы взаимной идентификации, рассмотренные в гл. 6. Сценарии распределения ключей (рис. 6.4, 6.5) в процессе установления соединения используют оказии, чтобы исключить возможность атаки на основе воспроизведения сообщений. Использование случайных чисел для оказий не дает шанса оппоненту определить или угадать значение оказии.

2. Генерирование сеансовых ключей, выполняемое либо центром распределения ключей, либо одним из участников соединения.

3. Генерирование ключей для алгоритма RSA-шифрования с открытым ключом.

Из этих приложений вытекает два четких и необязательно сочетаемых требования к используемой последовательности случайных чисел: случайность и непредсказуемость.

Случайность. Традиционно при генерировании последовательности якобы случайных чисел требуется, чтобы последовательность получаемых чисел была случайной в некотором вполне определенном статистическом смысле. Для проверки любой последовательности на случайность обычно служат два критерия:

– однородность распределения: распределение чисел в последовательности должно быть однородным, т.е. частота появления в последовательности конкретного значения должна быть примерно одинаковой для всех значений;

– независимость: ни одно из значений последовательности не должно логически выводиться из других значений.

Существуют вполне четкие алгоритмы для проверки того, что некоторая последовательность чисел соответствует заданному распределению, но вот алгоритма, позволяющего «доказать» независимость, нет. Здесь применяется ряд тестов, позволяющих лишь продемонстрировать, что последовательность не является независимой. Общая стратегия состоит в применении таких тестов до тех

пор, пока убеждение в независимости последовательности не станет достаточно правдоподобным.

Использование последовательности чисел, кажущихся статистически случайными, часто вытекает из криптографической структуры самого алгоритма шифрования.

Например, одним из главных требований схемы RSA-шифрования с открытым ключом является возможность генерирования простых чисел. При этом совсем не просто определить, является ли данное достаточно большое число N простым. Непосредственная проверка предполагает деление числа N на каждое целое число, меньшее $\sqrt{N}$. Если N оказывается порядка, скажем, 10^{150}, что не является нереальным для современных методов криптографии с открытым ключом, то такая проверка оказывается сегодня далеко за рамками аналитических знаний человека и вычислительных возможностей компьютера. Однако существует ряд эффективных алгоритмов проверки простоты числа с помощью использования последовательности случайно выбранных целых чисел в относительно простых вычислениях. Если такая последовательность достаточно длинная (но существенно меньше, чем $\sqrt{10^{150}}$), простота тестируемого числа может быть определена почти наверняка.

Такой подход, известный как рандомизация, служит для создания многих алгоритмов. По существу, если проблема чрезмерно сложна или требует очень много времени для точного решения, то для нахождения решения с любой требуемой долей уверенности применяется более простой и более быстрый способ, основанный на рандомизации.

Непредсказуемость. В приложениях типа взаимной идентификации или генерирования сеансовых ключей требование статистической случайности последовательности чисел оказывается не настолько важным, насколько требование непредсказуемости элементов последовательности. В «истинно» случайной последовательности каждое число статистически независимо от других чисел последовательности и, таким образом, непредсказуемо. Однако истинно случайные числа используются очень редко, чаще применяются последовательности чисел, которые выглядят случайными, но на самом деле генерируются с помощью некоторого алгоритма. В этом случае приходится заботиться о том, чтобы противник не имел возможности предсказать последующие элементы последовательности на основе предыдущих.

Физические источники случайных чисел. Нельзя сказать, что источники истинно случайных чисел являются очень распространенными устройствами. Потенциально такими источниками могут быть физические генераторы шумов, такие как импульсные детекторы ионизирующего излучения, газоразрядные лампы и конденсаторы с утечкой тока. Однако такие устройства могут найти весьма ограниченное применение в приложениях защиты сети. Здесь имеются проблемы как со случайностью, так и с точностью получаемых при этом чисел, не говоря уже о проблемах подключения такого рода устройств к каждой системе в сети. (Хотя известны случаи удачных применений их в генерации ключей, например, в российском криптографическом устройстве «Криптон»). Альтернативой может быть использование какого-нибудь из проверенных и опубликованных наборов случайных чисел. Однако такие наборы предлагают весьма ограниченный источник чисел по сравнению с потенциальными требованиями приложений защиты большой сети. К тому же хотя числа из подобных наборов действительно демонстрируют статистическую случайность, они вполне предсказуемы, так как оппонент, который знает, какой набор принят за основу, просто может взять его копию.

Поэтому криптографические приложения обычно используют алгоритмические методы генерирования случайных чисел. Соответствующие алгоритмы являются детерминированными и поэтому порождают последовательности чисел, которые статистически не случайны. Однако если алгоритм достаточно хорош, порождаемые им последовательности чисел выдерживают многие разумные тесты на случайность. Такие числа часто называют **псевдослучайными.**

16.2. ГЕНЕРАТОРЫ ПСЕВДОСЛУЧАЙНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ

Основная проблема классической криптографии долгое время заключалась в трудности генерирования непредсказуемых двоичных последовательностей большой длины с применением короткого случайного ключа. Существенный прогресс в разработке и анализе этих генераторов был достигнут лишь к началу шестидесятых годов.

Получаемые программно из ключа случайные или псевдослучайные ряды чисел называются на жаргоне отечественных криптографов ***гаммой***, по названию — буквы γ греческого алфавита, которой в математических записях обозначаются случайные величины.

Интересно отметить, что в книге «Незнакомцы на мосту», написанной адвокатом разведчика Абеля, приводится термин «гамма», который специалисты ЦРУ пометили комментарием «музыкальное упражнение?», т.е. в пятидесятые годы они не знали его смысла.

Следует при этом подчеркнуть, что заслуга конструирования длинных псевдослучайных рядов с «хорошими» статистическими свойствами полностью принадлежит криптографии.

Не следует думать, что они нужны лишь криптографам — картографирование Венеры стало возможным, когда длина периода случайного ряда импульсов превысила 10^{30}. Фотографирование этой планеты нельзя было сделать потому, что она всегда закрыта плотным слоем облаков. Локация же ее с Земли затруднена обилием помех и высокими требованиями к разрешению. Поэтому зондирование выполнялось случайной последовательностью импульсов указанного периода. После 300 зондирований, на что ушло более полугода, была получена карта, где различимы объекты размером около километра, а по высоте разрешение получено такое, какое достигнуто не везде на Земле. Генераторы псевдослучайных чисел используются очень широко в сотнях программных приложений — от конструирования ядерных реакторов и радиолокационных систем раннего обнаружения до поисков нефти и до многоканальной связи.

Можно сформировать три основных общих требования, которым должны удовлетворять криптографически стойкие генераторы псевдослучайных последовательностей и получаемые с их помощью гаммы:

1. Период гаммы должен быть достаточно большим для шифрования сообщений различной длины.

2. Гамма должна быть трудно предсказуемой. Это значит, что если известны тип генератора и кусок гаммы, то невозможно предсказать следующий за этим куском бит гаммы или предшествующий этому куску бит гаммы.

3. Генерирование гаммы не должно быть связано с большими техническими и организационными трудностями.

Самая важная характеристика генератора псевдослучайных чисел — это информационная длина его периода, после которого числа будут либо просто повторяться, либо их можно будет предсказать. Эта длина практически определяет возможное число ключей криптосистемы. Чем эта длина больше, тем сложнее подобрать ключ.

Второе из указанных выше требований связано со следующей проблемой: на основании чего можно сделать заключение, что гамма конкретного генератора действительно является непредсказуемой? Пока в мире нет универсальных и практически достоверных критериев для проверки этого свойства (см. обсуждение этого вопроса выше). Чтобы гамма считалась случайной и непредсказуемой, как минимум, необходимо, чтобы ее период был очень большим, а различные комбинации бит определенной длины равномерно распределялись по всей ее длине. Это требование статистически можно толковать и как сложность закона генерации псевдослучайной последовательности чисел. Если по достаточно длинному отрезку этой последовательности нельзя ни статистически, ни аналитически определить этот закон генерации, то в принципе этим можно удовлетвориться.

И, наконец, третье требование должно гарантировать возможность практической реализации генераторов псевдослучайных последовательностей с учетом требуемого быстродействия и удобства практичного использования.

Проблема генерации псевдослучайных последовательностей существует уже третье столетие. Одним из первых было предложение получать их как значения дробной части многочлена первой степени:

$$Y(n) = \mathrm{Ent}\,(a \times n + b),\ a, b = \mathrm{const}.$$

Если n пробегает значения натурального ряда чисел, то поведение $Y(n)$ выглядит весьма хаотичным. Еще Карл Якоби доказал, что при рациональном коэффициенте a множество $\{Y(n)\}$ конечно, а при иррациональном — бесконечно и всюду плотно в интервале от 0 до 1. Для многочленов больших степеней такая задача была решена лишь в 1916 г. выдающимся математиком прошедшего века Германом Вейлем. Кроме того, он установил критерий равномерности распределения любой функции от натурального ряда чисел. Небезынтересно привести его в краткой формулировке.

КРИТЕРИЙ ВЕЙЛЯ. Чтобы ряд $X_1, X_2, X_3, \ldots$ был распределен равномерно в интервале от 0 до 1, необходимо и достаточно, чтобы для любой интегрируемой по Риману функции $f(x)$ было справедливо соотношение $\mathrm{P}\{f(\mathrm{M}(x)) = \mathrm{M}f(x)\} = 1$.

Это соотношение выражает свойство, называемое эргодичностью и заключающееся в том, что среднее по реализациям псевдослучайных чисел равно среднему по всему их множеству с вероятностью 1. Таким образом, Вейль доказал, что эргодичность гарантирует отсутствие экзотичности в поведении последовательности X_n.

Однако эти результаты далеки от практики получения псевдослучайных рядов чисел. Дело в том, что теорема Якоби относится к действительным числам x, которые не могут быть использованы при вычислениях, потому что иррациональные действительные числа требуют для своей записи бесконечное число знаков. Попытки замены настоящего иррационального числа его приближением на ЭВМ для генерации псевдослучайной последовательности опасны, так как получаемые последовательности оканчиваются циклами с коротким периодом.

Наиболее давний вычислительный способ генерации псевдослучайных чисел на ЭВМ принадлежит Джону фон Нейману и относится к 1946 г. Этот способ базируется на том, что каждое последующее случайное число образуется возведением предыдущего в квадрат с последующим отбрасыванием цифр с обоих концов. Способ Неймана оказался ненадежным и очень быстро от него отказались.

Завершают доказательство непригодности полиномиальных и других функциональных преобразований натурального ряда чисел для получения псевдослучайных последовательностей результаты Пуанкаре. В частности, он установил, что непрерывное отображение T области U(x) числового пространства в себя обязательно приводит к короткой цикличности траекторий T(x) для всюду плотного в U множества точек. Ряды чисел, созданные такими методами, отягощены периодичностями.

Генератор последовательности Фибоначчи. Интересные классы генераторов случайных чисел неоднократно предлагались многими специалистами по целочисленной арифметике. В частности, Джордж Марсалиа и Ариф Зейман предложили класс генераторов псевдослучайных последовательностей, основанный на использовании последовательностей Фибоначчи, — в этой последовательности каждый последующий член, за исключением первых двух ее членов, равен сумме двух предыдущих:

$$\{0, 1, 1, 2, 3, 5, 8, 13, 21, 34 \ldots\}.$$

Если эта последовательность применяется для начального заполнения массива большой длины, то, используя этот массив, можно создать генератор случайных чисел Фибоначчи с запаздыванием, где складываются не соседние, а удаленные числа. Марсалиа и Зейман предложили ввести в схему Фибоначчи «бит переноса», который может иметь начальное значение 0 или 1. Построенный на этой основе генератор «сложения с переносом» приобретает интересные свойства, на их основе можно создавать последовательности, период которых значительно больше, чем у применяемых в настоящее время конгруэнтных генераторов. По образному выражению Марсалиа, генераторы этого класса можно рассматривать как усилители случайности: «Вы берете случайное заполнение длиной в несколько тысяч бит и генерируете длинные последовательности случайных чисел».

Один из вариантов генератора последовательности Фибоначчи показан на рис. 16.1:

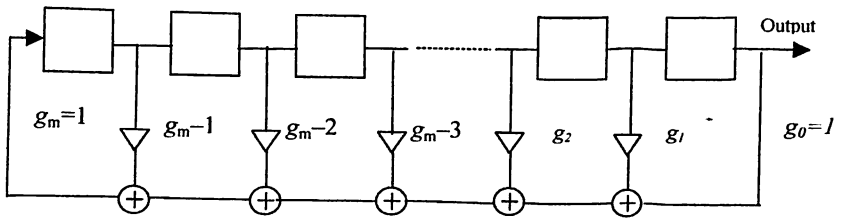


Рис. 16.1. Генератор последовательности Фибоначчи

Здесь квадратами обозначены разряды генератора, треугольниками — умножение на коэффициенты (на практике в зависимости от коэффициента там либо есть соединение с последующей логикой, либо его нет). Плюсы в кружках — это операция XOR: 0 + 0 = 0, 0 + 1 = 1, 1 + 1 = 0.

Рекуррентные двоичные последовательности. Линейные последовательности максимальной длины в основном получают с помощью N-генераторов, использующих в качестве основных элементов каскадные регистры сдвига и сумматоры по модулю 2 (**фильтр Хаффмана**). Генератор состоит из хорошо отработанных стандартных импульсных элементов и при минимальном их числе обеспечивает получение последовательности с максимальным периодом (**М-последовательность**). Достоинствами такого типа генератора являются фиксированная амплитуда, легко и в весьма широком диапазоне регулируемая ширина спектра сигнала, а также возможность путем небольших усложнений получать сдвинутые по шкале времени сигналы.

На рис. 16.2 представлена схема N-разрядного регистра сдвига, выходные сигналы которого после обработки при помощи специальной цифровой логической схемы снова вводятся в регистр, замыкая тем самым цепь рециркуляции. Этот регистр является базой для построения генератора псевдослучайных последовательностей. При этом необходимо выполнение следующих условий:

– должно быть задано правило подключения сумматоров (α_0, α_2, …, α_N);

– α_0 и α_N всегда равны 1 (поэтому на схеме их можно не указывать);

– из всех α_i, $i \in \{1, 2, \ldots, N-1\}$, хотя бы одно должно иметь значение ‘1’.

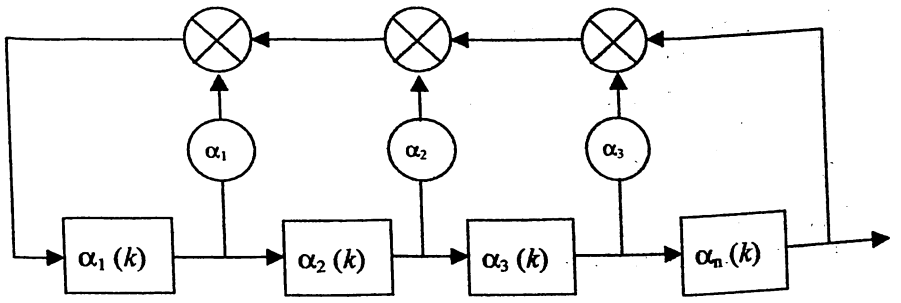


Рис. 16.2. Фильтр Хаффмана

Циклические свойства генератора последовательности определяются так называемым характеристическим полиномом

$$\varphi(x) = \sum_{k=0}^{N} \alpha_k x,^k$$

где $\alpha_0 = \alpha_N = 1$, $\alpha_j \in \{0,1\}$, $j = 1,2, \ldots, N-1$.

Период последовательности будет максимальным в том случае, когда многочлен $\varphi(x)$ удовлетворяет условиям примитивности и неприводимости. В нахождении такого многочлена заключается основная задача синтеза генератора псевдослучайных последовательностей максимальной длины (ГППМД).

ГППМД составляют основу так называемых генераторов псевдослучайных чисел. Для этого к регистру сдвига добавляются сумматоры по модулю 2. При работе регистра сдвига на выходе этих элементах образуются М-последовательности $\{a_{k+s}\}$, задержанные относительно исходной последовательности $\{a_k\}$, полученной на выходе цепи обратной связи, на определенное число тактов. Величину задержки можно регулировать в пределах от $N+1$ до $T = 2^N - 1$. При этом на выходе данного генератора получаются равномерно распределенные псевдослучайные числа.

Благодаря своему свойству цикличности, М-последовательности получили большое распространение в задачах помехоустойчивого кодирования, что дало возможность использования фильтров Хаффмана в качестве простых кодирующих и декодирующих устройств.

Метод линейного сравнения (конгруэнтные генераторы). Безусловно, самым популярным алгоритмом для генерирования псевдослучайных чисел является алгоритм, предложенный Д.Х. Лемером (*Lehmer*) и называемый методом линейного сравнения (конгруэнтным способом).

Этот алгоритм имеет четыре следующих параметра:

m	модуль сравнения	$m > 0$,
a	множитель	$0 \le a < m$,
c	приращение	$0 \le c < m$,
X_0	начальное или порождающее число	$0 \le X_0 < m$.

Последовательность случайных чисел $\{X\}$ получается с помощью итераций соотношения

$$X_{n+1} = (aX_n + c) \bmod m.$$

При этом если m, a, c и X_0 являются целыми, то будет получена последовательность целых чисел из диапазона $0 \le X_n < m$.

Выбор значений a, c и m оказывается очень важным в отношении разработки хорошего генератора случайных чисел. Выбор в

качестве a, c, m иррациональных чисел, требующих для своего представления бесконечного числа знаков, практически нереализуем. Выбор почти иррациональных чисел, представленных, например, 4 байтами в формате с плавающей запятой, дает псевдослучайные последовательности с периодами всего лишь 1225 и 147 в зависимости от начального заполнения. Исследования показали, что более эффективно вести вычисления в целых числах.

Для них, в частности, было показано, что при $c = 0$, $m = 2^n$ наибольший период составит $m/4$ при $a = 3 + 8j$ или $a = 5 + 8j$ и нечетном начальном числе. При этом в случае достаточно больших a последовательность производит впечатление случайной. Дальнейшие исследования показали, что если c нечетно, а $a = 1 + 4j$, то период можно увеличить до числа $m = 2^n$. После долгих поисков числа a исследователи остановились на значениях $a = 69\,069$ и $a = 71\,365$.

Желательно иметь m очень большим, чтобы потенциально могли генерироваться очень длинные серии различных случайных чисел. Общим правилом здесь является выбор значения m, близкого к максимально допустимому для данного компьютера неотрицательному целому числу. Поэтому довольно часто значение m выбирается равным или почти равным значению 2^{31}.

Кроме значений $m = 2^n$ широко используются выборы простых m, например, простого числа $m = 2^{31} - 1$, известного со времен Эйлера (это число «плохо» тем, что в двоичной записи содержит лишь единицы. Однако оно может использоваться, если вычисления выполняются в десятичной арифметике).

Для иллюстрации важности выбора «хороших» параметров алгоритма рассмотрим, например, случай $a = c = 1$. Порождаемая при этом последовательность, очевидно, не будет удовлетворительной. Теперь рассмотрим значения $a = 7$, $c = 0$, $m = 32$ и $X_0 = 1$. В этом случае генерируется последовательность {7, 17, 23, 1, 7, ...}, которая также, очевидно, не будет удовлетворительной. Из 32 возможных значений здесь оказываются задействованными только 4 (в данном случае говорят, что последовательность имеет период 4). Если же, оставив другие значения прежними, изменить значение a и положить $a = 5$, то результирующей последовательностью будет {1, 5, 25, 29, 17, 21, 9, 13, 1,...} и ее период будет равен уже 8.

При реализации псевдослучайных последовательностей в компьютере или в других цифровых устройствах предлагаются три следующих критерия (вытекающие из приведенных выше основных общих требований к криптографически стойким генераторам псевдослучайных последовательностей), по которым можно оценить качество любого генератора двоичных псевдослучайных чисел.

1. Генерирующая функция должна быть функцией полного периода, т.е. функция должна порождать все числа от 0 до m, прежде чем числа начнут повторяться.

2. Генерируемая последовательность должна вести себя как случайная. На самом деле эта последовательность не будет случайной, поскольку генерируется детерминированным алгоритмом, но существует множество статистических тестов, которые можно использовать, чтобы оценить степень случайности поведения последовательности.

3. Генерирующая функция должна эффективно реализовываться в рамках 32-битовой арифметики.

Все эти три критерия могут быть удовлетворены при подходящем выборе значений a, c и m. Относительно первого критерия можно доказать, что если m является простым и $c = 0$, то для определенных значений a период генерируемой функцией последовательности оказывается равным $m - 1$, и в этой последовательности будет отсутствовать только значение 0.

В 32-битовой арифметике удобным простым значением для m является значение $2^{31} - 1$ (см. его недостатки выше). В этом случае генерирующая функция принимает вид

$$X_{n+1} = (aX_n) \bmod (2^{31} - 1).$$

Из более чем двух миллионов возможных значений a только горстке множителей соответствуют функции, выдерживающие все три теста. Одним из них является значение $a = 7^5 = 16\,807$, которое было найдено и использовано для семейства компьютеров IBM/360. Соответствующий генератор находит очень широкое применение, и поэтому он был подвергнут более тщательному анализу, чем любой другой генератор псевдослучайных чисел. Он нередко рекомендуется для статистического и имитационного моделирования различных процессов.

Преимуществом алгоритма линейного сравнения является то, что если выбрать подходящие множитель и модуль сравнения, то генерируемая последовательность чисел оказывается статистически неотличимой от последовательности чисел, выбираемых случайно (но безвозвратно) из множества чисел 1, 2, ..., $m - 1$. Но в самом алгоритме нет ничего случайного вообще, кроме выбора начального значения X_0. Если это значение выбрано, остальные числа последовательности определяются им однозначно. Это оказывается очень важным с позиций криптоанализа.

Если противник знает, что используется алгоритм линейного сравнения, и если к тому же ему известны параметры алгоритма (например, $a = 7^5$, $c = 0$, $m = 2^{31} - 1$), то, открыв всего одно число, противник сможет получить и все последующие. Но даже если

оппонент знает только то, что выбран алгоритм линейного сравнения, знания небольшой части последовательности уже достаточно для того, чтобы определить все параметры алгоритма.

Предположим, например, что противник сможет определить значения X_0, X_1, X_2 и X_3. Тогда

$$X_1 = (aX_0 + c) \bmod m,$$
$$X_2 = (aX_1 + c) \bmod m,$$
$$X_3 = (aX_2 + c) \bmod m.$$

Эти уравнения могут быть решены относительно a, c и m.

Итак, хотя и удобно использовать хороший генератор псевдослучайных чисел, желательно позаботиться о том, чтобы генерируемая последовательность была в действительности невоспроизводимой, чтобы знание части последовательности не давало оппоненту возможности определить следующие элементы последовательности. Эта цель может быть достигнута целым рядом способов. Например, можно изменять поток случайных чисел, используя для этого системные часы. Один из способов на основе системных часов состоит в инициализации новой последовательности после получения каждых N чисел, используя для начального числа текущее значение времени (по mod m). А можно просто добавлять к каждому случайному числу текущее значение времени (по mod m).

16.3. КРИПТОГРАФИЧЕСКИ ГЕНЕРИРУЕМЫЕ ПСЕВДОСЛУЧАЙНЫЕ ЧИСЛА

Для криптографических приложений имеет смысл использовать уже имеющуюся логику шифрования для генерирования случайных чисел. На практике применяются различные подходы, и в этом разделе рассмотрим три из них.

Циклическое шифрование. На рис. 16.3 показана схема подхода. В данном случае с помощью главного ключа генерируются сеансовые ключи. Счетчик с периодом N обеспечивает вводимые параметры логики шифрования. Например, если требуется получить 56-битовые ключи DES, можно использовать счетчик с периодом, равным 2^{56}. После получения каждого ключа счетчик увеличивается на единицу. Поэтому получаемые таким образом псевдослучайные числа будут повторяться с соответствующим периодом: каждое из выходных чисел $X_0, X_1, \ldots, X_{N-1}$ будет определяться разными значениями счетчика, и поэтому $X_0 \neq X_1 \neq \ldots \neq X_{N-1}$. Ввиду того что главный ключ защищен, оказывается невозможным с помощью вычислений получить любой секретный ключ, зная один или несколько полученных ранее ключей.

Чтобы сделать алгоритм еще более защищенным, вместо значений простого счетчика в качестве вводимых значений можно использовать выходные значения некоторого полнопериодического генератора псевдослучайных чисел.

Генератор псевдослучайных чисел ANSI X9.17. Один из лучших (с позиций криптографии) генераторов псевдослучайных чисел определяется стандартом ANSI X9.17. Предложенный алгоритм используется целым рядом приложений, среди которых приложения, обеспечивающие безопасность финансовых платежей, и PGP.

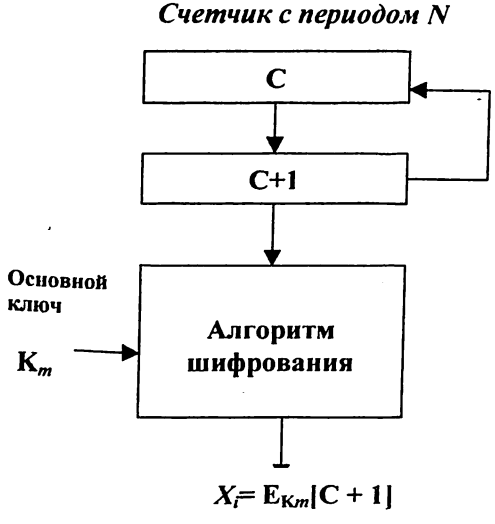


Рис. 16.3. Генерирование псевдослучайных чисел с использованием счетчика

На рис. 16.4 показана схема алгоритма, в котором для шифрования используется «тройной» DES. Этот алгоритм имеет следующие составляющие.

1. *Ввод.* На вход генератора подается два псевдослучайных значения. Одно из них является 64-битовым представлением текущих даты и времени и меняется для каждого нового генерируемого числа. Другое представляет собой 64-битовое начальное произвольное значение, которое обновляется в процессе вычислений.

2. *Ключи.* Генератор использует три модуля шифрования «тройного» DES. Каждый из этих трех модулей использует одну и ту же пару 56-битовых ключей, которые должны сохраняться в секрете и использоваться только для генерирования псевдослучайных чисел.

3. *Вывод.* Выводимыми значениями являются 64-битовое псевдослучайное число и 64-битовое начальное значение.

Определим следующие величины:

DT_i: значение даты / времени в начале i-го шага генерирования;

V_i: начальное значение для i-го шага генерирования;

R_i: псевдослучайное число, получаемое в результате i-го шага генерирования;

K_1, K_2: ключи DES, используемые на каждом шаге генерирования.

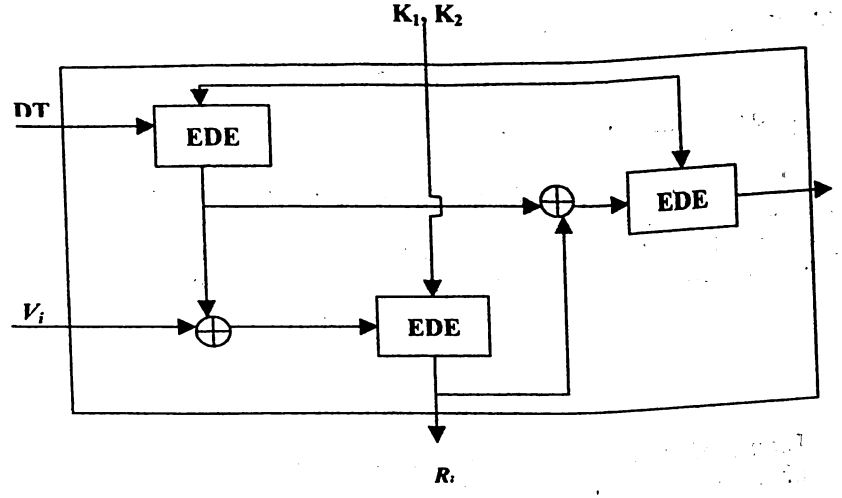


Рис. 16.4. Генератор псевдослучайных чисел ANSI X9.17

Тогда

$$R_i = \mathrm{EDE}_{K1,K2}[V_i \oplus \mathrm{EDE}_{K1,K2}[DT_i]],$$

$$V_{i+1} = \mathrm{EDE}_{K1,K2}[R_i \oplus \mathrm{EDE}_{K1,K2}[DT_i]],$$

где $\mathrm{EDE}_{K1,K2}$ означает последовательность «шифрования – дешифрования – шифрования» с использованием алгоритма «тройного» DES с двумя ключами.

Криптографическая надежность этого метода определяется несколькими факторами. Здесь используются 112-битовый ключ и три блока шифрования **EDE**, в сумме дающие девятикратное шифрование DES. Схема управляется двумя вводимыми псевдослучайными значениями: значением даты и времени и начальным значением, производимым генератором, но отличным от производимого генератором псевдослучайного значения. Таким образом, объем данных, которые должен анализировать противник, оказывается огромным. Даже если ему станет известным псевдослучайное значение R_i, из R_i невозможно будет вывести V_{i+1}, поскольку для генерирования V_{i+1} используется дополнительная операция **EDE**.

Генератор BBS. Один из популярных подходов к генерированию надежных последовательностей псевдослучайных чисел состоит в использовании генератора Блюма – Блюма – Шуба (*Blum — Blum, Shub, BBS*), названного так в честь его разработчиков. Доказательство криптографической надежности здесь является наиболее строгим из опубликованных. Заложенная в его алгоритме процедура выглядит следующим образом. Сначала выбираются два больших простых числа p и q, дающих при делении на 4 в остатке 3, т.е.

$$p \equiv q \equiv 3 \pmod 4.$$

Это означает, что $(p \bmod 4) \equiv (q \bmod 4) \equiv 3$.

Например, для простых чисел 7 и 11 как раз имеем $7 \equiv 11 \equiv 3 \pmod 4$.

Пусть теперь $n = p \times q$. Выберем случайное число s, взаимно простое с n — в данном случае это означает, что ни p, ни q не являются делителями s. Тогда генератор **BBS** порождает последовательность битов B_i в соответствии со следующим алгоритмом:

$$X_0 = s^2 \pmod n$$
$$\text{for } i = 1 \text{ to } \infty$$
$$X_i = (X_{i-1})^2 \pmod n$$
$$B_i = X_i \pmod 2$$

Таким образом, на каждой итерации выбирается *младший* бит. В табл. 16.1 показан пример последовательности, полученной в результате использования алгоритма *BBS*. Здесь n = 192 649 = 383 × × 503 и начальное значение s = 101 355.

Таблица 16.1

Применение генератора BBS

i	X_i	B_i	i	X_i	B_i
0	20 749		11	137 922	0
1	143 135	1	12	123 175	1
2	177 671	1	13	8630	0
3	97 048	0	14	114 386	0
4	89 992	0	15	14 863	1
5	174 051	1	16	133 015	1
6	80 649	1	17	106 065	1
7	45 663	1	18	45 870	0
8	69 442	0	19	137 171	1
9	186 894	0	20	48 060	0
10	177 046	0			

Генератор BBS обычно называют криптографически защищенным генератором псевдослучайных битов. Этот генератор является одним из генераторов, удовлетворяющих так называемому критерию следующего бита (*next-bittest*), который определяется так:

«Генератор псевдослучайных битов удовлетворяет критерию следующего бита, если не существует алгоритма с полиномиальной оценкой времени его выполнения, который по первым k битам выходной последовательности может предсказать ее $(k + 1)$-й бит с вероятностью, существенно большей, чем ½».

Другими словами, не должен существовать практически эффективный алгоритм, который позволил бы по первым k битам последовательности выяснить с вероятностью большей, чем ½, что следующий бит будет равен 1 (или 0). С позиций любого практического подхода последовательность будет непредсказуемой. Защищенность генератора BBS основана на трудности разложения значения n на множители, т.е. на трудности задачи нахождения простых множителей p и q по данному n.

ГЛАВА 17. КРИПТОГРАФИЧЕСКИЕ ШИФРАТОРЫ

В наши дни появились технологии, которые позволили максимально приблизить к широкому кругу пользователей средства гарантированной защиты информации и создать аппаратные персональные шифраторы, что до недавнего времени было невозможно из-за целого ряда проблем, таких как создание, хранение и распределение ключевой информации и т.п. В данной главе рассмотрены основные особенности, возможности использования и перспективы развития технологий аппаратных шифраторов.

Возможная классификация современных шифраторов приведена на рис. 17.1.

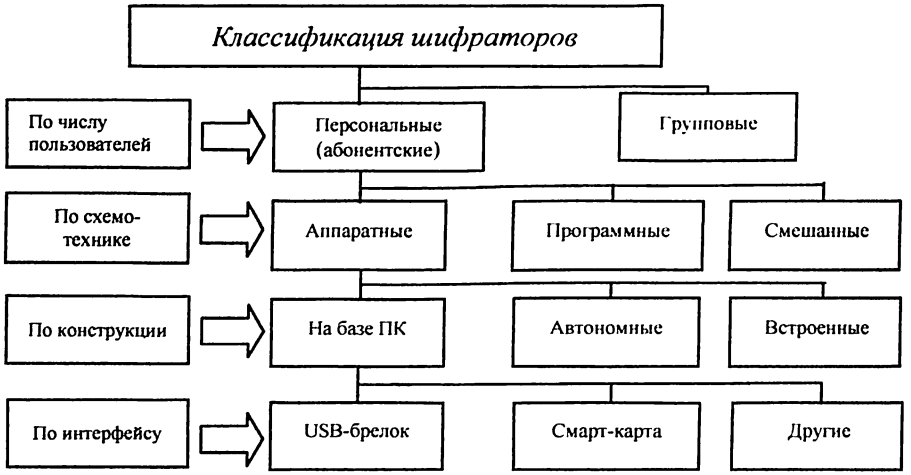


Рис. 17.1.Классификация современных шифраторов

17.1. ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ И СТРУКТУРА АППАРАТНОГО ШИФРАТОРА

Чаще всего аппаратный шифратор представляет собой плату расширения, вставляемую в разъем ISA или PCI системной платы ПК. Есть и другие варианты, например, в виде USB-ключа с криптографическими функциями, но сначала рассмотрим классический вариант — шифратор для шины PCI.

Функциональные возможности аппаратного шифратора. Использовать целую плату только для функций шифрования — непозволительная роскошь, поэтому производители аппаратных шифраторов стараются наполнить их различными дополнительными возможностями, среди которых следующие:

1. *Генерация случайных чисел.* Это необходимо прежде всего для получения криптографических ключей. Кроме того, многие алгоритмы защиты используют случайные числа и для других целей (например, алгоритм электронной подписи по ГОСТ Р 34.10–2001, которому при каждом вычислении подписи необходимо новое случайное число).

2. *Контроль входа на компьютер.* При включении ПК устройство требует от пользователя ввести персональную информацию (например, вставить дискету с ключами). Работа будет разрешена только после того как устройство опознает предъявленные ключи и сочтет их «своими». В противном случае придется разбирать системный блок и вынимать оттуда шифратор, чтобы загрузиться (как известно, информация на ПК тоже может быть зашифрована).

3. *Контроль целостности файлов операционной системы.* Это не позволит злоумышленнику в отсутствие истинного пользователя изменить какие-либо данные. Шифратор хранит в себе список всех важных файлов с заранее рассчитанными для каждого контрольными суммами (или хэш-значениями), и если при следующей загрузке не совпадет эталонная сумма хотя бы одного из них, компьютер будет блокирован.

Плата со всеми перечисленными возможностями называется **устройством криптографической защиты данных** — УКЗД.

Шифратор, выполняющий контроль входа на ПК и проверяющий целостность операционной системы, называют также «электронным замком». Ясно, что аналогия неполная — обычные замки существенно уступают этим интеллектуальным устройствам. Очевидно, что последним не обойтись без программного обеспечения — необходима утилита, с помощью которой формируются *ключи* для пользователей и ведется их список *для распознавания* «свой / чужой». Кроме того, требуется приложение для выбора важных файлов и расчета их контрольных сумм. Эти программы обычно доступны только администратору по безопасности, который должен предварительно настроить все УКЗД для пользователей, а в случае возникновения проблем разбираться в их причинах.

В случае постановки на компьютер УКЗД уже при следующей загрузке устройство проявится через несколько секунд после включения, как минимум, сообщив о себе и попросив ключи. Шифратор всегда перехватывает управление при загрузке ПК (когда BIOS компьютера поочередно опрашивает все функциональные узлы), после чего нелегко получить его обратно. УКЗД позволит продолжить за-

грузку только после всех своих проверок. Если ПК по какой-либо причине не отдаст управление шифратору, тот, немного подождав, все равно его заблокирует.

Структура аппаратного шифратора. Рассмотрим теперь, из чего должно состоять УКЗД, чтобы выполнять вышеперечисленные непростые функции (рис. 17.2):

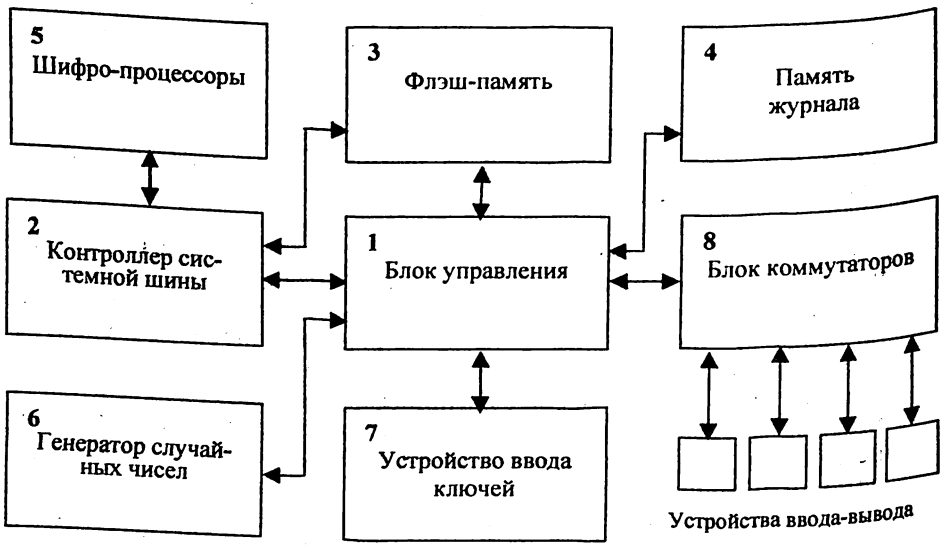


Рис. 17.2. Структура шифратора

1. *Блок управления* — основной модуль шифратора, который «заведует» работой всех остальных. Обычно реализуется на базе микроконтроллера, которых сейчас предлагается немало. Главное — быстродействие и достаточное количество внутренних ресурсов, а также внешних портов для подключения всех необходимых модулей.

2. *Контроллер системной шины* ПК (например, PCI). Через него осуществляется основной обмен данными между УКЗД и компьютером.

3. *Энергонезависимое запоминающее устройство* (ЗУ) — обычно на базе микросхем флэш-памяти. Оно должно быть достаточно емким (несколько мегабайт) и допускать большое число циклов записи. Здесь размещается программное обеспечение микроконтроллера, которое выполняется при инициализации устройства (т.е. когда шифратор перехватывает управление при загрузке компьютера).

4. *Память журнала*. Также представляет собой энергонезависимое ЗУ; это еще одна флэш-микросхема: во избежание возможных коллизий память для программ и память для журнала не должны объединяться.

5. *Шифропроцессор* (их может быть несколько) — это специализированная микросхема или микросхема программируемой логики PLD (Programmable Logic Device). Собственно, он и шифрует данные.

6. *Генератор случайных чисел*. Обычно представляет собой некое устройство, дающее статистически случайный и непредсказуемый сигнал — белый шум. Это может быть, например, шумовой диод. А перед использованием по специальным правилам белый шум преобразуется в цифровую форму.

7. *Блок ввода ключевой информации*. Обеспечивает защищенный прием ключей с ключевого носителя, через него также вводится идентификационная информация о пользователе, необходимая для решения вопроса «свой / чужой».

8. *Блок коммутаторов*. Помимо перечисленных основных функций, УКЗД может по указанию администратора безопасности отключать возможность работы с внешними устройствами: дисководами, CD-ROM, параллельным и последовательным портами, шиной USB и т.д. Если пользователь работает с настолько важной информацией, что ее нельзя ни печатать, ни копировать, УКЗД при входе на компьютер заблокирует все внешние устройства, включая даже сетевую карту.

Шифропроцессор. Шифрование в УКЗД должно выполняться так, чтобы посторонним невозможно было узнать ключи и каким-либо образом повлиять на реализуемые в нем алгоритмы. Иногда бывает полезно засекретить и правила преобразования ключей. Поэтому шифропроцессор состоит из нескольких структурных единиц (рис. 17.3).

1. *Вычислитель* — набор регистров, сумматоров, блоков подстановки и т.п., связанных между собой шинами передачи данных. Собственно, он и выполняет криптографические действия, причем должен делать это максимально быстро. На вход вычислитель получает открытые данные, которые следует зашифровать, и ключ шифрования, который, как известно, является случайным числом.

2. *Блок управления*. Это аппаратно реализованная программа, управляющая вычислителем. Изменение программы по какой-либо причине может привести к сбоям в работе шифропроцессора. Это чревато, например, появлением данных в открытом виде вместо зашифрованного (хотя это крайний случай; более вероятно получение такой шифровки, которую никто не расшифрует никогда). Поэтому программа должна не только надежно храниться и устойчиво функ-

ционировать, но и регулярно проверять сама себя. Для этого блок управления (описанный выше) тоже периодически посылает ей контрольные задачи. На практике для большей уверенности ставят два шифропроцессора, которые постоянно сравнивают свои результаты (если они не совпадают, шифрование повторяется). Все это требуется для обеспечения неизменности алгоритма шифрования.

3. *Буфер ввода-вывода* необходим для повышения производительности устройства: пока шифруется первый блок данных, загружается следующий, и т.д. То же самое происходит и на выходе. Такая конвейерная передача данных серьезно увеличивает скорость шифрования.

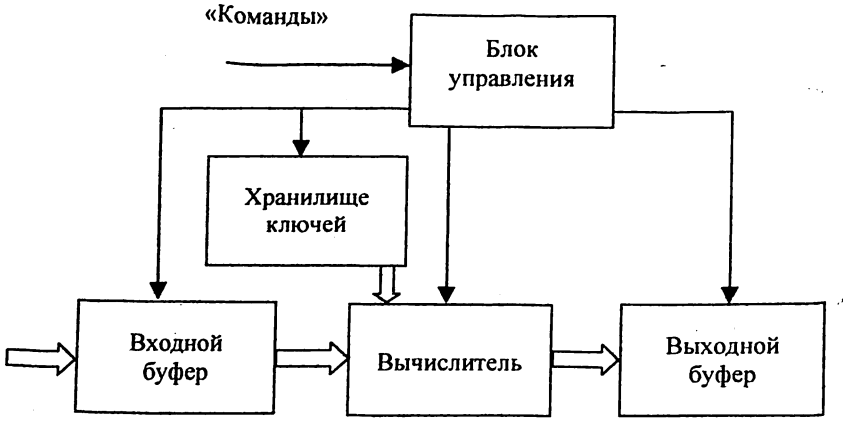


Рис. 17.3. Структура шифропроцессора

Быстродействие УКЗД. Современные УКЗД шифруют данные без помощи центрального процессора ПК. В шифратор лишь передается команда, а затем он сам извлекает данные из ОЗУ компьютера, шифрует их и отправляет в указанное место. Процессор же при этом вполне может выполнять другие задачи. Исследования современных УКЗД показывают, что во время их работы производительность ПК практически не снижается.

Возможно применение и нескольких УКЗД на одном компьютере, например, на криптографическом маршрутизаторе: один шифрует отправляемую в Интернет информацию, другой — принимаемую. Производительность такой системы не вносит задержек в работу локальной сети Fast Ethernet (100 Мбит/с).

Потоковая скорость обработки данных — это один из основных параметров, по которым оценивают аппаратные шифраторы. Она измеряется в мегабайтах в секунду и зависит прежде всего от сложности алгоритма шифрования. Проще всего оценить ее по формуле

$$V = F \times K / n,$$

где F — тактовая частота;

K — размер стандартного блока шифрования;

n — число тактов, требующееся на преобразование стандартного блока.

Например, отечественный алгоритм по ГОСТ 28147–89 имеет быстродействие 32 такта на 8-байтовый блок, а значит, теоретически скорость шифрования должна стремиться к 25 Мбайт/с при тактовой частоте 100 МГц. Однако последние опубликованные достижения скорости аппаратной реализации этого алгоритма — 9 Мбайт/с. Ограничения являются чисто технологическими и обусловлены отсутствием необходимого уровня разработок или элементной базы. Следует отметить, что программная реализация криптоГОСТа на самых современных ПК достигает 12–16 Мбайт/с при тактовой частоте процессора 1 ГГц. (Хотя в этом случае аппаратная скорость шифрования теоретически могла бы быть около 250 Мбайт/с).

17.2. ПРИНЦИП ДЕЙСТВИЯ АППАРАТНОГО ШИФРАТОРА

У аппаратных шифраторов существует два основных режима работы: режим начальной загрузки и режим выполнения операций.

Первый начинается при загрузке компьютера, в тот момент, когда *BIOS* ПК опрашивает все подключенные к нему внутренние и внешние устройства. В этот момент шифратор перехватывает управление и выполняет последовательность команд, «зашитую» в его память, предлагая пользователю прежде всего ввести главный ключ шифрования (т.е. вставить соответствующий ключевой носитель), который будет использоваться в дальнейшем. После завершения начальной загрузки шифратор ожидает от ПК команды и данные на исполнение операций шифрования.

Помимо собственно функций шифрования, каждый шифратор в этом режиме должен, как минимум:

- выполнять различные операции с ключами шифрования: их загрузку в шифропроцессор и выгрузку из него, а также взаимное шифрование ключей;
- рассчитывать имитовставки для данных и ключей;
- генерировать случайные числа по запросу.

Рассмотрим работу шифратора в операционных системах семейства *Microsoft Windows*. В общем случае шифратор может получать команды сразу от нескольких программ. Например, это могут быть команды программы шифрования файлов; команды шифрования данных и вычисления имитовставок от драйвера, выполняющего прозрачное (автоматическое) шифрование сетевых пакетов (например, реализующего механизмы виртуальных частных сетей); запросы на генерацию случайных чисел от программы-генератора криптографических ключей и т.д.

Во избежание возникновения коллизий программы не имеют прямого доступа к шифратору и управляют им с помощью специальных программных *API*-модулей. Например, устройствами серии «Криптон» (общий вид одного из которых показан на рис. 17.4) управляет универсальный программный интерфейс *Crypton API* (рис. 17.5).

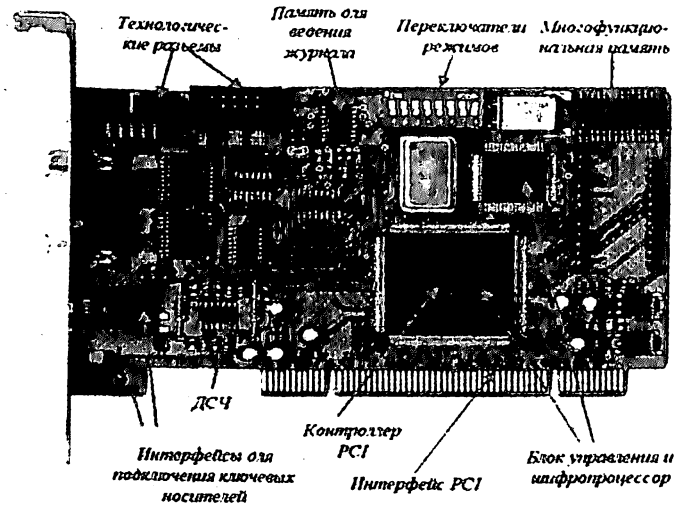


Рис. 17.4. Общий вид аппаратного шифратора типа «Криптон»

В функции данного *API* входит обеспечение корректного последовательного выполнения шифратором команд, инициированных различными программами. Для каждой программы создается отдельная сессия шифрования, а ресурсы шифратора поочередно переключаются между сессиями. Каждая сессия имеет собственный виртуальный шифратор со своими ключами шифрования, которые перезагружаются при переключениях между сессиями. Это несколько

напоминает разделение ресурсов ПК между приложениями в многозадачной операционной системе.

В тот же набор *Crypton API* входят модули, обеспечивающие стандартный интерфейс к функциям шифратора и Windows-приложениям — ключевым носителям. Кроме того, этот API поддерживает возможность подключения различных типов шифраторов через драйверы со стандартным набором функций. Это исключает зависимость прикладной программы от конкретного типа шифратора. Например, вместо аппаратного шифратора можно использовать программный — *Crypton Emulator*, работающий на уровне ядра операционной системы. Аналогичным образом поддерживается и работа с разными ключевыми носителями.

Таким образом, при обращении программы к УКЗД любая команда проходит четыре уровня (см. рис. 17.5): приложений, интерфейса между приложением и драйвером УКЗД, ядра операционной системы — драйвера УКЗД и аппаратный (собственно уровень шифратора).

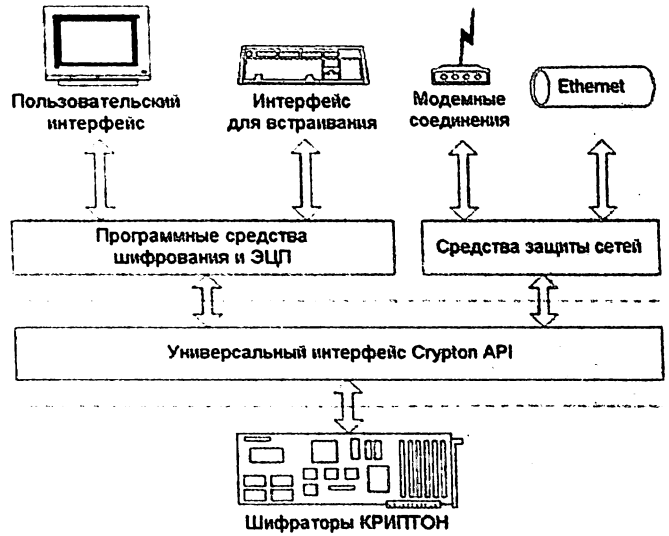


Рис. 17.5. Программный интерфейс Crypton API

Ключевые схемы и процесс шифрования файлов. Аппаратные шифраторы должны поддерживать несколько уровней ключей шифрования. Обычно реализуется трехуровневая иерархия ключей: большее число уровней, как правило, уже не дает заметного улучшения качества защиты, а меньшего может не хватить для ряда ключевых схем. Трехуровневая иерархия предусматривает использова-

ние сеансовых или пакетных ключей (1-й уровень), долговременных пользовательских или сетевых ключей (2-й уровень) и главных ключей (3-й уровень).

Каждому уровню ключей соответствует ключевая ячейка памяти шифропроцессора. При этом подразумевается, что шифрование данных выполняется только на ключах первого уровня (сеансовых или пакетных), остальные же предназначены для шифрования самих ключей при построении различных ключевых схем.

Трехуровневую схему лучше всего иллюстрирует следующий упрощенный пример процесса шифрования файла:

1. На этапе начальной загрузки в ключевую ячейку № 1 заносится главный ключ. Но для трехуровневого шифрования необходимо получить еще два.

2. Сеансовый ключ генерируется в результате запроса к датчику случайных чисел (ДСЧ) шифратора на получение случайного числа, которое загружается в ключевую ячейку № 2, соответствующую сеансовому ключу. С его помощью шифруется содержимое файла и создается новый файл, хранящий зашифрованную информацию.

3. Далее у пользователя запрашивается долговременный ключ, который загружается в ключевую ячейку № 3 с расшифровкой посредством главного ключа, находящегося в ячейке № 1. («Серьезный» шифратор должен иметь режим расшифровки одного ключа с помощью другого *внутри* шифропроцессора; в этом случае ключ в открытом виде вообще никогда «не покидает» шифратора).

4. И, наконец, сеансовый ключ зашифровывается при помощи долговременного ключа, находящегося в ячейке № 3, выгружается из шифратора и записывается в заголовок зашифрованного файла.

5. При расшифровке файла сначала с помощью долговременного ключа пользователя расшифровывается сеансовый ключ, а затем с его помощью восстанавливается информация.

В принципе можно использовать для шифрования и один ключ, но многоключевая схема имеет серьезные преимущества. Во-первых, снижается нагрузка на долговременный ключ — он используется только для шифрования коротких сеансовых ключей. А это усложняет потенциальному злоумышленнику криптоанализ зашифрованной информации с целью получения долговременного ключа. Во-вторых, при смене долговременного ключа можно очень быстро перешифровать файл: достаточно перешифровать сеансовый ключ со старого долговременного на новый. И в-третьих, разгружается ключевой носитель — на нем хранится только главный ключ, а все долговременные ключи (а их может быть сколько угодно — для различных целей) могут храниться в зашифрованном с помощью главного ключа виде даже на жестком диске ПК.

Дополнительные защитные функции шифраторов. Чтобы улучшить соотношение функциональность / цена, аппаратные шифраторы оснащают различными дополнительными защитными функциями. Из них наиболее полезная и часто применяемая — функция «электронного замка», обеспечивающая ПК защиту от несанкционированного доступа и позволяющая контролировать целостность файлов операционной системы и используемых приложений.

Память каждого шифратора, работающего в режиме «электронного замка», должна содержать следующую информацию, которая формируется администратором безопасности или аналогичным по функциям должностным лицом:

– список пользователей, которым разрешен вход на защищаемый данным шифратором компьютер, и данные, необходимые для их аутентификации;

– список контролируемых файлов с рассчитанным для каждого из них хэш-значением;

– журнал, содержащий список попыток входа на компьютер, как успешных, так и нет; в последнем случае — с указанием причины отказа в доступе.

В режиме начальной загрузки «электронный замок» шифратора прежде всего запрашивает у пользователя аутентификационную информацию. Обычно она хранится на том же ключевом носителе, что и главный ключ, и вводится в шифратор напрямую. В случае успешной аутентификации выполняется анализ целостности файлов согласно списку, хранимому в памяти шифратора (путем расчета хэш-значений файлов и сравнения их с эталонными). При нарушении целостности хотя бы одного из контролируемых файлов загрузка компьютера блокируется, а шифратор переходит в специальный режим работы — впредь вход на компьютер будет разрешен только администратору по безопасности, а обычным пользователям вход до устранения несоответствия будет закрыт. Зафиксировав попытку входа в собственном журнале, шифратор возвращает компьютеру управление, что позволяет продолжить загрузку ОС. Однако «электронный замок» продолжает контролировать процесс загрузки, в частности, блокируя попытки загрузки с альтернативных носителей — дискеты или компакт-диска.

Все эти меры обеспечивали бы совершенно исключительную безопасность, если бы не один серьезный недостаток «электронного замка» — его можно просто вытащить из компьютера. Тем не менее, если использовать его в паре с программой прозрачного шифрования логических дисков, загрузка компьютера без шифратора не даст злоумышленнику желаемого эффекта. Любые попытки модифицировать систему с целью, например, внедрения программной закладки будут

обнаружены при первой же загрузке с вставленным шифратором в процессе контроля целостности файлов. Такой программно-аппаратный комплекс (шифратор в режиме «электронного замка» *плюс* программа шифрования логических дисков) — весьма надежное средство защиты информации на ПК.

17.3. ОСНОВНЫЕ ТИПЫ СОВРЕМЕННЫХ ШИФРАТОРОВ

Одним из первых аппаратных персональных средств криптографической защиты информации в России является Шипка-1.5 ОКБ САПР (рис. 17.6).

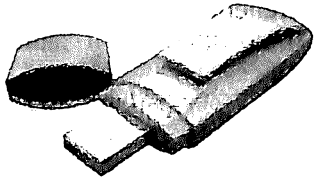


Рис. 17.6. Внешний вид персонального шифратора Шипка-1.5

Шипка-1.5 — это аббревиатура от слов «Шифрование — Идентификация — Подпись — Коды Аутентификации». Внешне это изделие ничем не отличается от обычного USB-ключа, но при этом выполняет все функции слов, из которых составлено его название. Шипка-1.5 — это USB-устройство, в котором аппаратно реализованы:

1. Все стандартные российские криптографические алгоритмы:
– шифрование (ГОСТ 28147–89);
– вычисление хэш-функции (ГОСТ Р 34.11–2012 Криптографическая защита информации. Функция хэширования);
– вычисление и проверка ЭЦП (ГОСТ Р 34.10–2012 Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи);
– вычисление защитных кодов аутентификации (ЗКА): чтобы убедиться, что данные правильно обрабатываются и нет нарушений в технологии, используются защитные коды аутентификации; для этого в некоторых точках происходит проверка результата операций, и если он не совпадает с «правильным», подается сигнал тревоги.

2. Ряд зарубежных алгоритмов:
– шифрование RC2, RC4 иRC5, DES, 3DES, RSA;
– хэш-функции MD5 и SHA-1;
– ЭЦП RSA, DSA.

3. Два изолированных энергонезависимых блока памяти:
- для хранения критичной ключевой информации — память объемом 4 Кбайт, размещенная непосредственно в вычислителе;
- для хранения разнообразной ключевой информации, паролей, сертификатов и т.п. — память объемом до 2 Мбайт, часть которой может быть выделена для организации защищенного диска небольшого объема.

4. Аппаратный генератор случайных чисел.

С помощью устройства Шипка-1.5 можно решать самые разные задачи защиты информации как персонального, так и корпоративного уровня, например:
– шифрование и / или подпись файлов;
– защищенное хранилище паролей для различных web-сервисов;
– аппаратная идентификация пользователя в бездисковых решениях типа «тонкий клиент»;
– аппаратная идентификация пользователя для ПАК «Аккорд-NT/2000», установленного на ноутбуках;
– аппаратная авторизация при загрузке ОС Windows на ПК;
– хранилище ключей и аппаратный датчик случайных чисел для криптографических приложений;
– использование смарт-карты в типовых решениях, таких как авторизация при входе в домен Windows, шифрование и/или подпись сообщений в почтовых программах (например, Outlook Express), для получения сертификатов Удостоверяющего Центра для пар «имя пользователя + открытый ключ».

Аппаратная реализация вычислений без привлечения ресурсов компьютера — это важное отличие устройства Шипка-1.5 от других известных решений на базе USB-ключей. В устройстве Шипка-1.5 программно реализуются только не влияющие на безопасность транспортные процедуры и процедуры согласования форматов данных, все остальные функции выполняются аппаратно.

Это значит, что никто не сможет вмешаться в протекание процессов аутентификации, шифрования или ЭЦП и фальсифицировать их. Также это значит, что после отключения устройства Шипка-1.5 в памяти компьютера не остается никаких следов секретных ключей пользователя и никто другой ими не воспользуется. При этом можно применять все вышеперечисленные возможности на любом компьютере, поскольку вся ключевая информация хранится в устройстве Шипка-1.5.

Однако это не значит, что любой, кто завладеет устройством Шипка-1.5, автоматически завладеет и всей хранящейся в нем информацией — доступ к ней защищен PIN-кодом, и в случае превышения допустимого числа неверных введений устройство блокируется и вся информация на нем уничтожается.

Возможность хранения в устройстве Шипка-1.5 паролей позволит пользователю не выбирать между надежностью пароля и простотой

его запоминания, и при этом избежать таких распространенных ошибок как хранение паролей в блокноте или на листочках, а также использование одного и того же пароля в разных случаях.

Кроме того, устройство Шипка-1.5 является полностью программируемым. Это дает возможность легко расширять его функциональность.

Сегодня на рынке средств защиты информации имеется, помимо устройства Шипка-1.5, уже целый ряд аппаратных устройств персональной криптографической (гарантированной) защиты.

В частности, компанией «Актив» совместно с фирмой «Анкад» разработан ряд персональных электронных идентификаторов серии ruToken (рис. 17.7), которые являются полнофункциональным аналогом смарт-карты, выполненным в виде миниатюрного USB-брелока. Эти электронные идентификаторы подключаются к компьютеру через USB-порт и не требуют дополнительного считывателя. Подобный ruToken имеет свою собственную файловую систему, аппаратную реализацию алгоритма шифрования по ГОСТ 28147–89 и содержит до 128 Кбайт защищенной энергонезависимой памяти.

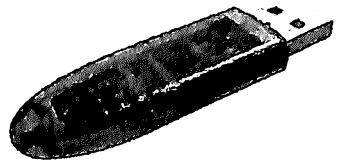


Рис. 17.7. Общий вид персональных идентификаторов типа ruToken

Применение ruToken позволяет существенно увеличить эффективность информационной защиты за счет того, что вход в сеть, защита электронной переписки и шифрование данных могут осуществляться на основе цифровых сертификатов, хранящихся в защищенной памяти ruToken. При использовании подобного электронного идентификатора одновременно существенно повышаются уровень безопасности сети и удобство работы пользователей.

Электронный идентификатор ruToken позволяет обеспечить:

– надежную двухфакторную аутентификацию пользователей;

– хранение в памяти ruToken ключей шифрования, паролей и сертификатов;

– защиту электронной почты (ЭЦП, шифрование);

– сокращение эксплуатационных затрат, простоту использования.

17.4. ОСНОВНЫЕ НАПРАВЛЕНИЯ РАЗВИТИЯ ТЕХНОЛОГИИ СМАРТ-КАРТ

В настоящее время наибольшими возможностями для реализации персональных шифраторов, отвечающих всем современным требованиям, обладают технологии интеллектуальных карт (смарт-карт), особенности и возможности которых рассмотрим более подробно.

Цифровые интеллектуальные карты — пластиковые карточки размером с кредитную со встроенным микроконтроллером и защищенной памятью — с каждым днем становятся все «умнее и умнее». Это происходит в результате использования в них более мощных и быстрых микроконтроллеров следующего поколения или ядер процессоров, оптимизированных специально для удовлетворения потребностей таких карт. Сейчас разработчики при создании оптимизированных кристаллов для интеллектуальных карт возлагают основные надежды на использование сверхмалогабаритных 16- или 32-разрядных ядер процессоров. Альтернативой этому являются более высокопроизводительные 8-разрядные устройства, подобные RISC-процессорам, выполняющим одну команду всего за один цикл, в отличие от 6–12 циклов на команду для большинства распространенных микроконтроллеров, например, серии 8051. Для различных приложений интеллектуальных карт в зависимости от требуемой производительности уже сейчас имеется широкий набор микропроцессорных устройств с различной вычислительной мощностью.

Однако увеличение производительности неминуемо влечет за собой рост потребляемой мощности, что абсолютно недопустимо, так как микроконтроллеры питаются от внешних источников энергии. Сейчас питание интеллектуальных карт осуществляется или через печатные контактные площадки, или по радиоканалу, когда ток, необходимый для питания устройства, наводится во встроенной в карту рамочной антенне внешним электромагнитным полем высокой частоты, и требование минимальной потребляемой мощности здесь является ключевым. Кроме того, поскольку плата должна сохранять данные после отключения питания, здесь потребуется наличие энергонезависимой, но электрически перепрограммируемой памяти (EEPROM).

Для реализации всех необходимых функций в современных сложных устройствах помимо высокопроизводительных микропроцессоров потребуется большой объем памяти, где будет храниться вся необходимая информация. Нынешнее поколение микроконтроллеров, как правило, имеет встроенное ПЗУ, а также ОЗУ и EEPROM. Но наличие относительно небольшого объема перепрограммируемой памяти объясняется физическими и технологическими ограниче-

ниями на плотность компоновки кристалла микросхемы. Самые простые микропроцессоры имеют ОЗУ объемом 128 байт, EEPROM объемом 256 байт и ПЗУ емкостью приблизительно 6 Кбайт. Самые сложные современные микроконтроллеры могут объединять ОЗУ объемом 6 Кбайт, EEPROM объемом 16 Кбайт и ПЗУ объемом до 32 Кбайт. Кроме того, в последних разработках становится популярным использование флэш-памяти, которая выступает реальной альтернативой ПЗУ. По мере развития интеллектуальных карт для хранения данных и программы потребуется двойное или 4-кратное увеличение емкости запоминающих устройств.

Поскольку полупроводниковые технологии постоянно совершенствуются, в будущем интеллектуальные карты смогли бы стать сверхмалогабаритным одноплатным компьютером. Такая карта, по мнению специалистов компании Siemens, представляла бы собой, например, не только высокопроизводительный процессор, объединенный с криптографическим оборудованием и запоминающим устройством большой емкости. Она могла бы содержать вспомогательную клавиатуру, с помощью антенны осуществлять бесконтактную подачу питания и связь с внешними устройствами. Для обеспечения питания, независимого от устройства считывания, карты могли бы содержать солнечные батареи. Возможно встраивание в карты биометрических датчиков для идентификации владельца, например, считывателя отпечатка пальца, а также какого-либо громкоговорителя, сигнализатора и дисплея для отображения и контроля информации. Хотя до практического воплощения всех этих замыслов еще далеко, многие из указанных технологий уже существуют и успешно совершенствуются.

Критерии выбора аппаратных шифраторов. В заключение коснемся основных технических характеристик аппаратных шифраторов, которые следует учитывать при выборе конкретного устройства.

Важнейшая характеристика — реализуемый алгоритм шифрования и размерность ключа. Согласно отечественному законодательству, государственные и ряд коммерческих организаций обязаны применять только те криптосредства, которые имеют сертификат ФСТЭК (Федеральная служба по техническому и экспортному контролю). Сертификация же шифратора в ФСТЭК в настоящий момент подразумевает, что в нем реализован отечественный алгоритм ГОСТ 28147–89.

Уместно упомянуть и о том, что деятельность по разработке, производству, распространению и техническому обслуживанию шифраторов (как аппаратных, так и программных) является лицензируемой как в нашей стране, так и в большинстве развитых стран мира. Следует убедиться, что производитель и / или поставщик шифраторов обладает необходимым набором лицензий.

Остальные параметры — такие, как скорость шифрования, число уровней ключевой системы шифратора, интерфейс (ISA / PCI / USB), набор поддерживаемых ключевых носителей с возможностью прямой загрузки ключей шифрования, наличие функциональности «электронного замка», наличие драйверов шифратора для различных ОС, наличие программного обеспечения, позволяющего использовать функциональность шифратора, — определяются в соответствии с техническими особенностями ПК, требованиями к защищенности информации и политикой безопасности, принятой в данной организации.

ГЛАВА 18. ОСОБЕННОСТИ ПРОГРАММНО-АППАРАТНОЙ РЕАЛИЗАЦИИ КРИПТОГРАФИЧЕСКОЙ ЗАЩИТЫ КОМПЬЮТЕРНЫХ СЕТЕЙ И СЕТЕЙ СВЯЗИ

18.1. ПРОХОДНЫЕ ШИФРАТОРЫ: СТРУКТУРА И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

18.1.1. ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ И СТРУКТУРА ПРОХОДНОГО ШИФРАТОРА

Для защиты передаваемой в сеть связи информации можно использовать как обычный аппаратный шифратор (см. гл. 17), так и **проходной шифратор** (ПШ), который, помимо всех вышеперечисленных функций обычного шифратора, является также полноценным сетевым адаптером Ethernet (т.е. шифратор и сетевой адаптер выполнены в качестве одной PCI-платы). Его достоинство состоит в том, что он полностью контролирует весь обмен данными по сети, а обойти его (как изнутри, так и снаружи) просто невозможно.

ПШ являются достаточно сложными устройствами, так как они вместо центрального процессора компьютера вынуждены выполнять дополнительные функции по обработке информации. Обычно в ПШ ставят два шифропроцессора: один из них отвечает за шифрование отправляемых данных, а другой расшифровывает принимаемые (рис. 18.1). Такое устройство может хранить в себе несколько сотен ключей, чтобы каждый блок информации был зашифрован на своем ключе, отличном от других. Это делает все ключи абсолютно недоступными злоумышленникам, но несколько затрудняет процесс управления ими.

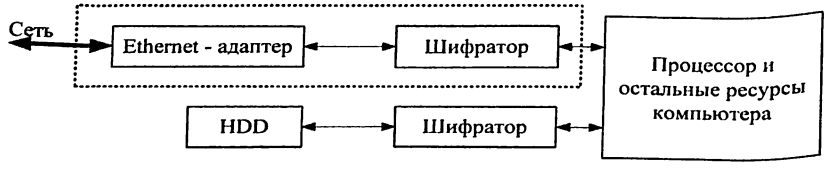


Рис. 18.1. Проходные шифраторы

Технические трудности не позволяли до последнего времени разработать надежные и быстродействующие ПШ. Однако с недавним появлением на рынке дорогих, но очень качественных микросхем **PLD** решаются многие проблемы создания сложных многофункциональных устройств, что стимулировало выпуск первых отечественных проходных шифраторов.

Следует отметить, что ПШ допускает и другое применение: он может стоять в разрыве между жестким диском компьютера и его контроллером. В этом случае все, что пишется на HDD, будет также автоматически шифроваться.

Разработчики аппаратных шифраторов и программного обеспечения для них полагают, что уже скоро будут созданы УКЗД, осуществляющие управление не только работой дисководов, CD-ROM и портов ввода-вывода, но и всеми ресурсами ПК. В ближайшем будущем компьютеру останется только передавать открытые данные между процессором и оперативной памятью и обрабатывать их, все остальное сделает само УКЗД. Ясно, что абсолютному большинству пользователей это не потребуется. Но там, где ведется работа с важными и конфиденциальными документами, информация должна быть серьезно защищена.

18.1.2. ЗАГРУЗКА КЛЮЧЕЙ ШИФРОВАНИЯ

Есть еще одна особенность использования ключей в ПШ, касающаяся безопасности: чтобы у злоумышленника не было совсем никаких шансов, необходимо ключи загружать в шифратор, минуя оперативную память компьютера, где их теоретически можно перехватить и даже подменить. Для этого УКЗД дополнительно содержит порты ввода-вывода, например, COM или USB, к которым напрямую подключаются разные устройства чтения ключевых носителей. Это могут быть любые смарт-карты (пластиковые карты с микросхемой памяти или микропроцессором), специальные USB-ключи или электронные таблетки Touch Memory. Помимо прямого ввода ключей в УКЗД, многие из таких носителей обеспечивают и их надежное хранение — даже украв USB-ключ, без специального кода доступа к его содержимому не подобраться.

18.1.3. ВЗАИМОДЕЙСТВИЕ ШИФРАТОРА С ПРОГРАММАМИ КОМПЬЮТЕРА

Установленный на компьютере шифратор может использоваться сразу несколькими программами, например, программой прозрачного шифрования, «прогоняющей» данные сквозь шифратор, и программой электронной подписи, использующей для вычисления подписи получаемые от шифратора случайные числа.

Для того чтобы не возникало коллизий при одновременном обращении к шифратору разных программ (представим, что одна из них шифрует логический диск, а вторая на другом ключе расшифровывает файл: если не управлять очередью выполнения шифратором

их требований, получится абракадабра), ставят специальное программное обеспечение (ПО) управления шифратором (рис. 18.2). Такое ПО выдает команды через драйвер шифратора и передает последнему данные, следя за тем, чтобы потоки информации от разных источников не пересекались, а также за тем, чтобы в шифраторе всегда находились нужные ключи.

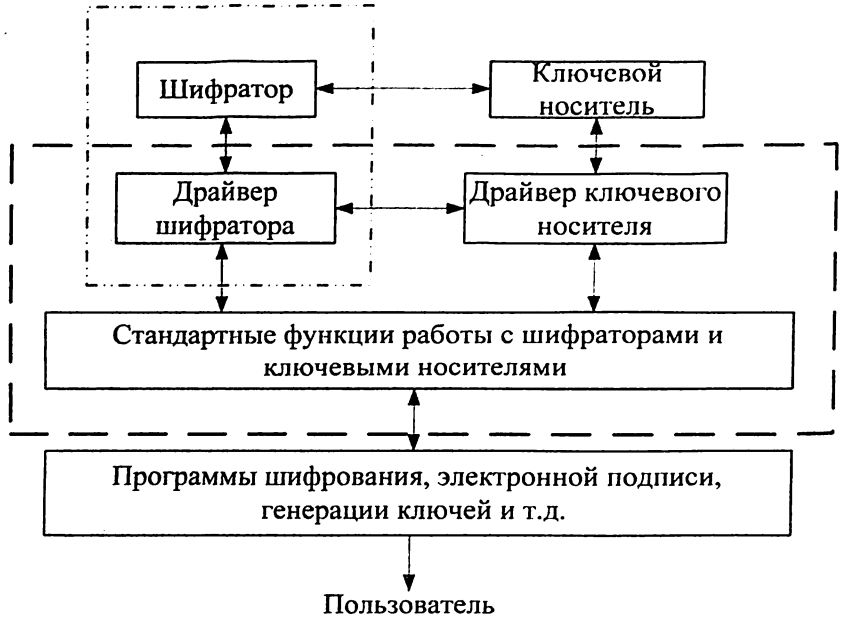


Рис. 18.2. Программный интерфейс для шифратора

Таким образом, УКЗД выполняет два принципиально разных вида команд:

– перед загрузкой операционной системы — команды, зашитые в память шифратора. Они осуществляют все необходимые проверки и устанавливают требуемый уровень безопасности, допустим, отключают внешние устройства.

– после загрузки, например, Windows, — команды, поступающие через модуль управления шифраторами: шифровать данные, перезагружать ключи, вычислять случайные числа и т.д.

Такое разделение необходимо из соображений безопасности — после выполнения команд первого блока, которые нельзя обойти, злоумышленник уже не сможет сделать что-либо запрещенное.

Еще одно назначение ПО управления шифраторами — обеспечить возможность замены одного шифратора на другой (скажем, на более «продвинутый» или быстрый), не меняя программное обеспечение. Это происходит аналогично, например, смене сетевой карты: шифратор поставляется вместе с драйвером, который позволяет программам выполнять стандартный набор функций. Те же программы шифрования и не заметят такой подмены, но будут работать в несколько раз быстрее.

Таким же образом можно заменить аппаратный шифратор на программный. Для этого программный шифратор выполняют обычно в виде драйвера, предоставляющего тот же набор функций.

Впрочем, такое ПО нужно не всем шифраторам — в частности, ПШ, стоящий по пути к HDD, достаточно настроить один раз, после чего о нем можно просто забыть.

18.1.4. ПРИКЛАДНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Программное обеспечение, выполняющее функции защиты информации, должно быть максимально удобным для пользователя и предъявлять минимальные требования к его квалификации. В противном случае можно утверждать, что максимальная эффективность от внедрения программных средств защиты достигнута не будет, поскольку:

– далеко не все пользователи, обрабатывающие подлежащую защите информацию, имеют высокую квалификацию;

– средство защиты, применение которого требует заметных затрат времени и сил, может вызывать у пользователей скрытое или явное противодействие его внедрению.

Поэтому любая программа защиты информации должна соответствовать, как минимум, одной из следующих характеристик:

1. Автоматическое и незаметное для пользователя выполнение — для программ прозрачного шифрования и средств построения VPN.

2. Дублирование возможностей стандартных программ типа Windows Explorer («Проводник») позволяет выполнять все действия с файлами и папками на диске, не выходя из программы защиты информации.

3. Встраивание функций защиты в известные и широко применяемые продукты: расширения меню (Shell Extensions) Windows Explorer, дополнительные панели инструментов в Microsoft Word и Microsoft Excel, автоматический перехват событий в Microsoft Outlook и т.д.

В качестве примера средства защиты, дублирующего основные функции Windows Explorer, можно привести специализированный архиватор (программа, выполняющая специализированное архивирование: вычисление ЭЦП информации, ее сжатие и зашифрование) Crypton ArcMail — его главное окно приведено на рис. 18.3.

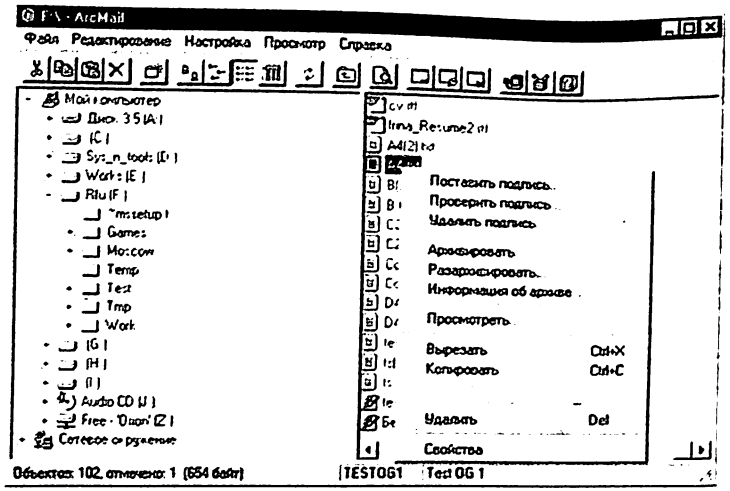


Рис.18.3. Главное окно специализированного архиватора Crypton ArcMail

Как видно из рис. 18.3, меню данного окна содержит все основные файловые функции, т.е. все операции с файлами, выполняемые пользователями обычно в Windows Explorer, можно производить непосредственно в программе Crypton ArcMail. Основной интерфейс пакетов программ «Криптон®Подпись» и «Криптон®Шифрование» (выполняющих, соответственно, операции с ЭЦП и шифрование файлов) — это расширения контекстного меню Windows Explorer, через которые можно выполнять основные команды данных программных продуктов. Все функции по защите информации могут быть выполнены пользователем, не выходя из стандартной оболочки — Windows Explorer.

Аналогично организован интерфейс к модулям защиты электронных документов Crypton Word и Crypton Excel: доступ к ним пользователь может осуществлять непосредственно из Microsoft Word и Microsoft Excel через дополнительные панели инструментов, т.е. при желании подписать электронной подписью документ и зашифровать его пользователь не должен выходить из Microsoft Word или переключаться на другую программу — достаточно нажать кнопку, после чего текущий (редактируемый в настоящий момент) документ будет сохранен, подписан и зашифрован.

Использовать же Crypton Outlook еще проще — Crypton Outlook осуществляет перехват письма при попытке его отправки, подписывает и шифрует текст письма и все вложения, после чего отправляет уже

зашифрованное письмо. А при получении зашифрованного письма Crypton Outlook его автоматически расшифровывает и проверяет ЭЦП.

Упомянутые выше программные продукты обладают также следующими общими характеристиками:

1. Они имеют сертификаты ФАПСИ или используют в своем составе сертифицированное ФАПСИ криптоядро.
2. Практически все продукты выпускаются в двух вариантах: полнофункциональная версия администратора и версия пользователя, позволяющая выполнять ограниченный набор функций.
3. Выполняют автоматическое протоколирование операций.
4. Существуют также в виде библиотек функций для встраивания.
5. Имеют различные ключевые системы, позволяющие клиенту выбрать оптимальную для конкретных задач.
6. Поддерживают совместимость по форматам с предыдущими версиями, что позволяет осуществлять постепенное обновление программного обеспечения — это особенно актуально для крупных организаций, имеющих территориально-распределенную структуру.
7. Многие из указанных продуктов являются взаимозаменяемыми и совместимыми между собой по основным форматам, например, Crypton Word, Crypton Excel и Crypton Outlook полностью совместимы как между собой, так и со всеми продуктами серии Crypton ArcMail.

18.2. ОРГАНИЗАЦИЯ КРИПТОЗАЩИТЫ ИНФОРМАЦИИ ПРИ ЕЕ ПЕРЕДАЧЕ ПО КАНАЛАМ ТЕЛЕФОННОЙ, МОБИЛЬНОЙ И СПЕЦИАЛЬНОЙ СВЯЗИ

В результате массового развития мобильных систем связи в последнее время весьма актуальной становится проблема гарантированной защиты персональной информации при ее хранении и передаче по каналам мобильной связи. Персональный шифратор обеспечивает возможность абонентского шифрования по принципу «точка-точка».

Первым реализованным в России коммерческим проектом стал продукт ФГУП «НТЦ «Атлас» и его партнера концерна «Гудвин» — специальный мобильный радиотелефон (SMP), с появлением которого в российских сетях GSM появилась возможность для дальнейшего повышения уровня криптографической защиты посредством использования дополнительного абонентского шифрования. Специальный сотовый телефон SMP-Атлас (М-539) стал первым в России законным защищенным аппаратом, который предназначен для передачи персональных конфиденциальных данных в зашифрованном виде. Аппарат имеет встроенный персональный шифратор, при отключении которого трубка работает как обычный GSM-телефон.

Телефон стандарта GSM 900/1800 в открытом режиме обеспечивает выполнение всех штатных функций GSM-терминала, а в защищенном — гарантированную защиту речевой информации. Габариты аппарата 140 × 48 × 25 мм, масса 180 г с аккумулятором, емкости которого хватает на 3,5 ч защищенных разговоров. Аппарат способен обеспечить шифрование с гарантированной стойкостью не только речи, но и SMS, MMS, компьютерных данных и электронной почты (стоимость порядка 2,5 тыс. долл.). Ключ шифрования — симметричный, 256 бит. Специальный процессор выполняет аппаратное шифрование.

Аналогичные функции выполняет и двухпроцессорный КриптоСмартТелефон (так названный разработчиками), созданный в ЗАО «АНКОРТ». Он может работать с аналоговыми, цифровыми и IP-криптотелефонами разработки той же компании в любых стандартных сетях GSM, обеспечивающих передачу данных. Для распределения ключей используется открытый ключ. Общий ключ формируется для каждого сеанса связи. Пользователь может самостоятельно формировать и вводить ключи.

Основные характеристики КриптоСмартТелефона приведены в табл. 18.1.

Таблица 18.1

Основные характеристики КриптоСмартТелефона

Категория характеристик	*Параметры и состав*	*Особенности и возможности*
Общие	Принцип передачи	Радиомодем стандарта 900/1800 МГЦ
	Режимы работы	Шифрование голоса в полнодуплексном режиме, стандартный режим GSM, шифрование SMS, шифрование данных в телефоне, шифрование и передача электронной почты
	Процессоры: – основной, –шифрующий	Моторола MX21 266 M, TMS 320 VC 5416
Криптографические	Криптоалгоритм	Симметричный, 256 бит
	Метод распределения ключей	Открытый ключ + Общий ключ (формируется для каждого сеанса)
	Ключевая мощность	10^{77}

Сравнительные характеристики всех рассмотренных выше персональных шифраторов (в том числе и в гл. 17) приведены в табл. 18.2.

Таблица 18.2

Сравнительные характеристики персональных шифраторов

Персональный шифратор	*Разработчик*	*Алгоритм шифрования*	*Назначение*	*Носитель ключевой информации*	*Примечание*
Шифратор, встроенный в специальный сотовый телефон SMP-Атлас	ФГУП «НТЦ «Атлас» + концерн «Гудвин»	ГОСТ 28147–89	Гарантированная защита информации, передаваемой по сетям GSM	Российская интеллектуальная карта РИК (микросхема КБ5004В Е1)	Персональный шифратор, встроенный в мобильный радиотелефон
Шифратор, встроенный в КриптоСмартТелефон	ЗАО «АНКОРТ»	Симметричный, 256 бит	То же	Шифропроцессор на основе TMS VC 5416	То же
Шифратор, встроенный в специальный сотовый телефон Талисман-GSM	НИИ «КВАНТ»	ГОСТ 2814–89	Криптозащита речевой информации в каналах GSM 900/1800	Микропроцессор	Аппаратный шифратор – гарнитура к телефону с поддержкой Bluetooth
Устройство защиты информации Шипка-1.5	ОКБ «САПР»	ГОСТ 28147–89, ГОСТ Р 34.10–94, ГОСТ Р 34.10-2001, ГОСТ Р 34.11–94	Гарантированная защита информации и информационных технологий	То же	Защищенная энергонезависимая память до 2 Мб
Персональный идентификатор ruToken RF	ЗАО «Актив»	ГОСТ 28147–89	Хранение ключевой информации, контроль доступа к ресурсам ПК и в помещения	USB-брелок	Полнофункциональный аналог смарт-карты + радиочастотная метка

Как видно из табл. 18.2, интеграция микропроцессора и флэш-памяти в персональном шифраторе решает одну из актуальнейших проблем генерации, хранения и распределения ключей, что позволяет уже сегодня решить многие задачи с использованием как автономных, так и встраиваемых персональных шифраторов.

Телефонный скремблер «Грот». Телефонный скремблер «Грот» предназначен для защиты конфиденциальной информации. Он обеспечивает шифрование речевого сигнала и защиту факсимильных сообщений, передаваемых по телефонной сети общего применения.

Характеристики 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 других стран СНГ, включая междугородные и международные каналы 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 посредством использования электронного блокнота индивидуальных ключей.

Шифрование:

– метод шифрования — мозаичный: частотные и временные перестановки;
– метод открытого распределения ключей, позволяющий работать без ручного набора ключей;
– общее число ключевых комбинаций — 2 × 1018;
– возможность введения дополнительного семизначного ключа для идентификации абонента;
– высокая степень криптографической защиты вследствие наличия дополнительных мастер-ключей, которые устанавливаются по желанию заказчика.

Аппаратура криптографической защиты речевой и документальной информации Е-20. АКЗ речевой и документальной информации с гарантированной стойкостью Е-20 обеспечивает:

– режим телефонного аппарата общего пользования;
– режим криптографической защиты речи;
– режим передачи и криптографической защиты данных со встроенным устройством имитозащиты.

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-ТФ.

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.

18.3. СПЕЦИАЛИЗИРОВАННЫЕ ШИФРАТОРЫ

Наиболее удобным для конечного пользователя вариантом применение аппаратного шифратора является автоматическое шифрование данных. Опытному администратору достаточно настроить такой шифратор один раз, после чего вся информация, обрабатываемая конечным пользователем, будет автоматически и незаметно для пользователя шифроваться (пользователю, однако, будет необходимо предъявить шифратору ключи шифрования).

Для таких случаев могут служить, например, следующие варианты «проходных шифраторов», т.е. устройств, выполняющих прозрачное шифрование проходящей через них информации:

КРИПТОН AncNet (фирма АНКАД) — шифратор, совмещенный с полноценной сетевой картой *Ethernet* (рис. 18.4). Слева на рисунке видно гнездо для подключения стандартного Ethernet-трансивера. Шифратор автоматически зашифровывает все данные, отправляемые в сеть, и расшифровывает полученную информацию.

КРИПТОН-IDE (фирма АНКАД) — шифратор для прозрачного шифрования жестких дисков интерфейса IDE (рис. 18.5). Шифрует

по алгоритму ГОСТ 28147–89 со скоростью до 70 Мбит в секунду. Как видно из рисунка, шифратор подключается к IDE-разъему материнской платы компьютера, а к шифратору, в свою очередь, подключается стандартным шлейфом жесткий диск. В результате вся записываемая на жесткий диск информация автоматически зашифровывается, причем, возможности «обойти» шифратор принципиально отсутствуют.

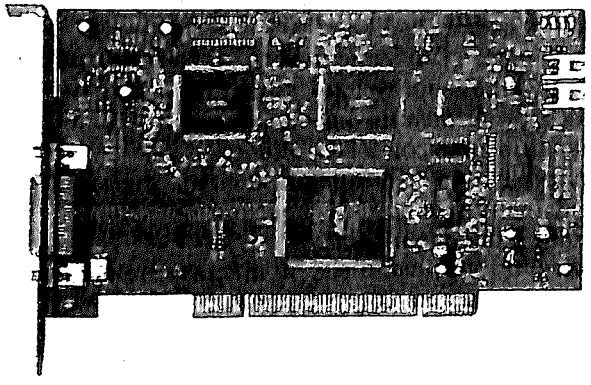


Рис. 18.4. Шифратор КРИПТОН AncNet, совмещенный с сетевой картой Ethernet

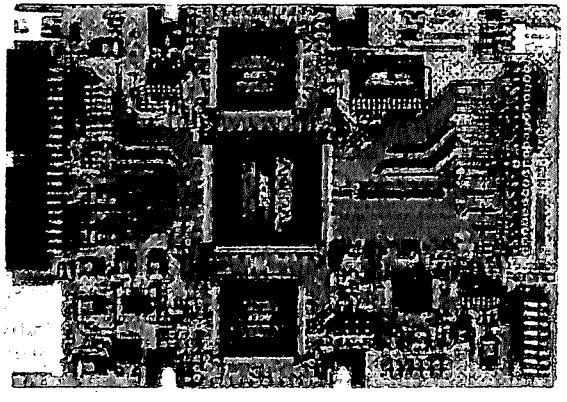


Рис. 18.5. Шифратор КРИПТОН-IDE для прозрачного шифрования жестких дисков интерфейса IDE

Для управления работой данных шифраторов — настройки и загрузки в них криптографических ключей (разъемы для подключения ключевых носителей у данных шифраторов отсутствуют) — предназначен аппаратно-программный модуль доверенной загрузки (АПМДЗ) КРИПТОН-ЗАМОК. Это устройство, помимо управления ПШ, является также полнофункциональным «электронным замком» и выполняет следующие функции:

1. Контроль целостности модулей операционной системы компьютера перед его загрузкой.
2. Аутентификация пользователей.
3. Контроль и блокировка доступа к жестким дискам и дисководам компьютера, а также к устройствам чтения компакт-дисков.
4. Контроль доступа к портам компьютера, а также к сетевым адаптерам.

АПМДЗ может быть центром обеспечения безопасности всего компьютера. Эта технология несколько перекликается с новейшей технологией от Microsoft-Palladium, основу применения которой также составляет аппаратный модуль безопасности.

Аппаратный шифратор «М-506». Отечественный аппаратный шифратор производства ЗАО НИП «Информзащита» СКЗИ (Система криптографической защиты информации) М-506 представляет собой программно-аппаратный комплекс криптографической защиты информации, реализующий алгоритм шифрования данных по ГОСТ 28147–89 (сертифицирован ФАПСИ). В этом комплексном средстве защиты информации возможности аппаратного шифратора дополнены широким спектром других функций информационной безопасности.

СКЗИ М-506 состоит из следующих компонентов:

– *сервер безопасности.* Установленный на выделенном компьютере или контроллере домена, он собирает и обрабатывает информацию о состоянии всех защищаемых рабочих станций и хранит данные о настройках всей системы защиты;

– *средство защиты информации* от несанкционированного доступа (СЗИ) Secret Net NT 4.0. Этим СЗИ оснащаются все рабочие станции сети, на которых устанавливается также «электронный замок “Соболь”». Тем самым защищаются ресурсы компьютера и обеспечивается регистрация входа пользователя в систему, предъявления незарегистрированного идентификатора, введения неправильного пароля, превышения числа попыток входа в систему;

– *подсистема управления.* Этот программный компонент устанавливается на рабочем месте администратора системы и позволяет ему конфигурировать СЗИ Secret Net (а также управлять встроенными возможностями операционной системы), контролировать все события, влияющие на защищенность системы, и реагировать на них в режиме реального времени и в терминах реальной предметной области (сотрудник, задача, подразделение, помещение);

– *криптоменеджер*. Он устанавливается на автономном компьютере и выполняет следующие функции: создание ключей шифрования, изготовление ключевых дискет, ведение базы созданных ключей на жестком диске компьютера.

Реализованная в СКЗИ М-506 клиент-серверная архитектура позволяет сосредоточить в одном месте функции управления безопасностью корпоративной сети и повысить живучесть всей системы защиты: даже выход из строя сервера безопасности не приводит к снижению уровня защищенности. Развитые возможности защиты от несанкционированного доступа интегрированы с криптографическими механизмами: электронной цифровой подписью, «прозрачным» шифрованием файлов и шифрованием сетевого трафика.

Для легального пользователя, которому предоставлено право работать с зашифрованным сетевым ресурсом, данный ресурс предстает в своем обычном, незашифрованном виде («прозрачный» режим криптографического преобразования). Но это не значит, что конфиденциальная информация передается по сети в открытом виде: сетевой трафик шифруется, а расшифровывание происходит только на рабочей станции пользователя.

Отметим, что все компоненты СКЗИ М-506 функционируют в замкнутой программной среде, недоступной воздействию вирусов.

Средства криптографической защиты информации «Верба-О», «Верба-OW». Средства криптографической защиты информации «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;
– 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, предназначенные 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»);

– 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, FrayRelay 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»).

Программный комплекс VCERT PKI. Этот продукт 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:

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.

2. VRA (VCERT Registration Authority) — программный комплекс Центр Регистрации (ЦР), предназначенный для регистрации пользователей и обеспечения взаимодействия пользователя с Центром Сертификации.

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.

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.

ГЛАВА 19. КРИПТОСИСТЕМЫ НА ОСНОВЕ ЭЛЛИПТИЧЕСКИХ КРИВЫХ

Большинство продуктов и стандартов, в которых для шифрования и проверки подлинности применяются методы криптографии с открытым ключом, базируется на алгоритме RSA. Однако число битов ключа, необходимое для надежной защиты данных при использовании RSA, за последние годы резко возросло, что обусловило соответствующий рост загрузки систем, использующих RSA. Криптография на основе **эллиптических кривых** (ECC — Elliptic Curve Cryptography) — появившийся сравнительно недавно подход, способный конкурировать с RSA.

Привлекательность подхода на основе эллиптических кривых в сравнении с RSA заключается в том, что с использованием эллиптических кривых обеспечивается эквивалентный уровень защиты при значительно меньшем числе разрядов, вследствие чего уменьшается загрузка процессора. В то же время, степень доверия к методам криптографии с использованием эллиптических кривых еще не настолько высока, как степень доверия к RSA.

Операция сложения в криптографии на основе эллиптических кривых является аналогом операции умножения по модулю простого числа в RSA, а многократное повторное сложение — аналогом возведения в степень. Чтобы построить криптографическую систему, используя эллиптические кривые, нужно найти «трудную проблему», соответствующую разложению на множители произведения двух простых чисел или дискретному логарифмированию [34].

Цель данной главы — описание построения криптографических систем с открытым ключом, основанных на конечной абелевой группе точек эллиптической кривой, определенной над F_q. Прежде чем описывать криптосистемы, нужно обсудить некоторые вспомогательные понятия.

19.1. АЛГОРИТМЫ НА ОСНОВЕ ЭЛЛИПТИЧЕСКИХ КРИВЫХ

Использование эллиптических кривых для создания криптосистем было независимо предложено Нилом Коблицем (Neal Koblitz) и Виктором Миллером (Victor Miller) в 1985 г. [8, 17]. При использовании алгоритмов на эллиптических кривых предполагается, что не существует быстрых алгоритмов для решения задачи дискретного логарифмирования в группах их точек. В настоящий момент известны лишь экспоненциальные алгоритмы вычисления обратных функций для эллиптических кривых. По сравнению с субэкспоненциальными алгоритмами разложения числа на простые сомножители (см. криптосистему RSA) это позволяет при одинаковом уровне стойкости уменьшить размерность ключа в несколько раз, а следовательно, упростить программную и аппаратную реализацию криптосистем.

Эллиптической кривой E называется множество точек (x,y), удовлетворяющих однородному уравнению Вейерштрасса

$$y^2 + a_1xy + a_2y = x^3 + a_3x^2 + a_4x + a_5, \qquad (19.1)$$

где a_i — коэффициенты уравнения.

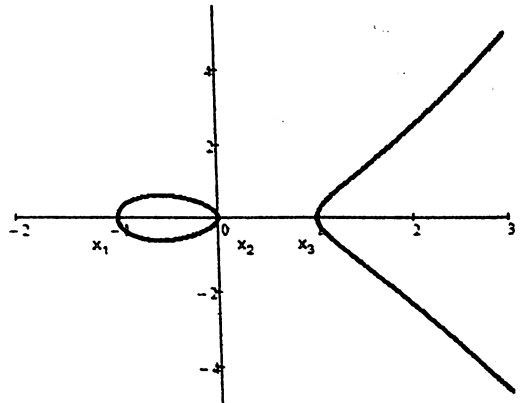
Введение операции сложения над парами точек E позволяет построить абелевую группу точек, если в сеточки E неособые (имеют однозначные производные). Такую кривую называют гладкой, или несингулярной.

Определение. Кривая E называется сингулярной (особой), если существует хотя бы одна точка (x, y), в которой частные производные (19.1) одновременно обращаются в 0, т.е.

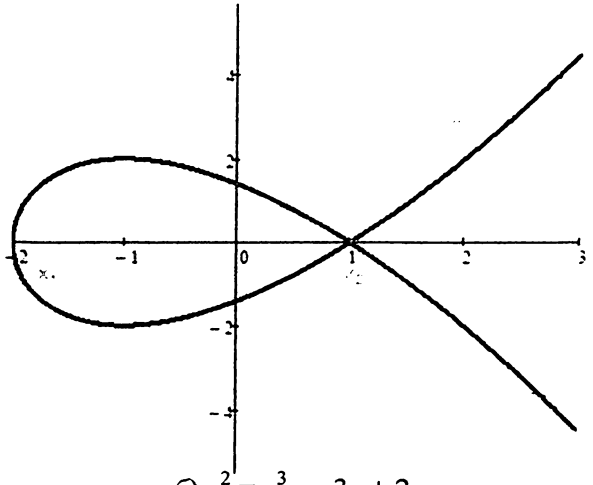
$$\partial F/\partial x = \partial F/\partial y = 0. \qquad (19.2)$$

В противном случае кривая E называется несингулярной (неособой). Такие кривые представляют интерес для криптографии.

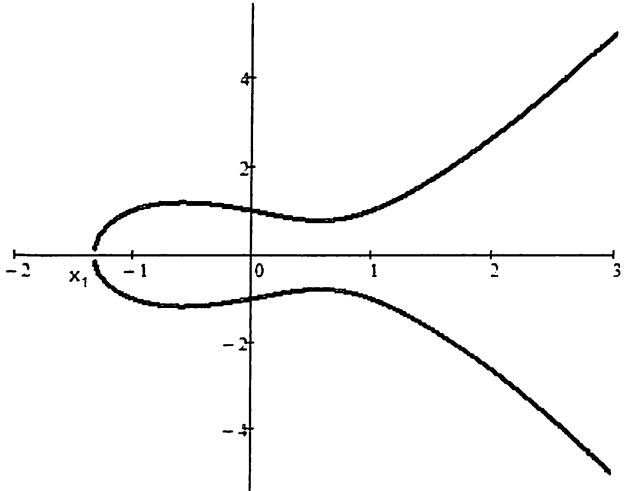
Примеры эллиптических кривых представлены на рис. 19.1 *а, б, в.*



а) $y^2 = x^3 - x$



б) $y^2 = x^3 - 3x + 2$



в) $y^2 = x^3 - x + 1$

Рис. 19.1. Примеры эллиптических кривых [34]

В криптографии эллиптические кривые рассматриваются над двумя типами конечных полей: простыми полями нечетной характеристики (где $n > 3$ — простое число) и полями характеристики 2 ($GF(2^m)$).

Эллиптические кривые над полями нечетной характеристики можно привести к виду, называемому эллиптической кривой в короткой форме Вейерштрасса

$$y^2 = x^3 + Ax + B \pmod{n}. \qquad (19.3)$$

Определение эллиптической кривой также требует, чтобы кривая не имела особых точек. Геометрически это значит, что график не должен иметь каспов и самопересечений. Алгебраически достаточно проверить, что дискриминант

$$\Delta = -16(4a^3 + 27b^2)$$

не равен нулю.

Если кривая не имеет особых точек, то ее график имеет две связные компоненты, если дискриминант положителен, и одну — если отрицателен.

Поскольку график кривой симметричен относительно оси абсцисс, то чтобы найти точки его пересечения с осью абсцисс, необходимо решить кубическое уравнение

$$x^3 + Ax + B = 0. \qquad (19.4)$$

Это можно сделать с помощью известных формул Кардано. Дискриминант этого уравнения

$$\Delta = \left(\frac{A}{3}\right)^3 + \left(\frac{B}{2}\right)^2 = \frac{4A^3 + 27B^2}{108}. \qquad (19.5)$$

Если $\Delta > 0$, то уравнение имеет три различных действительных корня (см. рис. 19.1, *а* — x_1, x_2 и x_3, $\Delta = 64$). Если $\Delta = 0$, то уравнение имеет три действительных корня, по крайней мере, два из которых равны (см. рис. 19.1, *б* — x_1 и x_2). Если $\Delta < 0$, то уравнение имеет один действительный корень (см. рис. 19.1, *в* — x_1, $\Delta = -368$) и два комплексно-сопряженных.

Используемые в криптографии кривые не должны иметь особых точек. Геометрически это значит, что график не должен иметь точек возврата и самопересечений (см. рис. 19.1, *б*). Если кривая не имеет особых точек, то ее график имеет две части при положительном дискриминанте (см. рис. 19.1, *а*), и одну — при отрицательном (см. рис. 19.1, *в*). Например, для графиков, приведенных выше, в первом случае дискриминант равен 64, а в третьем он равен –368.

Следует отметить, что в y каждого ненулевого элемента есть либо два квадратных корня, либо нет ни одного, поэтому точки эллиптической кривой разбиваются на пары вида $P = (x, y)$ и $-P = (x, -y)$. Например, эллиптическая кривая $y^2 = x^3 + 3x + 2$ при $x = 1$ и $n = 5$ имеет две точки в качестве решения: $P = (1, 1)$ и $-P = (1, -1)$, так как $1^2 \equiv 1^3 + 3 \times 1 + 2 \pmod 5$ и $(-1)^2 \equiv 1^3 + 3 \times 1 + 2 \pmod 5$.

Важным свойством несингулярных кривых является то, что любая прямая, проходящая через две различные точки кривой E, пере-

секает эту кривую в единственной третьей точке (см. рис. 19.2). Кроме того, касательная в любой точке кривой (кроме точки перегиба) пересекает ее еще в одной точке. Симметрия кривой относительно оси x позволяет дать наглядное определение обратной к точке $P = (x_1, y_1)$ точки $-P = (x_1, -y_1)$.

Эти свойства позволяют дать определение групповой операции, называемой сложением точек эллиптической кривой.

Введем две операции, которые можно выполнять над точками кривой.

Определение. Суммой двух точек $P_1 = (x_1, y_1)$ и $P_2 = (x_2, y_2)$ называется точка $P_3 = P_1 + P_2 = (x_3, y_3)$, обратная третьей точке пересечения E прямой линией, проходящей через точки P_1 и P_2.

Найдем координаты точки $P_3 = P_1 + P_2 = (x_3, y_3)$, выразив их через координаты точек P_1 и P_2. При этом точки P_1 и P_2 могут быть различными (рис. 19.2) или совпадающими (рис. 19.3).

В соответствии с этим имеют место два случая.

а) $y^2 = x^3 - x$ б) $y^2 = x^3 - x + 1$

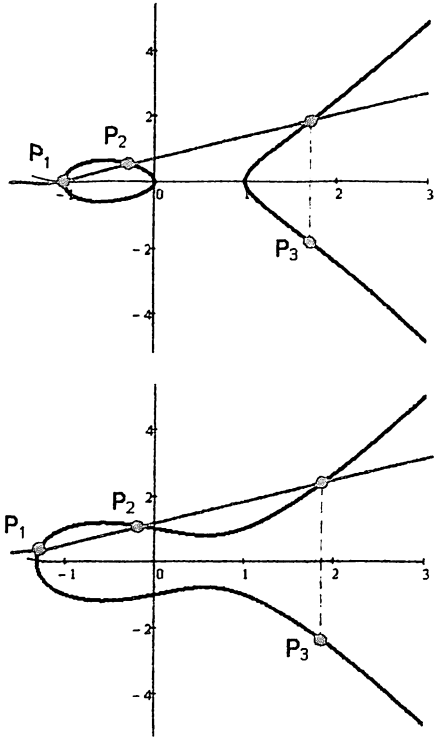


Рис. 19.2. Сложение точек

1. $P_1 \neq P_2$. Уравнение прямой линии, проходящей через точки P_1 и P_2 (рис. 19.2), имеет вид

$$y = \lambda x + \beta; \quad \lambda = \frac{y_2 - y_1}{x_2 - x_1} \quad \beta = y_1 - \lambda x_1 \tag{19.6}$$

Уравнение (19.6) в канонической форме можно переписать

$$F(x,y) = y^2 - x^3 - ax - b = 0. \tag{19.7}$$

Точки пересечения кривой E и прямой (19.6) имеют по оси x координаты x_1, x_2, x_3 точек P_1, P_2 и $-P_3$ соответственно. Поскольку они являются общими для функций (19.6) и (19.7), последнее уравнение можно записать в виде

$$(\lambda x + \beta)^2 - x^3 - ax - b = 0, \text{ или } -(x - x_1)(x - x_2)(x - x_3) = 0.$$

Приравнивая в этих кубических уравнениях коэффициенты при переменных x^2, получим

$$\lambda^2 = x_1 + x_2 + x_3 \tag{19.8}$$

Параметр λ прямой (19.6) можно также выразить в виде

$$\lambda = \frac{-y_3 - y_1}{x_3 - x_1}$$

Из (19.8) и последнего соотношения окончательно имеем координаты точки $P_3 = P_1 + P_2 = (x_3, y_3)$:

$$x_3 = \lambda^2 - x_1 + x_2,$$

$$y_3 = \lambda(x_1 - x_3) - y_1, \lambda = \frac{y_2 - y_1}{x_2 - x_1}. \tag{19.9}$$

2. $P_1 = P_2$, $P_3 = 2P_1$. В этом случае $x_1 = x_2$ и параметр λ в (19.9) не определен.

Дифференциал функции $y^2 = x^3 + ax + b$ равен

$$2ydy = 3x^2dx + adx,$$

тогда при $x = x_1$ производная равна параметру v касательной $y = \lambda x + \beta$ к кривой в точке P.

$$\lambda = \frac{dy}{dx} = \frac{3x_1^2 + a}{2y_1}.$$

Вместо(19.9) теперь можно записать координаты точки $P_3 = 2P_1 = (x_3, y_3)$:

$$x_3 = \lambda^2 - 2x_1, \text{ если } x_1 = x_2$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \quad \lambda = \frac{3x_1^2 + a}{2y_1}, \text{ если } x_1 = x_2$$

Таким образом, сложение точек $P_3(x_3, y_3) = P_1(x_1, y_1) + P_2(x_2, y_2)$ для двух случаев сводится к формулам (19.10) (19.11):

$$x_3 = \lambda^2 - x_1 - x_2 (\mathrm{mod} p) \tag{19.10}$$

$$y_3 = \lambda(x_1 - x_3) - y_1 \pmod p$$

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1}, \text{если } x_1 \neq x_2 \\ \dfrac{3x_1^2 + a}{2y_1}, \text{если } x_1 = x_2 \end{cases} \qquad (19.11)$$

В формулах (19.10) и (19.11) λ — угловой коэффициент прямой, проходящей через точки P_1 и P_2. Если $P_1 = P_2$, то λ равен значению производной в точке P_1.

Для построения абелевой группы точек E определим О группы как

$$P+(-P) = O, \forall P \in E.$$

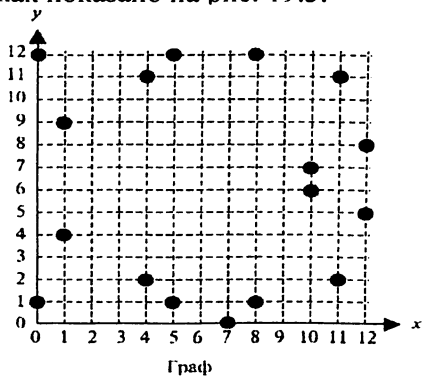
Если провести прямую через точки P и –P, то третья точка пересечения прямой и E уходит в бесконечную точку вдоль оси y. Поэтому О группы точек E называют «точкой на бесконечности».

Смысл перехода к обратной точке пересечения прямой и кривой E при определении суммы $P_3 = P_1 + P_2$ становится понятным, если выразить, например, точку P_1 как $P_1 = P_3 - P_2$. В этом случае прямая проходит через точки P_3, P_1 и P_2, а обратной к этой третьей точке является точка P_3. Для точек E выполняется ассоциативность сложения $P_3 + (P_1 + P_2) = (P_3 + P_1) + P_2$ и коммутативность $P_3 + P_1 = P_1 + P_3$. Таким образом, множество точек E замкнуто относительно операции сложения, удовлетворяет свойствам ассоциативности, коммутативности, имеет обратный элемент и О (нуль группы), т.е. удовлетворяет всем условиям аддитивной абелевой группы.

Пример 19.1. Определим эллиптическую кривую E_{13} (1, 1) по уравнению $y^2 = x^3 + x + 1$ и вычислим по модулю 13 точки на кривой, которые могут быть найдены, как показано на рис. 19.3.

(0,1)	(0,12)
(1,4)	(1,9)
(4,2)	(4,11)
(5,1)	(5,12)
(7,0)	(7,0)
(8,1)	(8,12)
(10,6)	(10,7)
(11,2)	(11,11)

Точки



Граф

Рис. 19.3. Точки на эллиптической кривой в поле GF(p)

Здесь необходимо обратить внимание на следующее:

1. Некоторые значения y^2 не имеют квадратного корня по модулю 13. Они не являются точками на этой эллиптической кривой. Например, точки $x = 2$, $x = 3$, $x = 6$ и $x = 9$ не находятся на кривой.
2. Каждая точка, определенная на кривой, имеет инверсию. Инверсии перечислены как пары. Заметим, что (7, 0) — инверсия самой себя.
3. Пары обратных точек значения y — аддитивные инверсии друг друга в Z_p. Например, 4 и 9 — аддитивные инверсии в Z_{13}. Так что можно сказать, что если 4 — это значение y, то 9 — это значение $(-y)$.
4. Инверсии находятся на тех же самых вертикальных линиях.

Сложение двух точек. Далее используем группу эллиптической кривой, определенную ранее, но вычисления сделаем в GF(p). Вместо вычитания и деления применяем аддитивные и мультипликативные инверсии.

Пример 19.2. Сложим две точки в примере 19.1, R = P + Q, где P = (4, 2) и Q = (10, 6).

1. $\lambda = (6-2) \times (10-4)^{-1} \bmod 13 = 4 \times 6^{-1} \bmod 13 = 5 \bmod 13$.
2. $x = (5^2 - 4 - 10) \bmod 13 = 11 \bmod 13$.
3. $y = [5\,(411) - 2] \bmod 13 = 2 \bmod 13$.
4. R = (11, 2) является точкой на кривой в примере 19.1.

Умножение точки на константу. В арифметике умножение числа на константу k означает прибавление числа самого к себе k раз. Здесь ситуация та же самая. Умножение точки P на эллиптической кривой на константу k означает прибавление точки P к себе k раз. Например, в $E_{13}(1, 1)$ если точка (1, 4) умножается на 4, результат есть точка (5, 1). Если точка (8,1) умножается на 3, результат— точка (10, 7).

В случае криптографии с использованием эллиптических кривых приходится иметь дело с редуцированной формой эллиптической кривой, которая определяется над конечным полем. Особый интерес для криптографии представляет объект, называемый эллиптический группой по модулю p, где p является простым числом. Такая группа определяется следующим образом. Выберем два неотрицательных целых числа a и b, которые меньше p и удовлетворяют условию

$$(4a^3 + 27b^2) \pmod p \neq 0.$$

Тогда $E_p(a, b)$ обозначает эллиптическую группу по модулю p, элементами (x, y)которой являются пары неотрицательных чисел, которые меньше p и удовлетворяют условию

$$y^2 = (x^3 + ax + b) \pmod p$$

вместе с точкой в бесконечности O.

Пример 19.3. Пусть $p = 23$. Рассмотрим эллиптическую кривую $y^2 = x^3 + x + 1$. В этом случае $a = b = 1$, и мы имеем $4 \times 1^3 + 27 \times 1^2 \pmod{23} = 8 \neq 0$, что удовлетворяет условиям эллиптической группы по модулю 23.

Для эллиптической группы рассматриваются только целые значения от (0, 0) до (p, p) в квадранте неотрицательных чисел, удовлетворяющих уравнению по модулю p. В табл. 19.1 представлены точки, являющиеся элементами $E_{23}(1,1)$, отличные от (0,0). В общем случае список точек создается по следующим правилам:

1. Для каждого такого значения x, что $0 \le x < p$, вычисляется

$$x^3 + ax + b \pmod{p}.$$

2. Для каждого из полученных на предыдущем шаге значений выясняется, имеет ли это значение квадратный корень по модулю p. Если нет, то в $E_p(a, b)$ нет точек с этим значением x. Если же корень существует, имеются два значения y, соответствующие x операции извлечения квадратного корня (исключением является случай, когда единственным таким значением оказывается $y = 0$). Эти значения (x, y) и будут точками $E_p(a, b)$.

Таблица 19.1

Точки $E_{23}(1,1)$ на эллиптической кривой

(0, 1)	(6, 4)	(12, 10)
(0, 22)	(6, 19)	(13, 7)
(1, 7)	(7, 11)	(13, 6)
(1, 16)	(7, 12)	(17, 3)
(3, 10)	(9, 16)	(19, 5)
(3, 13)	(11, 3)	(17, 20)
(4, 0)	(11, 20)	(18, 3)
(5, 4)	(12, 4)	(18, 20)
(5, 19)	(9, 7)	(19, 18)

Правила для сложения в $E_p(a,b)$ соответствуют следующим математическим приемам:

1. P + 0 = P.

2. Если P = (x, y), то P + $(x, -y)$ = 0. Точка $(x, -y)$ является отрицательным значением точки P и обозначается –P. Заметим, что $(x,-y)$ лежит на эллиптической кривой и принадлежит $E_p(a,b)$. Например, в случае $E_{23}(1, 1)$для P = (13, 7) имеем –P = (13, –7). Но –7 mod 23 = 16. Таким образом, –P = (13, 16), что и видим в табл.19.1.

3. Если P = (x_1,y_1)и Q = (x_2,y_2), где P ≠ Q, то P + Q = (x_3,y_3) определяется в соответствии с правилами:

$$x_3 = \lambda^2 - x_1 - x_2 \pmod{p},$$
$$y_3 = \lambda(x_1 - x_3) - y_1 \pmod{p},$$

где

1) $\lambda = (y_2 - y_1) / (x_2 - x_1) \pmod{p}$, если P ≠ Q;

2) $\lambda = (3x_1^2 + a) / 2y_1 \pmod{p}$, если P = Q.

Рассмотрим **пример**. Пусть P = (3, 10) и Q = (9, 7). Тогда

$\lambda = (7 - 10)/(9 - 3) = -3/6 = -1/2 = 11 \bmod 23$,

$x_3 = 11^2 - 3 - 9 = 109 - 17 \bmod 23$,

$y_3 = 11(3 - (-6)) - 10 = 89 = 20 \bmod 23$.

Поэтому P + Q = (17, 20). Чтобы получить 2P, найдем

$\lambda = (3(3^2)+1) / (2\times10) = 5/20 = 1/4 = 6 \bmod 23$,

$x_3 = 6^2 - 3 - 9 = 30 = 17 \bmod 23$,

$y_3 = 6(3 - 7) - 10 = -34 = 12 \bmod 23$,

и поэтому 2P = (7, 12).

Пример 19.4. Даны точки P, Q, R на кривой $E_{751}(-1,1)$. Найти точку 2P + 3Q – R.

Координаты точек		
P	Q	R
(72, 497)	(53, 474)	(90, 730)

Решение. Нахождение точки 2P: 2P = (72, 497) + (72, 497).

Так как $x_1 = x_2$, то, следовательно, λ определяется по формуле (19.11):

$$\lambda = \left(\frac{3\times x_1^2 + \alpha}{2y_1}\right) \bmod 751 = \left(\frac{3\times 72^2 - 1}{2\times 497}\right) \bmod 751 =$$
$$= (15551 \times 994^{-1}) \bmod 751 = (16427 \times 34) \bmod 751 =$$
$$= (528734) \bmod 751 = 30$$

Нахождение обратного элемента 994^{-1} осуществляется по формуле:

$$a^{m-2} \equiv a^{-1} \pmod{m}.$$

T.e. $994^{749} \equiv 994^{-1} \pmod{751} = 994^{749} \pmod{751} = 34$,

$x_3 = (\lambda^2 - x_1 - x_2) \bmod 751 = (30^2 - 74 - 53) \bmod 751 = (756) \bmod 751 = 5$

$y_3 = (\lambda \times (x_1 - x_3) - y_1) \bmod 751 = (30 \times (72 - 5) - 497) \bmod 751 = (1513) \bmod 751 = 11$

2P = (5, 11).

Нахождение точки 3Q. Для начала найдем точку 2Q.

2Q = (53, 474) + (53, 474).

Так как $x_1 = x_2$, то, следовательно, λ определяется по формуле(19.11):

$$\lambda = \left(\frac{3\times x_1^2 + \alpha}{2y_1}\right) \bmod 751 = \left(\frac{3\times 53^2 - 1}{2\times 474}\right) \bmod 751 =$$

$$= (8426\times 948^{-1}) \bmod 751 = (513986) \bmod 751 = 302$$

$x_3 = (\lambda^2 - x_1 - x_2) \bmod 751 = (302^2 - 53 - 53) \bmod 751 = (91098) \bmod 751 = 227$

$y_3 = (\lambda \times (x_1 - x_3) - y_1) \bmod 751 = (302 \times (53 - 227) - 474) \bmod 751 =$
$= (-53022) \bmod 751 = (-452) \bmod 751 = 299$

$$2Q = (227, 299)$$

$$3Q = 2Q+Q = (227, 299) + (53, 474).$$

Так как $x_1 \neq x_2$, то,следовательно, λ определяется по формуле (19.11):

$$\lambda = \left(\frac{y_2 - y_1}{x_2 - x_1}\right) \bmod 751 = \left(\frac{474 - 299}{53 - 227}\right) \bmod 751 =$$

$$= \left(\frac{175}{-174}\right) \bmod 751 = (-175\times 174^{-1}) \bmod 751 =$$

$$(-175\times 669) \bmod 751 = (-117075) \bmod 751 = (-670) \bmod 751 = 81$$

$x_3 = (\lambda^2 - x_1 - x_2) \bmod 751 = (81 - 227 - 53) \bmod 751 = (6281) \bmod 751 = 273$

$y_3 = (\lambda \times (x_1 - x_3) - y_1) \bmod 751 = (81 \times (227 - 273) - 299) \bmod 751 =$
$= (-4025) \bmod 751 = (-270) \bmod 751 = 481$

$$3Q = (273, 481).$$

Нахождение точки $2P + 3Q$: $2P + 3Q = (5, 11) + (273, 481)$.

Так как $x_1 \neq x_2$, то в соответствии с формулой (19.11) получим:

$$\lambda = \left(\frac{y_2 - y_1}{x_2 - x_1}\right) \bmod 751 = \left(\frac{481 - 11}{273 - 5}\right) \bmod 751 =$$

$$= \left(\frac{470}{268}\right) \bmod 751 = (470\times 268^{-1}) \bmod 751 =$$

$$(470\times 496) \bmod 751 = (233120) \bmod 751 = 310$$

$x_3 = (\lambda^2 - x_1 - x_2) \bmod 751 = (310^2 - 5 - 273) \bmod 751 =$
$= (95822) \bmod 751 = 445$

$y_3 = (\lambda \times (x_1 - x_3) - y_1) \bmod 751 = (310 \times (5 - 445) - 11) \bmod 751 =$
$= (-136411) \bmod 751 = (480) \bmod 751 = 271$

$$2P + 3Q = (445, 271).$$

Нахождение точки $-R$:

Так как R = (85, 730), следовательно, – R= (85, 21).

Найдем точку 2P + 3Q – R: 2P + 3Q – R = (445, 271) + (85, 21).

Так как $x_1 \neq x_2$, то в соответствии с формулой (19.11) получим:

$$\lambda = \left(\frac{y_2 - y_1}{x_2 - x_1}\right) \bmod 751 = \left(\frac{21 - 271}{85 - 445}\right) \bmod 751 =$$

$$= \left(\frac{-250}{-355}\right) \bmod 751 = (250\times 355^{-1}) \bmod 751 =$$

$$(250\times 696) \bmod 751 = (174000) \bmod 751 = 519$$

$x_3 = (\lambda^2 - x_1 - x_2) \bmod 751 = (519^2 - 445 - 90) \bmod 751 =$
$= (268826) \bmod 751 = 719$

$y_3 = (\lambda \times (x_1 - x_3) - y_1) \bmod 751 =$
$= (519 \times (445 - 719) - 271) \bmod 751 =$
$= (-142477) \bmod 751 = (538) \bmod 751 = 213$

$$2P + 3Q - R = (719, 213).$$

Проверим, что точка 2P+3Q–R=(719, 213) принадлежит эллиптической кривой $E_{751}(-1,1)$:

$$(y^2) \bmod 751 = (x^3 - x + 1) \bmod 751$$

$$(213^2) \bmod 751 = (719^3 - 719 + 1) \bmod 751$$

$$(45369) \bmod 751 = (371694241) \bmod 751$$

309 = 309, т.е. точка 2P+3Q–R=(719, 213) принадлежит эллиптической кривой $E_{751}(-1,1)$.

Умножение точки на число $P_k(x_k, y_k) = k \times P(x, y)$. Точку kP называют скалярным произведением, а процесс ее вычисления — экспоненцированием точки P(возведением в степень).

Определение. Порядком N_E эллиптической кривой называется число всех ее точек (x, y) вместе с точкой на бесконечности (точкой O).

Определение. Порядком точки P эллиптической кривой называется наименьшее натуральное число $m \neq 0$, для которого $mP = O$.

$$y^2 = x^3 - x$$

А) *вариант* 1 — выполнить сложение точки P k раз:

$$P_k(x_k, y_k) = k \times P(x,y) = \underbrace{P+P+P+\ldots+P}_{k} \qquad (19.12)$$

В) *вариант* 2 — с использованием двоичного представления числа $k = (b_L, \ldots, b_2, b_1)$ и операции удвоения точки (рис. 19.4). Например, $k = 110 = (1101110)_2$, тогда $P_k = 64P + 32P + 8P + 4P + 2P$.

Для $k = 110$ вместо 109 сложений будет одно присваивание, четыре сложения и семь удвоений (сложений).

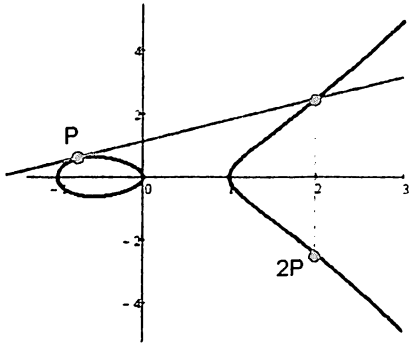


Рис. 19.4. Удвоение точки

Пример 19.5. Дана точка P = (78, 480) на кривой $E_{751}(-1,1)$ и натуральное число n=147. Найти точку nP.

Решение. Представим точку n в двоичном виде: n = 101 = = $(10010011)_2$. При помощи операции удвоения точки представим $P_n = 128P + 16P + 2P + P$.

Нахождение точки 2P: 2P = (78, 480) + (78, 480).

Так как $x_1=x_2$, следовательно, λ определяется по формуле (19.11):

$$\lambda = \left(\frac{3\times x_1^2+\alpha}{2y_1}\right)\bmod 751 = \left(\frac{3\times 78^2-1}{2\times 480}\right)\bmod 751 =$$
$$= \left(\frac{18251}{961}\right)\bmod 751 = (18251\times 960^{-1})\bmod 751 =$$
$$= (18251\times 539)\bmod 751 = (9837289)\bmod 751 = 691$$
$$x_3 = (\lambda^2 - x_1 - x_2)\bmod 751 = (691^2 - 78 - 78)\bmod 751 =$$
$$= (477325)\bmod 751 = 440$$
$$y_3 = (\lambda\times(x_1 - x_3) - y_1)\bmod 751 = (691\times(78 - 440) - 480)\bmod 751 =$$
$$= (-250622)\bmod 751 = (-539)\bmod 751 = 212$$

Нахождение точки 4P: 4P = (440, 212) + (440, 212).

Так как $x_1=x_2$, следовательно, λ определяется по формуле (19.11):

$$\lambda = \left(\frac{3\times x_1^2+\alpha}{2y_1}\right)\bmod 751 = \left(\frac{3\times 440^2-1}{2\times 212}\right)\bmod 751 =$$
$$= \left(\frac{580799}{424}\right)\bmod 751 = (580799\times 424^{-1})\bmod 751 =$$
$$= (580799\times 271)\bmod 751 = (157396529)\bmod 751 = 447$$
$$x_3 = (\lambda^2 - x_1 - x_2)\bmod 751 = (447^2 - 440 - 440)\bmod 751 =$$
$$= (198929)\bmod 751 = 665$$
$$y_3 = (\lambda\times(x_1 - x_3) - y_1)\bmod 751 = (447\times(440 - 665) -$$
$$- 212)\bmod 751 =$$

$$= (-100787)\bmod 751 = (-153)\bmod 751 = 598$$

Нахождение точки 8P: 8P = (665, 598) + (665, 598).

Так как $x_1=x_2$, следовательно, λ определяется по формуле (19.11):

$$\lambda = \left(\frac{3\times x_1^2+\alpha}{2y_1}\right)\bmod 751 = \left(\frac{3\times 665^2-1}{2\times 598}\right)\bmod 751 =$$
$$= \left(\frac{1326674}{1196424}\right)\bmod 751 = (1326674\times 1196^{-1})\bmod 751 =$$
$$= (1326674\times 724)\bmod 751 = (960511976)\bmod 751 = 249$$
$$x_3 = (\lambda^2 - x_1 - x_2)\bmod 751 = (249^2 - 665 - 665)\bmod 751 =$$
$$= (60671)\bmod 751 = 591$$
$$y_3 = (\lambda\times(x_1 - x_3) - y_1)\bmod 751 = (249\times(665 - 591) - 598)\bmod 751 =$$
$$= (17828)\bmod 751 = 555$$
$$8P = (591, 555).$$

Нахождение точки 16P: 16P = (591, 555) + (591, 555).

Так как $x_1=x_2$, следовательно, λ определяется по формуле (19.11):

$$\lambda = \left(\frac{3\times x_1^2+\alpha}{2y_1}\right)\bmod 751 = \left(\frac{3\times 591^2-1}{2\times 555}\right)\bmod 751 =$$
$$= \left(\frac{1047842}{1110}\right)\bmod 751 = (1047842\times 1110^{-1})\bmod 751 =$$
$$= (1047842\times 182)\bmod 751 = (190707244)\bmod 751 = 557$$
$$x_3 = (\lambda^2 - x_1 - x_2)\bmod 751 = (557^2 - 591 - 591)\bmod 751 =$$
$$= (309067)\bmod 751 = 406$$
$$y_3 = (\lambda\times(x_1 - x_3) - y_1)\bmod 751 = (557\times(591 - 406) - 555)\bmod 751 =$$
$$= (102490)\bmod 751 = 354$$
$$16P = (406, 354).$$

Нахождение точки 32P: 32P = (406, 354) + (406,354).

Так как $x_1=x_2$, следовательно, λ определяется по формуле (19.11):

$$\lambda = \left(\frac{3\times x_1^2+\alpha}{2y_1}\right)\bmod 751 = \left(\frac{3\times 406^2-1}{2\times 354}\right)\bmod 751 =$$
$$= \left(\frac{494507}{708}\right)\bmod 751 = (494507\times 708^{-1})\bmod 751 =$$
$$= (494507\times 489)\bmod 751 = (241813923)\bmod 751 = 184$$
$$x_3 = (\lambda^2 - x_1 - x_2)\bmod 751 = (184^2 - 406 - 406)\bmod 751 =$$
$$= (33044)\bmod 751 = 0$$
$$y_3 = (\lambda\times(x_1 - x_3) - y_1)\bmod 751 = (184\times(406 - 0) - 354)\bmod 751 =$$
$$= (74350)\bmod 751 = 1$$
$$32P = (0, 1).$$

Нахождение точки 64P: 64P = (0, 1) + (0, 1).

Так как $x_1=x_2$, следовательно, λ определяется по формуле (19.11):

$$\lambda = \left(\frac{3\times x_1^2+\alpha}{2y_1}\right)\bmod 751 = \left(\frac{3\times 0^2-1}{2\times 1}\right)\bmod 751 =$$
$$= \left(\frac{-1}{2}\right)\bmod 751 = (-1\times 2^{-1})\bmod 751 =$$
$$= (-1\times 376)\bmod 751 = (-376)\bmod 751 = 375$$
$$x_3 = (\lambda^2 - x_1 - x_2)\bmod 751 = (375^2 - 0 - 0)\bmod 751 =$$
$$= (140625)\bmod 751 = 188$$
$$y_3 = (\lambda \times (x_1 - x_3) - y_1)\bmod 751 = (375 \times (0 - 188) - 1)\bmod 751 =$$
$$= (-70501)\bmod 751 = (-658)\bmod 751 = 93$$
$$64P = (188, 93).$$

Нахождение точки 128P:128P = (188, 93) + (188, 93).

Так как $x_1=x_2$, следовательно, λ определяется по формуле (19.11):

$$\lambda = \left(\frac{3\times x_1^2+\alpha}{2y_1}\right)\bmod 751 = \left(\frac{3\times 188^2-1}{2\times 93}\right)\bmod 751 =$$
$$= \left(\frac{106031}{186}\right)\bmod 751 = (106031\times 186^{-1})\bmod 751 =$$
$$= (106031\times 214)\bmod 751 = (22690634)\bmod 751 = 671$$
$$x_3 = (\lambda^2 - x_1 - x_2)\bmod 751 = (671^2 - 188 - 188)\bmod 751 =$$
$$= (449865)\bmod 751 = 16$$
$$y_3 = (\lambda \times (x_1 - x_3) - y_1)\bmod 751 = (671 \times (188 - 16) - 93)\bmod 751 =$$
$$= (115319)\bmod 751 = 416$$
$$128P = (16, 416).$$

Нахождение точки 128P + 16P:128P + 16P = (16, 416) + (406, 354).

Так как $x_1\neq x_2$, следовательно, λ также определяется по формуле (19.11):

$$\lambda = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)\bmod 751 = \left(\frac{354-416}{406-16}\right)\bmod 751 =$$
$$= \left(\frac{-62}{390}\right)\bmod 751 = (-62\times 390^{-1})\bmod 751 =$$
$$(-62\times 518)\bmod 751 = (-32116)\bmod 751 =$$
$$= (-574)\bmod 751 = 177$$
$$x_3 = (\lambda^2 - x_1 - x_2)\bmod 751 = (177^2 - 16 - 406)\bmod 751 = (30907)$$
$$\bmod 751 = 116$$
$$y_3 = (\lambda \times (x_1 - x_3) - y_1)\bmod 751 = (177 \times (16 - 116) - 416)\bmod 751 =$$
$$= (-18116)\bmod 751 = (-92)\bmod 751 = 659$$
$$128P + 16P = (116, 659).$$

Нахождение точки 128P+16P+2P: 128P+16P+2P=(116, 659)+(440, 212).

Так как $x_1\neq x_2$, следовательно, λ определяется по формуле (19.11):

$$\lambda = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)\bmod 751 = \left(\frac{212-659}{440-116}\right)\bmod 751 =$$
$$= \left(\frac{-447}{324}\right)\bmod 751 = (-447\times 324^{-1})\bmod 751 =$$
$$(-447\times 401)\bmod 751 = (-179247)\bmod 751 =$$
$$= (-509)\bmod 751 = 242$$
$$x_3 = (\lambda^2 - x_1 - x_2)\bmod 751 = (242^2 - 116 - 440)\bmod 751 = (58008)$$
$$\bmod 751 = 181$$
$$y_3 = (\lambda \times (x_1 - x_3) - y_1)\bmod 751 = (242 \times (116 - 440) -$$
$$- 659)\bmod 751 = (-16389)\bmod 751 = (-618)\bmod 751 = 133$$
$$128P + 16P + 2P = (181, 133).$$

Нахождение точки 128P + 16P+ 2P+P: 128P + 16P+ 2P+P = = (181, 133) + (78, 480).

Так как $x_1\neq x_2$, следовательно, λ определяется по формуле (19.11):

$$\lambda = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)\bmod 751 = \left(\frac{480-133}{78-181}\right)\bmod 751 =$$
$$= \left(\frac{347}{-103}\right)\bmod 751 = (347\times(-103)^{-1})\bmod 751 =$$
$$(347\times 576)\bmod 751 = (199872)\bmod 751 = 106$$
$$x_3 = (\lambda^2 - x_1 - x_2)\bmod 751 = (106^2 - 181 - 78)\bmod 751 =$$
$$= (10977)\bmod 751 = 463$$
$$y_3 = (\lambda \times (x_1 - x_3) - y_1)\bmod 751 = (106 \times (181 - 463) -$$
$$- 133)\bmod 751 = (-30025)\bmod 751 = (-736)\bmod 751 = 15$$
$$128P + 16P + 2P+P = (463, 15).$$

Проверим, что точка 128P + 16P+ 2P+P = (463, 15) принадлежит эллиптической кривой $E_{751}(-1,1)$.

$$(y^2)\bmod 751 = (x^3 - x + 1)\bmod 751$$
$$(15^2)\bmod 751 = (463^3 - 463 + 1)\bmod 751$$
$$(225)\bmod 751 = (99252385)\bmod 751$$
$$225=225.$$

Все эти формулы в том случае, когда эллиптическая кривая определена над полем действительных чисел, допускают простую геометрическую интерпретацию и становятся интуитивно понятными. Для получения суммы двух разных точек проводится секущая через точки P и Q, находится третья точка пересечения секущей с кривой, эта точка отражается относительно оси абсцисс — полученная точка и есть сумма точек P и Q. Сказанное иллюстрирует классический рисунок, приведенный на рис. 19.2. При удвоении точки (вычислении точки 2P) вместо секущей используется касательная, проходящая через точку P. Обратная точка — это точка, симметричная относительно оси абсцисс.

сительно оси абсцисс. Заметим, что бесконечно удаленная точка лежит в направлении положительного направления оси ординат.

Выписанные выше формулы сложения точек и удвоения точки представляют собой аналитическое выражение приведенных геометрических соображений. Сумма n точек P обозначается nP, эта операция называется скалярным произведением, она аналогична операции возведения в степень в простом конечном поле. Эта операция естественным образом распространяется на все целые числа: если $n < 0$, то $n\mathrm{P} = (-n)(-\mathrm{P})$, $0\mathrm{P} = \mathrm{O}$. В последней формуле в левой части стоит целое число 0, а в правой части — нулевой элемент группы точек эллиптической кривой, т.е. бесконечно удаленная точка.

Формулы сложения показывают, что координаты суммы двух точек выражаются через координаты слагаемых рационально, поэтому если координаты слагаемых принадлежат основному полю, то и сумма этих точек принадлежит этому же полю. Это свойство и позволяет говорить о конечной группе точек и строить в этой группе криптографические алгоритмы. Эти формулы показывают также, что операция в группе точек эллиптической кривой достаточно проста и эффективность ее выполнения определяется эффективностью вычислений в конечных полях характеристики 2. Выполнение последних намного эффективнее по сравнению с вычислениями в простых полях и, что немаловажно, сравнительно просто реализуются в аппаратном виде.

Самая сложная в вычислительном отношении операция в формулах сложения — вычисление обратного элемента. Используя так называемое проективное представление точек эллиптической кривой, можно избавиться от этой операции, правда, за счет некоторого увеличения числа других операций.

Вообще говоря, имеется много вариантов выполнения операций в конечных полях и на эллиптических кривых, зависящих от выбора базиса конечного поля и представления точек кривой, что дает возможность выбора, наилучшим образом подходящего в каждом конкретном случае.

Операция скалярного умножения выполняется с помощью алгоритмов, вполне аналогичных широко известным алгоритмам возведения в степень, которые обычно применяются при реализации обычных криптографических алгоритмов. Кроме того, есть и специфические алгоритмы вычисления скалярного произведения, применимые к определенным классам кривых, например, использующие разложения целых чисел в группах эндоморфизмов эллиптических кривых.

Таким образом, операции имеют следующие свойства:

1. **Замкнутость.** Может быть доказано, что сложение двух точек с использованием операции сложения, определенное в предыдущем разделе, создает другую точку на кривой.

2. **Ассоциативность.** Может быть доказано, что $(\mathrm{P} + \mathrm{Q}) + \mathrm{R} = \mathrm{P} + (\mathrm{Q} + \mathrm{R})$.

3. **Коммутативность.** Группа, состоящая из точек несингулярной эллиптической кривой, — абелева группа. Может быть доказано, что

$$\mathrm{P} + \mathrm{Q} = \mathrm{Q} + \mathrm{P}.$$

4. **Существование нейтрального элемента.** Аддитивный нейтральный элемент в этом случае — нулевая точка. Другими словами,

$$\mathrm{P} + 0 = 0 + \mathrm{P}.$$

5. **Существование инверсии.** Каждая точка на кривой имеет инверсию. Инверсия точки — это ее отражение относительно оси X. Другими словами, точки $\mathrm{P} = (X_1, Y_1)$ и $\mathrm{Q} = (X_1, -Y_1)$ — инверсии друг друга; это означает, что $\mathrm{P} + \mathrm{Q} = 0$. Заметим, что нейтральный элемент — это инверсия самого себя.

Наиболее распространенной операцией во всех криптографических алгоритмах является k-кратное сложение точки P, обозначаемое как kP.

Эту операцию обычно называют скалярным умножением, или, в терминологии мультипликативной группы, экспоненцированием точки кривой.

Дадим краткое описание методов повышения производительности при вычислении точки kP. Подход к расчету точки kP может отличаться в зависимости от того, является ли точка P фиксированной (заранее известной) или произвольной точкой. В первом случае всегда можно пользоваться предвычислениями точек, например, 2^iP, которые хранятся в памяти. Двоичное представление числа k позволяет селектировать те из них, которые в результате суммирования образуют точку kP. Во втором, более общем случае все вычисления приходится проводить в реальном времени.

19.2. ПРОЦЕДУРА СОЗДАНИЯ КЛЮЧЕЙ

Процедура создания ключей представлена в табл. 19.2 и описана далее.

Таблица 19.2

Процедура создания ключей

№ п/п	Основные операции	Пример
1	Выбирается модуль эллиптической кривой — простое число n ($n > 2^{255}$ ГОСТ Р 34.10 - 2001)	$n = 41$
2	Выбираются коэффициенты эллиптической кривой A и B. Должно соблюдаться условие $(4A^3 + 27B^2) \bmod n \neq 0$, в противном случае меняются параметры эллиптической кривой n, A или B	A = 3, B = 7 $(4 \times 3^3 + 27 \times 7^2) \times$ $\times \bmod 41 = 37$
3	Выбирается точка эллиптической кривой $P(x_p, y_p)$ и вычисляется порядок циклической подгруппы группы точек эллиптической кривой q*. Принимается произвольное x_p $(0 < x_p < n)$ и определяется y_p из уравнения эллиптической кривой. Должны соблюдаться условия: – для x_p должен существовать y_p (не для всякого x_p при данных параметрах кривой (A, B, n) может существовать y_p); – $P \neq O$ и $qP = O$, где O — нулевая точка эллиптической кривой $(P+(-P)=0)$; – q_t — простое число; – $n \bmod q \neq 1$, для всех целых $t = 1, 2, \ldots T$, где $T \geq 11$; – $2^{254} < q < 2^{566}$ (ГОСТ Р 34.10 - 2001) В противном случае меняются либо параметры эллиптической кривой n, A или B, либо выбирается другая точка P	$x_p = (7^3 + 3 \times 7 + 7) \times$ $\times \bmod 41 = 2$ $y_p = 17$ $(17^2 \bmod 41 = 2)$ $q = 47$
4	Выбирается закрытый ключ d $(0 < d < q)$	$D = 10$

Окончание табл. 19.2

№ п/п	Основные операции	Пример
5	Определяется точка эллиптической кривой $Q(x_q, y_q)$ $Q(x_q, y_q) = d \times P(x_p, y_p)$	$x_q = 36, y_q = 20$
6	Публикуется открытый ключ [(A, B), $P(x_q, y_q)$], n, $Q(x_q, y_q)$ в специальном хранилище, где исключается возможность его подмены (общедоступном сертифицированном справочнике). Для выработки и проверки электронной цифровой подписи q является частью открытого ключа вместо n	

*) Прямой способ вычисления порядка группы точек эллиптической кривой q.

1) Рассчитываются координаты первой точки из выражения $y^2 = x^3 + Ax + B \pmod n$, т.е. $y^2 = x^3 + 3x + 7 \pmod{41}$.

Примем $x_1 = 7$, тогда $y_1^2 \bmod 41 = (7^3 + 3 \times 7 + 7) \bmod 41 = 2$, откуда $y_1 = 17$.

$P(x_p, y_p) = P(x_1, y_1) = P(7, 17)$.

2) Находим координаты второй точки. Для этого вначале вычисляется коэффициент λ:

$$\lambda = \left(\frac{3 \times x_1^2 + \alpha}{2y_1}\right) \bmod n = \left(\frac{3 \times 7^2 + 3}{2 \times 17}\right) \bmod 41 = \left(\frac{150}{34}\right) \bmod 41 \Rightarrow$$

$$34\lambda \bmod 41 = 150 \bmod 41 \Rightarrow$$

$$34\lambda \bmod 41 = 27$$

Решая последнее уравнение, получаем $\lambda = 2$ ($34 \times 2 \bmod 41 = 27$).

Координаты второй точки получаем путем удваивания первого из выражений:

$x_2 = (\lambda^2 - 2x_1) \bmod n = (2^2 - 2 \times 7) \bmod 41 = -10 \bmod 41 = 31$,

$y_2 = (\lambda(x_1 - x_2) - y_1) \bmod n = (2\,(7 - 31) - 17) \bmod 41 = -65 \bmod 41 = 17$.

3) Каждую следующую точку рассчитываем по приведенным выше формулам (19.9), (19.10), пока в знаменателе первой формулы не будет получен 0:

$$\lambda_i = \frac{y_{i-1} - y_1}{x_{i-1} - x_1} (\mathrm{mod}\, n)$$

$$\begin{cases} x_i = (\lambda_i^2 - x_1 - x_{i-1}) \bmod n \\ x_i = (\lambda_i (x_1 - x_i) - y_1) \bmod n \end{cases}$$

Получаем:

$-\lambda_3 = 0, x_3 = 3, y_3 = 24;$

$-\lambda_4 = 29, x_4 = 11, y_4 = 31;$

$-\lambda_5 = 24, x_5 = 25, y_5 = 2;$

...;

$-\lambda_{46} = 2, x_{46} = 7, y_{46} = 24.$

К полученному числу точек добавляем точку О, в результате чего q = 46 + 1 = 47. Эта точка есть результат сложения любых двух точек $P(x_i, y_i)$ и $-P(x_i, -y_i)$ и представляет собой бесконечно удаленную точку, в которой гипотетически сходятся все вертикальные кривые.

Процедура шифрования отдельного блока выполняется следующим образом (табл. 19.3) [25].

Таблица 19.3

Процедура шифрования отдельного блока (буквы)

№ п/п	*Описание операции*	*Пример*
1	Определяется десятичное представление буквы *t*	Буква «К» *t* = 12
2	Выбирается случайное число *k* (0 < *k* < *n*)	*K* = 5
3	Определяется точка $P_k(x_{pk}, y_{pk}) = kP$	$P_k(25,2)$
4	Определяется точка $Q_k(x_{qk}, y_{qk}) = kQ$	$Q_k(3,24)$
5	Вычисляется $c = (t \times x_{qk}) \bmod n$	$c = (12 \times 3) \bmod 41 = 36$
6	Шифрограмма-пара $[P_k, c]$	$[P_k(25,2),36]$

Процедура расшифрования отдельного блока выполняется согласно табл. 19.4.

Обмен ключами. Операция сложения в криптографии на основе эллиптических кривых является аналогом операции умножения по

модулю простого числа в RSA, а многократное повторное сложение — аналогом возведения в степень. Чтобы построить криптографическую систему, используя эллиптические кривые, нужно решить «трудную проблему», соответствующую разложению на множители двух простых чисел или дискретному логарифмированию.

Рассмотрим уравнение $Q = kP$, где Q, P принадлежат $E_p(a,b)$ и $k < p$. Относительно легко вычислить Q данным k и P, но трудно определить k, имея Q и P.

Таблица 19.4

Процедура расшифрования отдельного блока (буквы)

№	*Описание операции*	*Пример*
1	Вычисляется точка $D(x_d y_d) = dP_k$	D(3,24)
2	Вычисляется десятичное представление зашифрованной буквы $t = (c \times x_d^{-1}) \bmod n$, где x_d^{-1} —обратное число к x_d по модулю *n*	$x_d^{-1} = 14((3 \times 14) \bmod 41 = 1)$ $t = (36 \times 14) \bmod 41 = 12$
3	Определяется исходное сообщение по ее десятичному представлению	Буква «К»

Обмен ключами с использованием эллиптических кривых может быть выполнен следующим образом. Сначала выбирается простое число p порядка 2^{180} и параметры a и b для эллиптической кривой, заданной уравнением (19.1). Это задает эллиптическую группу точек $E_p(a, b)$. Затем в $E_p(a, b)$ выбирается генерирующая точка $G = (x_1, y_1)$. При выборе G важно, чтобы наименьшее значение n при котором $nG = O$, оказалось очень большим простым числом. Параметры $E_p(a, b)$ и G криптосистемы являются параметрами, известными всем участникам.

Обмен ключами между пользователями **A** и **B** можно провести по следующей схеме:

1. Сторона **A** выбирает целое число n_A, меньшее *n*. Это число будет ключом участника А. Затем участник **A** генерирует открытый ключ $P_A = n_A \times G$. Открытый ключ представляет собой некоторую точку из $E_p(a, b)$.

2. Точно также участник **B** выбирает личный ключ n_B и вычисляет открытый ключ P_B.

3. Участник А генерирует секретный ключ $K = n_A \times P_B$, а участник В генерирует секретный ключ $K = n_B \times P_A$.

Две формулы дают один и тот же результат, поскольку
$n_A \times P_B = n_A \times (n_B \times G) = n_B \times (n_A \times G) = n_B \times P_A$.

Чтобы взломать эту схему, противник должен будет вычислить k по данным G и kG, что предполагается трудным делом.

В качестве примера возьмем $p = 211$, $E_p(0, -4)$ и $G = (2, 2)$. Можно подсчитать, что $241G = O$. Личным ключом пользователя **A** является $n_A = 121$, поэтому открытым ключом участника **A** будет $P_A = 121\ (2, 2) = (115, 48)$. Личным ключом пользователя **B** является $n_B = 203$, поэтому открытым ключом участника **B** будет $203(2, 2) = (130, 203)$. Общим секретным ключом является $121(130, 203) = 203(115, 48) = (161, 169)$.

Обратите внимание на то, что общий секретный ключ представляет собой пару чисел. Если этот ключ предполагается использовать в качестве сеансового ключа для традиционного шифрования, то из этой пары чисел необходимо генерировать одно подходящее значение. Можно, например, использовать просто координату x или некоторую простую функцию от x.

19.3. ШИФРОВАНИЕ И ДЕШИФРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ ЭЛЛИПТИЧЕСКИХ КРИВЫХ

В специальной литературе можно найти анализ нескольких подходов к шифрованию / дешифрованию, предлагающих использование эллиптических кривых. Рассмотрим наиболее простой из подходов. Первой задачей является шифрование открытого текста сообщения m, которое должно будет пересылаться в виде координат точки P_m. Данная точка будет представлять зашифрованный текст и впоследствии будет дешифроваться.

Предположим, абонент **A** хочет отправить абоненту **B** зашифрованное сообщение. Для этого ему понадобится открытый ключ абонента **B**. Генерация открытого и закрытого ключей абонента **B** проводится по следующему алгоритму:

1. Абонент **B** выбирает эллиптическую группу $E_p(a,b)$.
2. Абонент **B** выбирает генерирующую точку на кривой $G(x_1,y_1)$.
3. Абонент **B** выбирает целое число n_B — свой личный ключ.
4. Абонент **B** вычисляет свой открытый ключ: $P_B = n_B \times G$. Обратите внимание: умножение здесь означает многократное сложение и определяется как раньше.

Секретный и открытый ключи абонента **A** вычисляются аналогичным образом.

Для того чтобы отправить абоненту **B** сообщение P_m, пользователь A выбирает случайное положительное целое число k и вычисляет шифрованный текст C_m, состоящий из пары точек

$$C_m = \{kG, P_m + kP_B\}.$$

Заметим, что сторона **A** использует открытый ключ P_B стороны **B**. Чтобы дешифровать этот шифрованный текст, **B** умножает первую точку в паре на секретный ключ **B** и вычитает результат из второй точки:

$$P_m + kP_B - n_B(kG) = P_m + k(n_B G) - n_B(kG) = P_m.$$

Пользователь **A** замаскировал сообщение P_m с помощью добавления к нему kP_B. Никто, кроме этого пользователя, не знает значения k, поэтому, хотя P_B и является открытым ключом, никто не сможет убрать маску kP_B. Однако пользователь **A** включает и сообщение, и «подсказку», которой будет достаточно, чтобы убрать маску тому, кто имеет личный ключ n_B. Злоумышленнику для восстановления сообщения придется вычислить k по данным G и kG, что представляется трудной задачей.

В качестве примера подобного шифрования рассмотрим случай $p = 751$, $E_p(-1, 188)$ (что соответствует кривой $y^2 = x^3 - x + 188$) и $G = (0, 376)$. Предположим, что пользователь **A** собирается отправить пользователю **B** сообщение, которое кодируется эллиптической точкой $P_m = (562, 201)$, и что пользователь **A** выбирает случайное число $k = 386$. Открытым ключом **B** является $P_B = (201, 5)$. Мы имеем $386(0, 376) = (676, 558)$ и $(562, 201) + 386(201, 5) = (385, 328)$. Таким образом, пользователь **A** должен послать шифрованный текст $\{(676, 558), (385, 328)\}$.

Принцип шифрования / дешифрования сообщений с использованием метода эллиптических кривых показан на рис. 19.5.

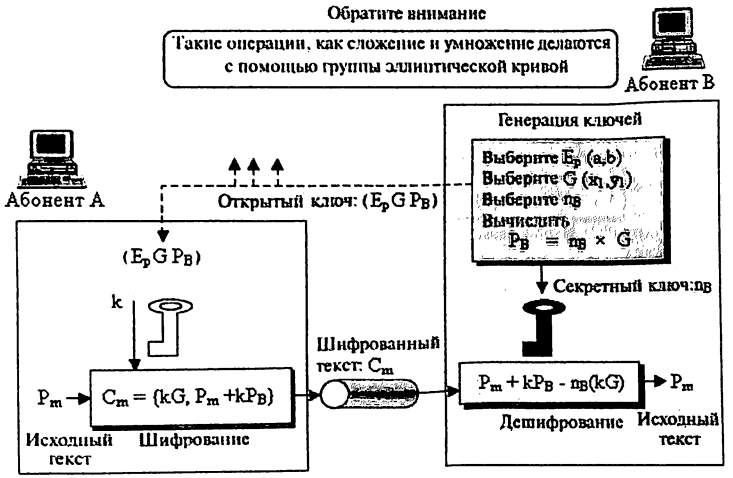


Рис. 19.5. Шифрование (дешифрование) данных с использованием эллиптических кривых

Приведенный выше способ шифрования является вариацией шифрования Эль-Гамаля. Если стойкость алгоритма шифрования Эль-Гамаля базируется на сложности решения задачи дискретного логарифмирования, то стойкость шифрования с помощью эллиптических кривых базируется на сложности нахождения множителя k точки P по их произведению, т.е. если Q = kP, то, зная P и k, довольно легко вычислить Q. Эффективное решение обратной задачи (найти k при известных P и Q) на текущий момент пока не опубликовано.

19.4. КРИПТОСИСТЕМЫ ЭЛЬ-ГАМАЛЯ, ОСНОВАННЫЕ НА ЭЛЛИПТИЧЕСКИХ КРИВЫХ

Для шифрования и дешифрования текстов с помощью эллиптических кривых используются несколько методов. Один из них состоит в том, чтобы моделировать криптосистему Эль-Гамаля, используя эллиптическую кривую в GF(p) или GF(2^n), как это показано на рис. 19.6.

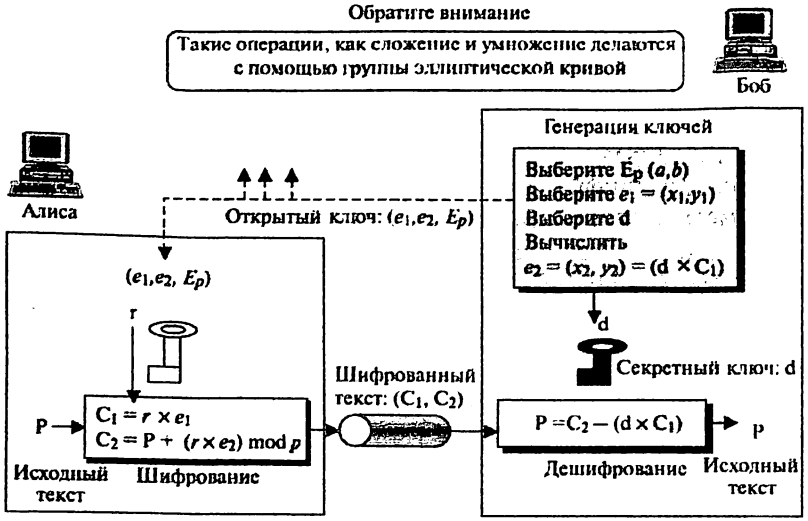


Рис. 19.6. Криптосистема Эль-Гамаля, использующая эллиптическую функцию

Генерация общедоступных и частных ключей:

1. Боб выбирает $E(a, b)$ с эллиптической кривой в GF(p) или GF(2^n).
2. Боб выбирает точку на кривой $e_1(x_1, y_1)$.
3. Боб выбирает целое число d.

4. Боб вычисляет $e_2(x_2, y_2) = d \times e_1(x_1, y_1)$. Обратите внимание: умножение здесь означает многократное сложение и определяется как раньше.
5. Боб объявляет $E(a, b)$, $e_1(x_1, y_1)$и $e_2(x_2, y_2)$ как свой открытый ключ доступа; он сохраняет d как секретный ключ.

Шифрование. Алиса выбирает точку P на кривой как ее исходный текст P. Затем она вычисляет пару точек, направляет их как зашифрованный текст:$C_1 = r \times e_1$, $C_2 = P + r \times e_2$.

Здесь можно задаться вопросом: как произвольным исходным текстом может быть точка на эллиптической кривой. Это одна из основных проблем в применении эллиптической кривой для моделирования. Алиса должна использовать соответствующий алгоритм, чтобы найти непосредственное соответствие между символами (или блоками текста) и точками на кривой.

Дешифрование. Боб после получения C_1 и C_2 вычисляет P — исходный текст, используя следующую формулу:

$$P = C_2 - (d \times C_1).$$

Знак «минус» здесь означает сложение с инверсией.

Текст P, вычисленный Бобом, — тот же, что передан Алисой, как это показано ниже:

$$P + r \times e_2 - (d \times r \times e_1) = P + (r \times d \times e_1) - (r \times d \times e_1) = P + 0 = P.$$

Обратите внимание, что результат сложения двух обратных точек на кривой — нулевая точка.

Пример. Вот тривиальный пример шифровки с использованием эллиптической кривой в GF(p).

1. Боб выбирает $E_{67}(2, 3)$ как эллиптическую кривую в GF(p).
2. Боб выбирает $e_1 = (2, 22)$ и $d = 4$.
3. Боб вычисляет $e_2 = (13, 45)$, где $e_2 = d \times e_1$.
4. Боб публично объявляет кортеж (E, e_1, e_2).
5. Алиса хочет передать исходный текст P = (24, 26) Бобу. Она выбирает $r = 2$.
6. Алиса находит точку $C_1 = (35, 1)$, где $C_1 = r \times e_1$.
7. Алиса находит точку $C_2 = (21, 44)$, где $C_2 = P \times e_1$.
8. Боб получает C_1 и C_2. Он использует 2 × C_1(35, 1) и получает (23, 25).
9. Боб инвертирует точку (23, 25) и получает точку (23, 42).
10. Боб складывает (23, 42) с C_2 = (21, 44) и получает первоначальный исходный текст P = (24, 26).

Сравнение. Ниже приводится краткое сравнение алгоритма Эль-Гамаля с его вариантом, использующим эллиптическую кривую.

1. Алгоритм Эль-Гамаля использует мультипликативную группу; вариант — эллиптическую группу.

2. Алгоритм Эль-Гамаля в качестве параметров использует числа, являющиеся членами мультипликативной группы; при применении варианта числа заменяются точками на эллиптической кривой.

3. Секретный ключ в каждом алгоритме — целое число.

4. Секретные числа, выбираемые Алисой в каждом алгоритме, — целые числа.

5. Возведение в степень в алгоритме Эль-Гамаля заменено умножением точки на константу.

6. Умножение в алгоритме Эль-Гамаля заменено сложением точек.

7. Инверсия в алгоритме Эль-Гамаля — мультипликативная инверсия в мультипликативной группе; инверсия в группе эллиптической кривой заменяется аддитивной инверсией точки на кривой.

8. Вычисление обычно легче в эллиптической кривой, потому что умножение проще, чем возведение в степень, сложение проще, чем умножение, и нахождение инверсии намного проще в группе эллиптической кривой, чем в мультипликативной группе.

Безопасность метода с использованием эллиптической кривой. Чтобы расшифровать сообщение, Ева должна найти значение r или d.

1. Если Ева знает значение r, она может использовать $P = C_2 - (r \times e_2)$ чтобы найти точку P, относящуюся к исходному тексту. Но для того чтобы найти r, Ева должна решить уравнение $C_1 = r \times e_1$. Это значит — найти две точки на кривой, C_1 и e_1. Ева должна найти множитель, который создает C_1, начиная с e_1. Эта проблема известна как проблема логарифма эллиптической кривой, единственный известный метод решения этой проблемы — алгоритм Поларда, который неосуществим, если заданы большие r и p в GF(p) или большое n в GF(2^n).

2. Если Ева знает значение d, она может использовать $P = C_2 - (d \times C_1)$, чтобы найти точку P, относящуюся к исходному тексту. Поскольку $e_2 = d \times e_1$, это тот же самый тип проблемы, что и в предыдущем пункте. Ева знает значение e_1 и e_2 — она должна найти d.

Пример генерации и проверки электронной цифровой подписи. Пусть используется эллиптическая кривая $E_{751}(-1,1)$ и генерирующая точка G = (384, 475) порядка n = 13 (13 — наибольший из делителей порядка кривой N = 728). Предположим, абонент подписывает личным секретным ключом d = 12 сообщение, хэш-свертка которого равна e = 12.

Пусть абонент, подписывающий сообщение, выбрал случайное k = 3. Тогда он вычисляет $kG = (x,y) = 3\ (384, 475) = (596, 318)$ и затем $r = x \bmod n = 596 \bmod 13 = 11$.

Используя расширенный алгоритм Евклида, определяем $z = k^{-1} \bmod n = 3^{-1} \bmod 13 = 9$ (так как $3 \cdot 9 = 27 \equiv 1 (\mathrm{mod}\, 13)$). Наконец, $s = z(e+dr) \bmod n = 9 \cdot (12+12 \cdot 11) \bmod 13 = 9$. Таким образом, $(r,s) = (11, 9)$ – цифровая подпись данного абонента для сообщения.

Пусть теперь необходимо проверить подлинность данной подписи. Открытый ключ абонента, подписавшего сообщение, равен $Q = dG = 12 \cdot (384, 475) = (384, 276)$. Проверка подписи начинается с проверки условий $1 \le r \le n-1$, $1 \le s \le n-1$ – в данном случае они соблюдаются. Затем последовательно вычисляем $v = s^{-1} \bmod n = 9^{-1} \bmod 13 = 3$, $u_1 = ev \bmod n = 12 \cdot 3 \bmod 13 = 10$ и $u_2 = 11 \cdot 3 \bmod 13 = 7$. Находим точку $X = u_1 G + u_2 Q = 10\ (384, 475) + 7\ (384, 276) = (596, 318)$. Наконец, сравниваем значения $r = 11$ и $x \bmod n = 596 \bmod 13 = 11$ – они совпадают, следовательно, подпись действительная.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Баричев С.Г., Гончаров В.В., Серов Р.Е.* Основы современной криптографии. — М.: Горячая линия — Телеком, 2001.
2. *Бернет С., Пэйн С.* Криптография. Официальное руководство RSA Security. — М.: Бином, 2002.
3. *Брассар Ж.* Современная криптология. — М.: Полимед, 1999.
4. *Виноградов И.М.* Основы теории чисел. — М.: Наука, 1972.
5. ГОСТ 28147–89. Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования информации.
6. ГОСТ Р 34.10–2001. Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронно-цифровой подписи.
7. ГОСТ Р 34.10–94. Информационная технология. Криптографическая защита информации. Процедуры выработки и проверки электронной цифровой подписи на базе асимметричного криптографического алгоритма.
8. ГОСТ Р 34.11–94. Информационная технология. Криптографическая защита информации. Функция хэширования.
9. ГОСТ Р 51583. Защита информации. Порядок создания автоматизированных систем в защищенном исполнении.
10. Гостехкомиссия России — точка зрения на техническую защиту информации // JetInfo. 1999. — № 11. — С. 2–12.
11. *Грушо А.А., Тимонина Е.Е.* Теоретические основы защиты информации. — М.: Изд-во Агентства «Яхтсмен», 1996.
12. *Грэм Р., Кнут Д., Паташник О.* Конкретная математика. — М.: Мир, 1998.
13. *Девянин П.Н.* и др. Программно-аппаратные средства обеспечения информационной безопасности. Теоретические основы компьютерной безопасности / П.Н. Девянин, О.О. Михальский, Д.И. Правиков, А.Ю. Щербаков. М.: Радио и связь, 2000.
14. *Жельников В.Г.* Криптография от папируса до компьютера. — М.: ABF, 1996.
15. Журнал «Cryptologia» (с 1977).
16. Журнал «Journal of Cryptology» (с1988 г.).
17. *Завгородний В.И.*. Комплексная защита информации в компьютерных системах — М.: Логос, 2001.
18. *Зегжда Д.П., Ивашко А.М.* Основы безопасности информационных систем. — М.: Горячая Линия — Телеком, 2000.

19. *Зима В., Молдовян А.А., Молдовян Н.А.* Компьютерные сети и защита передаваемой информации. — СПб.: Изд-во СПб. Университета, 1998.
20. *Кнут Д.* Искусство программирования для ЭВМ. Т. 2: Получисленные алгоритмы. — М.: Мир, 1977.
21. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ. — М.: МЦНМО, 1999.
22. *Малюк А.А., Пазизин С.В., Погожин Н.С.* Введение в защиту информации в автоматизированных системах. — М.: Горячая Линия — Телеком, 2001.
23. *Мао В.* Современная криптография: теория и практика. — СПб.: Вильямс, 2005.
24. *Молдовян А.А., Молдовян Н.А., Гуц Н.Д., Изотов Б.В.* Криптография: скоростные шифры. — СПб.: БХВ, 2002.
25. *Молдовян А.А., Молдовян Н.А., Советов Б.Я.* Криптография. — СПб.: Лань, 2000.
26. *Нечаев В.И.* Элементы криптографии (Основы теории защиты информации) / под ред. В.А. Садовничего. — М.: Высшая школа, 1999.
27. *Ноден П., Ките К.* Алгебраическая алгоритмика. — М.: Мир, 1999.
28. Обзор криптотехнологий // JetInfo. — 2001. — № 3. — 16 с.
29. 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 // Сб. руководящ. документов по защите информации от несанкционированного доступа. — М.: ГТК при Президенте РФ, 1998.
30. Руководящий документ. Защита от несанкционированного доступа к информации. Термины и определения // Сб. руководящ. документов по защите информации от несанкционированного доступа. — М.: ГТК при Президенте РФ, 1998.
31. *Саломаа А.* Криптография с открытым ключом. — М.: Мир, 1995.
32. *Соболева Т.А.* Тайнопись в истории России. — М.: Международные отношения, 1994.
33. *Соболева Т.А.* История шифровального дела в России. — М.: ОЛМА-Пресс, 2002.
34. *Столлингс В.* Криптография и защита сетей. — М., СПб., Киев: Вильямс, 2001.
35. *Шнайер Б.* Прикладная криптография. — М.: Триумф, 2002.

36. *Kahn D.* The Codebreakers. — N.Y.: Scribner, 1996. [имеется сокр. перевод: Кан Д. Взломщики кодов. — М.: Центрполиграф, 2000.].

37. *LevyS.*Crypto. — N.Y.: Viking, 2001.

38. *Панасенко С.* Аппаратное шифрование для ПК // Безопасность. — 2003. — № 4 (56).

39. *Жданов О.Н., Чалкин В.А.* Эллиптические кривые и их применение в криптографии: учеб. пособие. — Красноярск, Сиб. гос. аэрокосмич. ун-т., 2011.

40. НОУ Интуит. Математика криптографии и теория шифрования [Электронный ресурс]. — URL: http://window.edu.ru/resource/555/66555.

41. 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.

42. *Жданов О.Н., Золотарев В.В.* Методы и средства криптографической защиты информации: учеб. пособ. — Красноярск, СибГАУ,2007.

43. *Галуев Г.А.* Математические основы криптологии: учебно-методич. пособ. — Таганрог: Изд-во ТРТУ, 2003.

44. *Безбогов А.А.* Методы и средства защиты компьютерной информации: учеб. пособие / А.А. Безбогов, А.В. Яковлев, В.Н. Шамкин. — Тамбов: Изд-во Тамб. гос. техн. ун-та, 2006.

45. *Воронков Б.Н., Щеголеватых А.С.* Элементы теории чисел и криптозащиты. — Воронеж: Изд.-полиграф. центр Воронежского госуд. ун-та. 2008.

46. Введение в криптографию / под общ. ред. В.В. Ященко. 4-е изд., доп. — М.: МЦНМО, 2012.

47. *Кан Д.* Взломщики кодов / пер. А. Ключевский — М.: «Центрполиграф», 2000.

48. Базовые принципы информационной безопасности вычислительных сетей: учеб. пособие для студентов, обучающихся по специальностям 08050565, 21040665, 22050165, 23040165 / А.А. Гладких, В.Е. Дементьев. — Ульяновск: УлГТУ, 2009.

49. О развитии криптографии в XIX веке / А. В. Бабаш, Ю.И. Гольев, Д. А. Ларин, Г. П. Шанкин // журнал «Конфидент» – 2003. – № 5 (http://ftp.agentura.ru/press/about/jointprojects/confident/krypto19/)

50. *Дориченко С.А., Ященко В.В.* 25 этюдов о шифрах — М: ТЕИС, 1994.

51. *Отставнов М.* Крипто: от тайного знания к знанию о тайне // Домашний компьютер. — 2001. — № 10. — с. 56–59.

52. *Жданов О.Н., Куденкова И.А.* Криптоанализ классических шифров. Лабораторный практикум для студентов, обучающихся по специальностям «Комплексное обеспечение информационной безопасности автоматизированных систем» и «Информационная безопасность телекоммуникационных систем». — Красноярск, 2008.

53. *Борисова С.Н.* Методы и средства криптографической защиты данных в вычислительных системах. Часть 2. — Пенза, Пензенский госуд. технологич. ун-т, 2013.

54. *Бабенко Л.К., Мишустина Е.А.* Методическое пособие по изучению современных методов криптоанализа по курсу «Криптографические методы и средства обеспечения защиты информации». — Таганрог: Изд-во ТРТУ, 2003.

55. *Алферов А.П., Зубов А.Ю., Кузьмин А.С., Черемушкин А.В.* Основы криптографии. Учебное пособие — М.: Гелиос АРВ, 2002.

СОДЕРЖАНИЕ

По вопросам приобретения книг обращайтесь:
Отдел продаж «ИНФРА-М» (оптовая продажа):
127282, Москва, ул. Полярная, д. 31В, стр. 1
Тел. (495) 280-15-96; факс (495) 280-36-29
E-mail: books@infra-m.ru

•

Отдел «Книга–почтой»:
тел. (495) 280-15-96 (доб. 246)

Учебное издание

КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА ИНФОРМАЦИИ

Учебное пособие

Подписано в печать 12.02.2019.
Формат 60×90/16. Гарнитура Times.
Бумага офсетная. Печать цифровая.
Усл. печ. л. 20,06. Уч.-изд. л. 21,16
Доп. тираж 30 экз. Заказ № 01656.
Цена свободная.

ТК 654164 — 1018903 — 120219

ООО «Издательский Центр РИОР»
127282, Москва, ул. Полярная, д. 31В.
Тел.: (495) 280-38-67.
info@riorp.ru https://riorpub.com

ООО «Научно-издательский центр ИНФРА-М»
127282, Москва, ул. Полярная, д. 31В, стр. 1.
Тел.: (495) 280-15-96. Факс: (495) 280-36-29.
E-mail: books@infra-m.ru http://www.infra-m.ru

Отпечатано в типографии
ООО «Научно-издательский центр ИНФРА-М»
127282, Москва, ул. Полярная, д. 31В, стр. 1
Тел.: (495) 280-15-96, 280-33-86. Факс: (495) 280-36-29

КНИГИ ИНФРА-М ПОЧТОЙ

ООО «Научно-издательский центр ИНФРА-М» осуществляет рассылку книг по почте на территории Российской Федерации.

Информацию о наличии книг можно получить, воспользовавшись прайс-листом Научно-издательского центра ИНФРА-М, который можно бесплатно заказать и получить по почте. Также информацию о книгах можно посмотреть на сайте http://www.infra-m.ru в разделах «Прайс-лист» и «Иллюстрированный каталог».

Для оформления заказа необходимо прислать заявку, где следует указать:

- для организаций:

название, полный почтовый адрес, банковские реквизиты (ИНН/КПП), номера телефона, факса, контактное лицо (получателя), наименование книг, их количество;

- для частных лиц:

Ф.И.О., полный почтовый адрес, номер телефона для связи, наименование книг, их количество.

При заполнении заявки необходимо указывать код книги что значительно ускорит оформление Вашего заказа.

Заказ оформляется по оптовым ценам, указанным в прайс-листе. На основании заявки Вам будет выставлен счет на имеющуюся в наличии литературу с учетом почтовых расходов (при сумме заказа свыше 5000 рублей, предоставляются скидки).

Произвести оплату вы можете:

по безналичному расчету:

перечислите сумму на расчетный счет ООО «Научно-издательский центр ИНФРА-М»;

за наличный расчет:

в отделении Сбербанка: по квитанции-извещению на сумму счета, где получатель платежа - ООО «Научно-издательский центр ИНФРА-М».

В течение 5 рабочих дней с момента зачисления денежных средств на расчетный счет заказ будет подобран и отправлен по указанному в заявке адресу с сопроводительными документами (счет-фактура, накладная).

Заявку можно прислать по факсу, электронной почте или по адресу:

127282, г. Москва, ул. Полярная, д. 31В, стр. 1

Телефон: (495) 280-1596 (доб.: 246, 248)

Факс: (495) 280-1596 (доб. 232)

E-mail: podpiska@infra-m.ru; poster3@infra-m.ru

Книги Научно-издательского центра ИНФРА-М в книжных магазинах:

МОСКВА

Московский дом книги на Арбате (сеть магазинов)
ул. Новый Арбат, 8
тел.: (495) 789-35-91
www.mdk-arbat.ru

Молодая гвардия
ул. Большая Полянка, 28
тел.: (499) 238-50-01, (499) 238-50-01;
ул. Братиславская, 26М
тел.: (495) 346-99-00
www.bookmg.ru

Библио-Глобус
ул. Мясницкая, д. 6/3, стр. 1
тел.: (495) 781-19-12, 781-19-00
www.biblio-globus.ru

Медведково
Заревый пр-д. 12
тел.: (499) 476-16-90, (495) 656-92-97
www.bearbooks.ru

ТДК «Москва»
ул. Тверская, д. 8, стр. 1
тел.: (495) 629-64-83, 797-87-71
www.moscowbooks.ru

САНКТ-ПЕТЕРБУРГ

Дом книги
Невский пр-т, д. 28, литера А
тел.: 8 (812) 448-83-55
www.spbdk.ru

Буквоед
Парк культуры и чтения «Буквоед»
Невский пр-т, 46;
Лиговский пр-т, 10
(гостиница «Октябрьская»)
тел.: 8 (812) 601-06-01
www.bookvoed.ru

ИНТЕРНЕТ-МАГАЗИНЫ

http://www.ozon.ru
http://www.neobook.ru
http://www.bookler.ru
http://www.setbook.ru
http://www.chitai-gorod.ru
http://www.colibri.ru
http://www.urait-book.ru
http://www.bolero.ru
http://www.chaconne.ru
http://my-shop.ru

499200 сум