В. И. КОРЖИК
В. А. ЯКОВЛЕВ

ОСНОВЫ КРИПТОГРАФИИ

учебное пособие

Санкт-Петербург, 2016

В. И. Коржик, В.А.Яковлев

ОСНОВЫ КРИПТОГРАФИИ

Учебное пособие для обучающихся
по направлениям подготовки бакалавров и магистров:
10.04.01, 10.03.01 «Информационная безопасность», 43.03.01 «Сервис»,
11.03.02, 11.04.02 «Инфокоммуникационные технологии и системы связи»,
а также по специальности 210403 «Защищенные системы связи»

Санкт-Петербург
ИЦ Интермедия
2016

УДК 004.056(075.8)
ББК 32.973я73
К66

Рецензенты:
доктор технических наук, профессор кафедры ТОСиР СПбГУТ *Р. Р. Биккенин*;
кандидат технических наук, ведущий специалист по защите информации
ЗАО «Научные приборы» *Б. В. Изотов*

К66 Основы криптографии: учебное пособие / В. И. Коржик, В. А. Яковлев. – СПб., ИЦ Интермедия, 2016. – 296 с. : илл.

ISBN 978-5-89160-097-3

Книга предназначена для помощи в изучении курса “Основы криптографии “ и "Основы криптографии с открытым ключом" студентами, обучающимся по направлениям подготовки: 090900 "Информационная безопасность", 100100 "Сервис", 210700 "Инфокоммуникационные технологии и системы связи", а так же для специалистов по специальности 210403 "Защищенные системы связи". Она может быть также полезна всем самостоятельно изучающим вопросы криптографии и интересующимся ее практическим применением.

В первой части описаны идеальные (совершенные) криптосистемы , блоковые и потоковые шифры. Представлены основные методы криптоанализа данных шифров, а также способы их построения устойчивые к этим методам. Для пояснения современных алгоритмов криптографии, в книге кратко излагается математический аппарат теории конечных полей и булевых функций. Рассматриваются модификации блоковых шифров и методы многократного шифрования. Представлена техника безусловно стойкой аутентификации сообщений и вычислительно стойкой аутентификации на основе использования блоковых шифров. Описаны используемые на практике стандарты шифрования (DES, ГОСТ-28147-89, ГОСТ Р 34.12-2015, AES). Значительное внимание уделено построению потоковых шифров, включая основные принципы их разработки. В качестве примера рассмотрен шифр A5/1, используемый в стандарте мобильной связи GSM.

Во второй части рассматриваются несимметричные криптосистемы. Изложению основного материала предшествует краткое описание необходимого для этого математического аппарата теории чисел. Далее подробно описываются криптосистемы РША, Рабина, Диффи-Хеллмана, Эль-Гамаля и Мак-Элис. Излагается подход к методам построения криптосистем на основе эллиптических кривых. Рассматривается техника выполнения цифровой подписи. Приводится описание хеш-функций и ряда криптографических протоколов.

Часть III посвящена управлению ключами в криптографических системах, где рассмотрены различные алгоритмы распределения ключей для симметричных и асимметричных криптосистем. Приведены некоторые стандарты распределения ключей в сетевых протоколах.

При написании пособия использованы последние открытые научные результаты в области криптографии, а также учтён опыт постановки аналогичных курсов в отечественных и зарубежных университетах.

Для понимания основного содержания книги достаточно знания основ теории вероятностей в объеме программ технических университетов.

УДК 004.056(075.8)
ББК 32.973я73

ISBN 978-5-89160-097-3

СОДЕРЖАНИЕ

ПРЕДИСЛОВИЕ

Слово криптография в переводе с греческого означает тайное письмо. Примитивная криптография возникла с появлением письменности у разных народов. (Если у людей появлялась потребность что-либо сообщать друг другу, то появлялась и потребность скрывать это от некоторой части сообщества.) Значительная потребность в криптографии открылась сразу же с возникновением государственной переписки. Однако в данном пособии не будет даваться история развития криптографии с древнейших времен и до наших дней, поскольку этот материал имеет такое же отношение к содержанию настоящего курса, как, скажем, история математики к курсу математики. Увлекательная и временами драматичная история криптографии заслуживает отдельного рассмотрения. Интересующиеся этим вопросом могут прочитать специально посвященную ей книгу [1]. Наша же задача состоит в изложении основных понятий современной криптографии в ориентации на студентов, обучающихся по направлениям подготовки: 10.04.01, 10.03.01 «Информационная безопасность», 43.03.01 «Сервис», 11.03.02, 11.04.02 «Инфокоммуникационные технологии и системы связи», а также на специалистов по специальности 210403 «Защищенные системы связи».Это не простая задача, поскольку,с одной стороны, при обучении по этим специальностям не ставится задача подготовить будущих профессиональных криптографов, а с другой стороны, это не должно быть лишь поверхностным изложением основных определений и параметров современных криптосистем. Обучаемые должны усвоить их основные функции, возможные нападения на них со стороны злоумышленников, а также принципы построения современных алгоритмов шифрования, аутентификации и обеспечения целостности сообщений. Тут уместно привести известное высказывание А. Эйнштейна: «Все нужно делать так просто, как это возможно, но не проще...».

Очевидно, что невозможно изучать стойкость криптосистем без рассмотрения методов их криптоанализа (т. е. дешифрования без знания ключа). С другой стороны, криптоанализ требует весьма обширных знаний в области математики и вычислительной техники, что выходит за рамки базовых дисциплин для обучаемых студентов. Поэтому в данном пособии рассматриваются лишь некоторые (хотя и вполне современные) методы криптоанализа, которые помогают пояснить необходимость тех или иных усложнений криптоалгоритмов; последнее, хотя и уменьшает быстродействие шифров, является вполне оправданными с точки зрения обеспечения их стойкости. Значительное внимание в настоящем пособии уделяется побочным атакам на шифры, поскольку они часто не требуют для своего пояснения привлечения серьезного математического аппарата, однако пренe-

брежение этими атаками приводит к простому дешифрованию криптосистем без знания ключа.

Пособие состоит из трех частей. Часть I посвящена традиционной (симметричной или одноключевой) криптографии, включая также некоторые вопросы аутентификации. Часть II посвящена несимметричной (двухключевой) криптографии, называемой иначе криптографией с открытым ключом, а также вопросам выполнения цифровой подписи и криптографических протоколов. Часть III рассматривает различные протоколы распределения ключей, используемых в криптосистемах, изложенных в первых двух частях. Особое внимание уделяется подтверждению подлинности и актуальности этих ключей.

Для понимания материала, изложенного в настоящем пособии, достаточно знаний по дисциплинам математики, теории линейных цепей и вычислительной техники, за исключением таких разделов, как теория чисел и теория конечных полей. Последние в необходимом объеме излагаются в соответствующих разделах настоящего пособия.

Знания, полученные студентами на лекциях, закрепляются при выполнении ими компьютерных лабораторных работ, которые охватывают абсолютно все разделы курса и построены таким образом, чтобы студенты смогли достаточно свободно варьировать параметры криптосистем и исследовать вопросы именно криптографии, не отвлекаясь на составление программ.

Основная терминология будет дана в ходе изложения последующих разделов. Здесь же дадим лишь несколько самых общих определений. Это законные (или легальные) пользователи криптосистемы, которые обладают секретными ключами, незаконные или нелегальные пользователи, которые не обладают этими ключами и стремятся их найти, а также прочитать зашифрованное сообщение или нарушить его целостность (т. е. подделать его или фальсифицировать авторство), а также нарушить секретность выполняемых протоколов. Криптоаналитиком будем называть пользователя, который пытается разработать методы для нарушения секретности криптосистемы. (Заметим, что криптоаналитик не обязательно является недружественным пользователем, поскольку он может применять свои методы для проверки стойкости существующих или разрабатываемых легальных криптосистем.) Попытку дешифрования (криптоанализа) криптосистемы без знания ключа, попытку подделки цифровой подписи или нарушения секретности криптопротокола будем для краткости называть ее взломом или атакой на криптосистему. Нарушитель называется пассивным, если он только пытается прочитать сообщение, и активным, если он стремится изменить (подделать) сообщение или криптопротокол.

Списки современной литературы по криптографии, которая в той или иной степени использовалась при написании данного пособия, приведены в конце каждой из частей. К сожалению, ни одна из приведенных там книг

не может в полной мере претендовать на роль учебного пособия по курсу «Основы криптографии», поскольку некоторые из них содержат недостаточно материала, тогда как другие – избыточный и чересчур сложный материал для студентов, не специализирующихся в области криптоанализа.

При подготовке пособия был использован личный опыт чтения лекций по подобным дисциплинам как в СПбГУТ, так и в зарубежных университетах США, Франции, Мексики, Южной Кореи и Сирии. Ссылки на разделы, рисунки, формулы, таблицы и литературу даны в пределах каждой из частей, а при перекрестных ссылках указаны номера частей.

ЧАСТЬ I

СИММЕТРИЧНЫЕ КРИПТОСИСТЕМЫ

1. ВВЕДЕНИЕ В КРИПТОГРАФИЮ

1.1. Модель криптосистемы

Рассмотрим сначала такую функцию криптосистем, как обеспечение *конфиденциальности*, т.е. невозможности чтения передаваемых или хранимых сообщений для всех пользователей телекоммуникационных (компьютерных) сетей, за исключением пользователей, имеющих на это разрешение, т. е. авторизованных пользователей. Помимо функции конфиденциальности криптосистемы обеспечивают выполнение и других функций, например *аутентификации* (т. е. обеспечения подлинности сообщений и пользователей), функции выполнения *криптографических протоколов*, а также функции распределения ключей, которые будут рассмотрены во второй и в третьей частях книги.

Очевидно, что функция конфиденциальности обеспечивается при помощи различных методов *шифрования* и *дешифрования*. Для изучения этой функции рассмотрим математическую модель произвольной криптосистемы:

$$E = f\left(M, k_{\text{ш}}\right),\ M = g\left(E, k_{\text{д}}\right),$$

где M – шифруемое сообщение;

E – результат шифрования (*криптограмма*);

$k_{\text{ш}}$ – ключ шифрования;

$k_{\text{д}}$ – ключ дешифрования;

$f(\cdot)$, $g(\cdot)$ – функции, определяющие алгоритмы шифрования и дешифрования соответственно, зависящие от ключей шифрования и дешифрования.

В книге будут рассматриваться цифровые сообщения, так как это самый распространенный вид представления информации в настоящее время. Соответственно будут рассматриваться цифровые криптограммы и цифровые ключи, которые могут быть представлены в виде последовательности символов конечного алфавита (например, двоичного).

1.2. Принцип Керхгоффа [2]

Согласно этому принципу предполагается, что в данной модели любому пользователю известно все, за исключением ключа дешифрования $k_{\text{д}}$.

Основные типы криптографических атак:

1) атака только при известной криптограмме E;

2) атака при известной криптограмме E и соответствующей ей части сообщения M;

3) атака при специально выбранном сообщении и соответствующей ему криптограмме.

1.3. Типы криптосистем

Криптосистемы классифицируются:

1) *по типу ключа*: симметричные ($k_{ш} = k_{д} = k$) и несимметричные криптосистемы ($k_{ш} \neq k_{д}$). Несимметричные называют также криптосистемами с открытым ключом. В этой части книги мы будем рассматривать только симметричные криптосистемы;

2) *по способу формирования криптограммы из сообщения*: блоковые криптосистемы, потоковые криптосистемы.

Криптосистема называется *блоковой*, если каждый блок сообщения определенной длины шифруется по одному и тому же правилу; блок криптограммы имеет обычно ту же длину, что и блок сообщения (рис. 1.1).

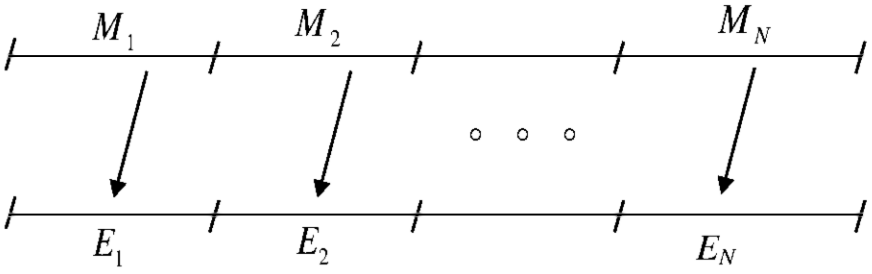


Рис. 1.1. Блоковое шифрование

Криптосистема называется потоковой, если каждый очередной символ криптограммы вырабатывается независимо по очередному символу сообщения, сучетом различных правил, зависящих от номера символа и ключа (рис. 1.2).

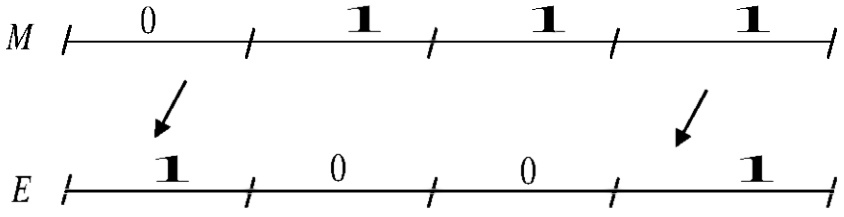


Рис. 1.2. Потоковое шифрование

Частным случаем потокового шифра является *шифр гаммирования*:

$$E_i = M_i \oplus \gamma_i(k),$$

где γ_i – функция, зависящая от ключа k; $\oplus$ – означает суммирование по $\mathrm{mod}\,2$ (рис. 1.3).

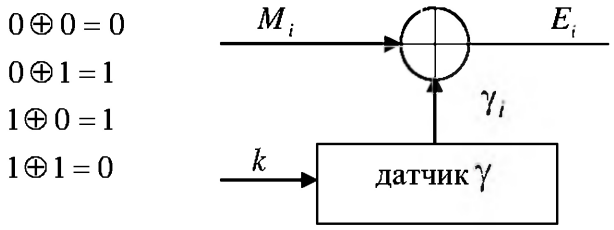


Рис. 1.3. Шифрование гаммированием

В действительности скорее можно говорить не о типе шифра, а о модификации в виде блокового или потокового шифров, поскольку, как мы увидим далее, один и тот же алгоритм шифрования можно преобразовать как в блоковый, так и в потоковой шифр.

Постойкости шифры условно делят на два класса:

а) *идеальные (совершенные, безусловно стойкие)*;

б) *вычислительно стойкие*.

Идеальные шифры обеспечивают невозможность чтения сообщения без знания ключа при любом вычислительном ресурсе у атакующего, в отличие от вычислительно стойких, которые нельзя вскрыть, только если вычислительный ресурс ограничен по времени или по объему.

1.4. Влияние ошибок в криптограмме на дешифрование

В некоторых случаях криптограмма подвержена возникновению случайных ошибок $E' = E \oplus e$, где появляется образец ошибок $e = (e_1, e_2, \ldots, e_i, \ldots)$, $e_i = 0$ (если нет ошибок), $e_i = 1$ (если есть ошибки) в i-м символе.

Пусть t – число ошибок, которые произошли в период времени передачи криптограммы, $t = |e|$, где $|e|$ – вес образца ошибок e. После дешифрования количество ошибок может измениться: $M' = g(E', k), M' = M \oplus e'$, $|e'| = t'$.

Если $t = t'$, то говорят, что это шифр *без размножения ошибок*, а если $t' > t$, то это шифр *с размножением ошибок*.

2. ИДЕАЛЬНЫЕ (БЕЗУСЛОВНО СТОЙКИЕ) КС

2.1. Необходимые и достаточные условия для построения идеальных КС

Как было отмечено ранее, *идеальные* КС обладают тем свойством, что если ключ не известен, то знание криптограммы E не дает никаких преимуществ, и лучшим способом их дешифрования является простое угадывание сообщения.

Строгое математическое определение идеальных КС:

$$P\left(M_i \middle| E_j\right) = P\left(M_i\right), \text{ для любых } (i, j),$$

где $P\left(M_i \middle| E_j\right)$ – условная вероятность того, что именно сообщение M_i было зашифровано криптограммой E_j;

$P\left(M_i\right)$ – априорная (при неизвестной криптограмме) вероятность присутствия сообщения M_i.

Эквивалентное определение:

$$I\left(E_j, M_i\right) = 0$$

(шенноновская информация, содержащаяся в криптограмме E_j о сообщении M_i, равна нулю).

Пример идеальной КС

Пусть $\bar{M}, \bar{K}, \bar{E} \in (0,1)^N$ – двоичные цепочки длины *N*. Криптограмма $\bar{E} = \bar{M} \oplus \bar{K}$, где каждый бит сообщения складывается с каждым битом ключа по $\mathrm{mod}\,2$, имеет вид

$$\begin{array}{rl} \bar{M} = & 000111\ldots \\ \bar{K} = & 100101\ldots \\ \hline \bar{E} = & 100010\ldots \end{array}$$

Дешифрование выполняется следующим образом: $\bar{M} = \bar{E} \oplus \bar{K}$.

Теорема 1. *Если все варианты ключа выбирать равновероятными, т.е.* $P\left(\bar{K}\right) = \dfrac{1}{2^N}$, *то такая система будет идеальной КС (ИКС).*

Доказательство.

Необходимо доказать, что выполнено условие $P\left(M_i \middle| E_j\right) = P\left(M_i\right)$.

Выразим $P\left(M_i \middle| E_j\right)$ по формуле Байеса [3]:

$$P\left(M_i \middle| E_j\right) = \frac{P\left(E_j \middle| M_i\right) \cdot P\left(M_i\right)}{P\left(E_j\right)}; \tag{2.1}$$

$$P\left(E_j \middle| M_i\right) = P\left\{K = E_j \oplus M_i\right\} = \frac{1}{2^N}, \tag{2.2}$$

где $P\left\{K = E_j \oplus M_i\right\}$ – вероятности выбора ключа*K*.

Подставляя (2.2) в (2.1), получим

$$P\left(M_i \middle| E_j\right) = \frac{1}{2^N} \cdot \frac{P\left(M_i\right)}{P\left(E_j\right)}. \tag{2.3}$$

По формуле полной вероятности [3] определим

$$P\left(E_j\right) = \sum_i P\left(M_i\right) \cdot P\left(E_j \middle| M_i\right) = \frac{1}{2^N} \cdot \sum_i P\left(M_i\right) = \frac{1}{2^N}. \tag{2.4}$$

Подставив (2.4) в (2.3), получаем

$$P\left(M_i \middle| E_j\right) = \frac{1/2^N}{1/2^N} \cdot P\left(M_i\right) = P\left(M_i\right).$$

Данный шифр называют также *одноразовым.* Это подчеркивает, что нельзя использовать шифр (т.е. одну и ту же гамму) более одного раза.

Приведенный выше пример показывает, что ИКС может быть реализована достаточно просто, однако в приведенном выше примере длина ключа оказалась равной длине сообщения, что, безусловно, является неудобным для практического использования. Возникает вопрос – можно ли построить ИКС с меньшим расходом ключа? Ответ дает теорема 2.

Теорема 2. Необходимое условие существования ИКС. *Число возможных ключей, используемых в ИКС, должно быть не меньше числа сообщений, которые можно зашифровать на этих ключах.*

Доказательство.

По определению ИКС должно выполняться условие $P\left(M_i \middle| E_j\right) = P\left(M_i\right)$.Из этого условия следует, что

$$P\left(E_j \middle| M_i\right) = \frac{P\left(M_i \middle| E_j\right) \cdot P\left(E_j\right)}{P\left(M_i\right)} = P\left(E_j\right). \tag{2.5}$$

Пусть $\tilde{E}$ некоторая криптограмма, которая может появиться в ИКС, т.е. $P\left(\tilde{E}\right) > 0$. Тогда из (2.5) получаем, что $P\left(\tilde{E} \middle| M_i\right) > 0$ для любого M_i (рис. 2.1).

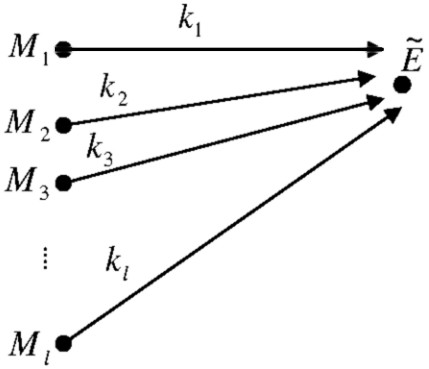


Рис. 2.1.
Часть графического описания (графа) КС

Из соображений возможности однозначной дешифровки следует, что все ключи, ведущие от разных сообщений к одной криптограмме, должны быть разными. Отсюда и вытекает утверждение теоремы о том, что число ключей должно быть равно или более числа возможных сообщений.

Пусть $\bar{M} \in \{0,1,\ldots,m-1\}^n$, $K \in \{0,1,\ldots,L-1\}^N$, $\bar{E} \in \{0,1,\ldots,m-1\}^n$, т. е. это цепочки длины n, N, n соответственно, с символами из алфавитов, указанных в фигурных скобках.

Тогда получаем следующее условие теоремы в этих обозначениях:

$$L^N \geq m^n \Rightarrow N \geq \frac{n \log m}{\log L}.$$

Видно, что длина ключа оказалась пропорциональной длине сообщения. Существует метод, позволяющий уменьшить длину ключа, но не имеющий прямого отношения к криптографии. Он связан с возможностью сжатия сообщения. На практике сообщение предварительно сжимается,а затем уже шифруется ключом, длина которого равна длине сжатого файла.

Из теории информации [4] известно, что возможность сжатия определяется энтропией источника сообщений $H(M)$, и тогда число типичныхпоследовательностей источника равно $2^{nH(M)}$, в то время как нетипичные последовательности имеют вероятность, близкую к 0. Следовательно, необходимое условие ИКС можно записать в следующем виде:

$$L^N \geq 2^{nH(M)} \Rightarrow N \geq \frac{n \cdot H(M)}{\log_2 L}.$$

Вывод. Ни при каких способах построения ИКС нельзя получить длину ключа, которая не возрастала бы пропорционально длине сообщения. Используя сжатие сообщений, можно добиться лишь уменьшения коэффициента пропорциональности. Таким образом, наилучшая ИКС имеет следующий вид, приведенный на рис. 2.2.

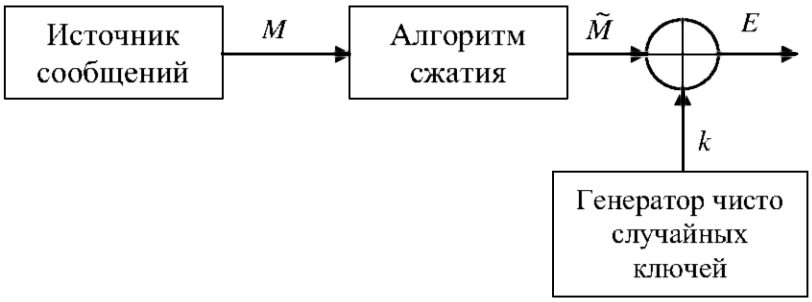


Рис. 2.2. Реализация идеальной КС

Итак, ввиду того что в ИКС необходима большая длина ключа, она оказывается малопригодной для массового использования, но может быть выбрана для привилегированных пользователей.

2.2. Методы распределения ключевых потоков в условиях возможного перехвата

Хотя распределение длинных цепочек ключей для ИКС представляется на первый взгляд нереализуемой задачей, тем не менее в последнее десятилетие появился ряд теоретических работ, и даже попыток практической реализации решения этой задачи с использованием телекоммуникационных каналов связи в условиях пассивного, или даже активного перехвата всей информации, передаваемой по этим каналам (причем, конечно, предполагается, что легальные пользователи не имеют абсолютно никаких предварительно распределенных ключевых данных, поскольку в противном случае задача оказалась бы тривиальной).

Возможность решения данной задачи основывается на принципиальных различиях между каналами легальных и нелегальных пользователей, причем важнейшую роль здесь играет фактор случайности сигналов и параметров этих каналов. Далее будут представлены три основных подхода, причем в самом упрощенном изложении. Те, кто заинтересуется данными методами, смогут найти более полное описание в цитируемой далее литературе. Заметим, что такое направление получило даже специальное название *«Бесключевой криптографии» («Keyless cryptography»* [30]*)*, причем под этим понимается тот факт, что легальные пользователи обеспечивают конфиденциальность обмена сообщениями, не имея предварительного распределения секретных ключей.

2.2.1. Распределение секретных ключей по каналам с шумом

Предположим, что между парой легальных пользователей A и B имеется канал связи с шумом (для простоты двоичный симметричный канал без памяти ДСК [4] с вероятностью ошибки символа P_m), тогда как перехватчик E может получить всю информацию, которой обмениваются **A** и **B** также по каналу с шумом (для простоты ДСК с вероятностью ошибки символа P_w).

Тогда, если выполнено условие $P_m<P_w$ (т.е. канал перехвата более шумный, чем основной канал), то существует достаточно простой протокол обмена информацией между **A** и **B**, который позволяет выработать ключевую битовую цепочку, одинаковую для **A** и **B** с вероятностью, сколько угодно близкой к 1, причем шенноновская информация, которую получит об этой цепочке перехватчик **E**, может быть сделана сколько угодно малой, а скорость формирования ключей $R=k/n$, т.е. отношение количества бит ключа k к длине цепочки n, переданной от **A** и **B** (или от **B** к **A**), оказывается постоянной величиной, но не превосходящей величины $g(P_w) - g(P_m)$, где $g(x)=-x\log_2 p - (1-p)\log_2(1-p)$ – энтропийная функция. (Заметим, что данное свойство сохраняется, если даже **E** является активным перехватчиком, т.е. она может произвольно изменять любую информацию, которой обмениваются легальные пользователи; в этом случае вероятность необнаружения такой подменыможно сделать сколько угодно малой величиной.)

Протоколы, реализующие условия, представленные выше, подробно описаны в работах [27,29]. Суть их состоит в том, что один из легальных пользователей,скажем**A**, посылает по ДСК двоичную цепочку $\boldsymbol{x}$ длины $\boldsymbol{n}$. **B** принимает ее, как $\tilde{\boldsymbol{x}}$с ошибками, далее **A** посылает к **B** цепочку проверочных символов $\boldsymbol{y}$длиной **r** по заранее согласованному между ними коду, а также аутентификаторы **x** и **y**. **B** проверяет, используя алгоритм аутентификации в канале с шумом [27,30], не произведена ли **E** подмена цепочек **x** и **y**. Если **B** обнаруживает подмену, то он прерывает выполнение протокола, в противном случае исправляет ошибки в$\tilde{\boldsymbol{x}}$,и далее оба легальных пользователя **A** и **B** выполняют преобразования $K_A = h(x)$, $K_B = h(x)$, где h – так называемая хеш-функция, получая с большой вероятностью совпадающие цепочки $\mathbf{K_A}$ и $\mathbf{K_B}$.

Если выполнено условие $P_m>P_w$,то возможна реализация такого протокола, который сначала преобразует его к противоположному условию(т.е. $P_m<P_w$), затем выполняется описанный выше протокол.

К сожалению, условие $P_m<P_w$ далеко не всегда выполняется. (Типичным примером, когда это не так, является ситуация распределения ключей между пользователями по электрическому или оптоволоконному кабелю, находящимися внутри защищенного объекта, когда перехватчик **E** вынужден вести

прием по каналам побочного электромагнитного излучения.) Однако, даже если условие $P_m<P_w$ выполнено, то для выбора параметров протокола, обеспечивающих требуемую секретность, нужно знать точные значения P_m и P_w. На практике значение P_w может быть не известно точно, поскольку зависит от чувствительности приемника, коэффициента усиления антенны и других технических факторов канала перехвата. Поэтому рассмотрим далее метод, не зависящий от его характеристик.

2.2.2. Распределение ключей по многолучевому каналу

Другая идея формирования общих ключей для легальных пользователей **A** и **B** основана на использовании случайности многолучевых каналов и случайности движения мобильных пользователей **A** и **B**. В этом случае один из них (скажем **A**) посылает **B** гармонический сигнал по многолучевому каналу. **B**, приняв и запомнив этот сигнал, посылает **A** такой же сигнал. Далее **A** и **B** находят амплитуду или фазу этого сигнала. Поскольку многолучевые каналы подвержены случайным замираниям, эти амплитуда или фазы будут случайными величинами, однако ввиду свойства «обратимости» канала связи они оказываются примерно одинаковыми для **A** и **B**, если весь процесс занимает малое время по сравнению с временем корреляции замираний.

Наконец, **A** и **B** сравнивают принятые имислучайные величины с порогом и принимают решение о формировании бита общего ключа. Поскольку канал перехвата **E** имеет другие случайные параметры, то биты ключа для легальных и нелегальных пользователей оказываются слабо зависимыми. (Для дальнейшего уменьшения этой зависимости используется процедура хеширования.) Аналогичным образом формируются и другие биты общего ключа.

Недостатком данного метода является необходимость присутствия случайности параметров многолучевого канала или случайности движения пользователей **A** и **B**, что может происходить не всегда. Поэтому в [31] был предложен метод, который не предполагает наличия случайностей в каналах связи или шумов в приемнике перехватчика. В этом случае случайность создается специальной антенной (типа ESPAR) со случайно управляемой диаграммой направленности. Все же остальное реализуется таким же образом, как это было описано выше для случая многолучевого канала.

Как было показано теоретически в [32], данный метод обеспечивает высокую секретность распределения ключей (в терминах шенноновской информации, утекающей к перехватчику) при достаточно большой вероятности совпадения ключей **A** и **B** и значительной скорости формирования ключевой цепочки. Он может эффективно использоваться при формировании ключей для пользователей WLAN, находящихся в одном помещении, где многолучевой канал образуется за счет переотражений от стен, пола и потолка.

2.2.3. Распределение ключей на основе принципов квантовой физики («квантовая криптография») [18,33]

В этом случае предполагается, что между**A** и **B** существует пространственная или оптико-волоконная линия связи, по которой можно посылать импульсы поляризованных фотонов с весьма низким средним числом фотонов в каждом импульсе $\mu << 1$.

Процедура выработки ключевой цепочки бит между**A** и **B** называется первичным протоколом. Она показана на рис. 2.3.

1	↺	↕	↻	↔	↕	↕	↔	↔	↻	↺	↕	↻	↺	↺	↕	↻	←
2	+	o	o	+	+	o	o	+	o	+	o	o	o	o	+	+	o
3	↕		↻		↕	↺	↺	↔		↕	↻	↻		↺	↕	↔	↺
4	+		o		+	o	o	+		+	o	o		o	+	+	o
5			V		V			V				V		V	V		
6			↻		↕			↔				↻		↺	↕		
7			I		I			o				I		o	I		

Рис. 2.3.Описание первичного квантового протокола:
1 – **A** посылает **B** последовательность фотонов, случайно модулированных в одной из четырех поляризаций;
2 – **B** производит измерение в случайных базисах поляризаций;
3 – **B** измеряет поляризации фотонов в выбранных им базисах для зарегистрированных фотонов;
4 – **B** сообщает **A** по открытому каналу выбранные им базисы для зарегистрированных фотонов;
5 – **A** сообщает **B** позиции с совпавшими базисами;
6 – **A** и **B** удерживают поляризации в позициях с совпавшими базисами;
7 – **A** и **B** преобразуют значения поляризации для удержанных позиций в биты первичной цепочки:↔=↺= 0,↕=↻= 1

Первичный протокол выполняется также легальными пользователями и в условиях перехвата.

Очевидно, что при отсутствии перехвата со стороны**E** пользователи **A** и **B** после выполнения протокола получают полностью совпадающие ключевые цепочки. Если же **E** выполняет процедуру активного перехвата, то в случае, когда в импульсе содержится два или более фотонов (из общего числа фотонов с совпадавшими базисами), перехватчик получает детерминированный бит ключа с вероятностью 1.

Для импульсов с одиночными фотонами E выполняет оптимальную процедуру (если не рассматривают так называемую неразрушающую стратегию), называемую перехват-пересылкой, которая заключается в измерении поляризации в случайно выбранных канонических базисах (0 или +), и пере-

сылки измеренного в выбранном базисе значения поляризации другому пользователю.

Пример такой стратегии показан на рис. 2.4.

0	1	1	O	0	1	1	0	0
1	↻	↕	↺	↔	↻	↕	↺	↔
2	+	+	+	+	O	O	O	O
3	O	+	O	+	O	+	O	+
4	1 или 0	1	1 или 0	O	1	1 или 0	O	1 или 0
5	1 или 0	1	1 или 0	0	1	1 или 0	0	1 или 0

Рис. 2.4.Описание стратегии «перехват – пересылка»:
0 – значение бита у **A**;
1 – поляризационная модуляция **A**;
2 – выбор базиса перехватчиком **E**;
3 – выбор базиса легальным пользователем**B** (согласован затем с **A**);
4 – значение бита у **E**;
5 – значение бита у **B** (получает ошибочный бит с вероятностью 1/4).

После выполнения первичного протокола формируется цепочка **x** длины n, которая возможно содержит ошибки и относительно которой осуществлен перехват в соответствии с описанной выше стратегией.

Вторичный протокол, выполняемый легальными пользователями с использованием цепочки **x**, полученной после завершения первичного протокола, имеет следующее назначение:

•сформировать для пользователей A и B цепочки Z_A и Z_B длиной $K \le n$ такие, что:

1) $P\{Z_A = Z_B\} \ge p_{зад}$(близко к 1);

2) $I(Z; E) \le I_0$ (близко к 0), где E – вероятностное пространство, моделирующее перехват с учетом наблюдения вторичного протокола;

•отказаться от формирования Z (общий ключ), если условия 1 и 2 не могут быть обеспечены, причем вероятность отказа от формирования ключа при отсутствии перехвата не должна быть больше некоторой величины $P_{ст}$.

Заметим, что квантово-криптографическая система не может ставить задачу по формированию ключа в любых условиях перехвата (канал, например, может быть просто оборван). Она не ставит также в качестве прямой цели обнаружение факта перехвата. (Это иногда может быть достигнуто как побочный результат при решении основной задачи).

Таким образом, необходимость использования квантового канала заключается в том, что только в этом случае можно гарантировать (без нарушения принципа Гейзенберга), что канал перехвата не может быть улучшен.

Вторичный протокол состоит из двух частей:

1) согласование цепочек, полученных **A** и **B** после выполнения первичного протокола (исправления ошибок);

2) обеспечение малости утечки информации о финальном ключе к перехватчику (усиление секретности).

Исправление ошибок может быть выполнено при помощи:

1) итеративного алгоритма сравнения четностей случайно выбираемых подцепочек x и дихотомического поиска ошибок (результаты сравнения четностей передаются по открытому каналу связи);

2) передачи по открытому каналу связи цепочки проверочных символов к цепочке **x**, полученной после выполнения первичного протокола.

В обоих случаях перехватчик, наблюдающий за вторичным протоколом, получает дополнительную информацию об **x**, и это обстоятельство должно быть учтено при формировании финального ключа.

Усиление секретности может быть выполнено на основе:

1) случайного хеширования (рандомизированного кодирования);

2) детерминированного кодирования.

В обоих случаях малость количества информации о финальном ключе, утекающей к перехватчику, обеспечивается только тогда, когда известно, что количество детерминированных бит, полученных перехватчиком об **x** (с учетом контроля протокола исправления ошибок), не превосходит некоторой заданной величины l^*. Поэтому необходимо обеспечить, чтобы вероятность получения им суммарного количества детерминированного количества бит $l > l^*$ была достаточно мала.

Оказывается, параметры данной системы можно выбрать так, что вероятность получения **E** информации о ключе больше некоторой весьма малой величины, не превосходит также некоторой малой величины.

Кроме того, во-первых, при отсутствии вмешательства со стороны**E** вероятность отказа от выполнения протокола со стороны **A** и **B** может быть обеспечена достаточно малой, и, во-вторых, скорость выработки бит ключа (т.е. отношение количества бит распределения ключа к общему количеству бит), которые обеспечивают **A** и **B**, может быть сделана близкой к 1.

Таким образом, система квантово-криптографического распределения ключей теоретически оказывается вполне осуществимой и достаточно эффективной. Вопрос ее практического применения сводится лишь к возможности организации квантового канала связи между легальными пользователями. В литературе имеются уже ссылки на реализацию такой системы с использованием оптико-волоконной линии длиной порядка 50 км со скоростью 1Мбайт/с [34].

2.3. КС с единственным и неединственным дешифрованием сообщений при заданной криптограмме. Расстояние единственности

Рассмотрим следующую постановку задачи: известна криптограмма E и алгоритм дешифрования $g(E,K)$. Тогда можно попытаться, подставляя различные ключи, выявить кандидатов на сообщения, зашифрованные данной криптограммой, причем принимать во внимание только те сообщения $\tilde{M}$, которые имеют смысл (т.е., скажем, смысловые сообщения на каком-либо языке). Очевидно, что при таком методе только одно смысловое сообщение будет истинным, а остальные – ложными. Если число ложных расшифровок окажется очень большим, то невозможно будет определить истинное сообщение даже при переборе всех ключей, т. е. даже когда вычислительный ресурс не ограничен.

Если же ложных расшифровок не будет, то можно гарантировать, что при полном переборе ключей истинное сообщение будет дешифровано верно. (Такой тип атаки называется атакой *тотального* (полного) *перебора* всех ключей.) Интуитивно очевидно, что чем длиннее криптограмма, тем меньше будет число ложных расшифровок.

Определение. *Расстоянием единственности* (РЕ) шифра называется минимальная длина криптограммы ($n_{\text{РЕ}}$), которая приводит при полном переборе ключей к единственному смысловому сообщению, что, очевидно, в точности соответствует истинному ключу. Если $n_{\text{РЕ}} = \infty$, то это означает, что такого единственного ключа не существует.

2.4. Вывод формулы расстояния единственности для произвольного шифра

Теорема Шеннона–Хеллмана. *Пусть шифр удовлетворяет условию единственной шифруемости и дешифруемости и не имеет эквивалентных ключей (т. е. каждое сообщение переводится в определенную криптограмму не более чем одним ключом). Тогда среднее число ложных расшифровок* $\bar{n}_{\text{л.р}}$ *будет иметь следующую нижнюю границу:*

$$\bar{n}_{\text{л.р}} \geq 2^{NH(K)-Dn} - 1, D = \log_2 m - H(M), \qquad (2.6)$$

где m – объем алфавита сообщения и криптограммы; $H(M)$ – энтропия источника сообщения; $H(K)$ – энтропия одного элемента ключа; N – длина ключа; n – длина сообщения.

Доказательство. Рассмотрим для наглядности граф криптосистемы (рис.2.5), где левый столбец соответствует сообщениям, правый – крипто-

граммам, а линия, соединяющая M_i и E_j, означает, что существует ключ k, который переводит сообщение M_i в криптограмму E_j.

Все сообщения могут быть разбиты на две группы: *смысловые* и *бессмысленные*. Из теории информации [4] известно, что для источника сообщений с энтропией $H(M)$ при большой длине сообщения число смысловых (типичных) сообщений $\approx 2^{nH(M)}$.

Общее число возможных сообщений длины n с объемом алфавита m будет равно $m^n = 2^{n\log_2 m}$, и, следовательно, число бессмысленных сообщений будет равно $2^{n\log_2 m} - 2^{nH(M)}$.

Ясно, что при шифровании может выполняться переход к криптограммам только от смысловых сообщений.

При дешифровании, если выбран правильный ключ, переход всегда происходит в смысловое сообщение, а при неправильном ключе переход может происходить в другие смысловые(тогда возникает *ложная расшифровка*) и в бессмысленные (тогда такой ключ отбрасывается как недопустимый) сообщения.

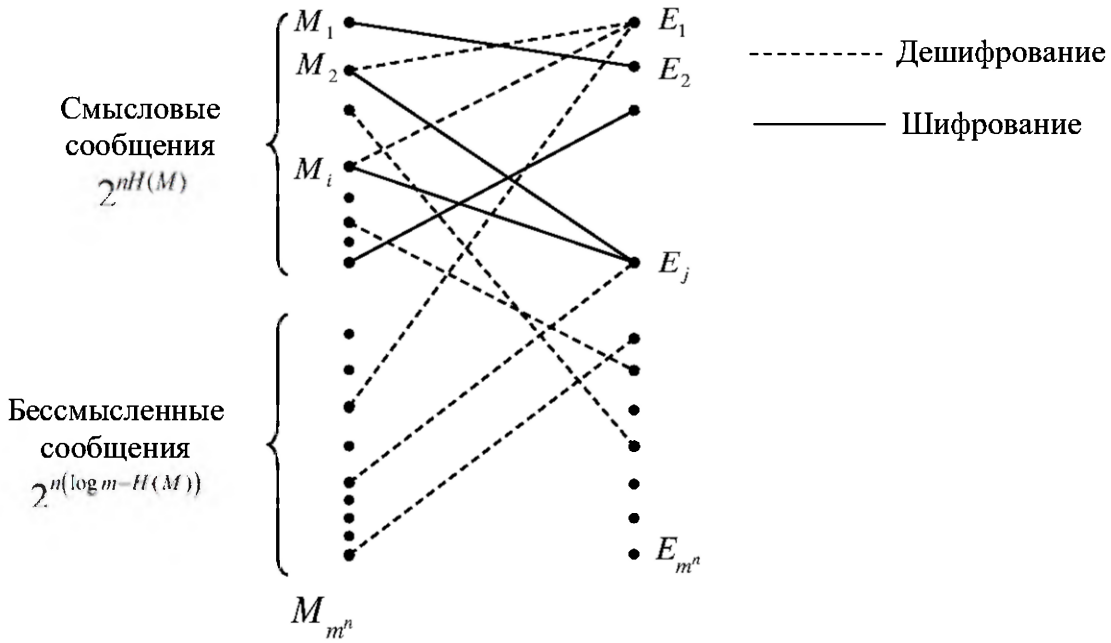


Рис. 2.5. Граф КС

Найдем вероятность появления криптограммы E_j. Из графа видно, что

$$P(E_j) = \sum_{i=1}^{2^{nH(M)}} P(M_i) \cdot P(E_j | M_i), \qquad (2.7)$$

где вероятность каждого типичного сообщения определяется формулой

$$P(M_i) = 2^{-nH(M)}, \quad i = 1,2,\ldots,2^{nH(n)}. \qquad (2.8)$$

Подставив (2.8) в (2.7), получаем

$$P\left(E_j\right)=2^{-nH(M)}\sum_{i=1}^{2^{nH(M)}}P\left(E_j\left|M_i\right.\right)=2^{-nH(M)}\sum_{i=1}^{2^{nH(M)}}P\left(k:M_i\to E_j\right). \qquad (2.9)$$

Поскольку предполагаемые разряды ключа выбираются взаимно независимо, а каждый разряд имеет энтропию $H\left(K\right)$, то при больших длинах ключа N все ключи оказываются равновероятными и имеют вероятность

$$P\left(K\right)=2^{-NH(K)}. \qquad (2.10)$$

Подставив (2.10) в (2.9), получаем

$$P\left(E_j\right)=2^{-nH(M)-NH(K)}\sum_{i=1}^{2^{nH(M)}}S_{ij}, \qquad (2.11)$$

причем $S_{ij}=0$, когда M_i и E_j не соединены линией; $S_{ij}=1$, когда M_i и E_j соединены линией.

Из графа КС мы можем видеть, что

$$\sum_{i=1}^{2^{nH(M)}}S_{ij}=n_j, \qquad (2.12)$$

где n_j – общее число линий, входящих в E_j.

Подставив (2.12) в (2.11), получаем

$$P\left(E_j\right)=2^{-nH(M)-NH(K)}\cdot n_j. \qquad (2.13)$$

Найдем среднее число ложных расшифровок

$$\overline{n}_{\text{л.р}}=\sum_{j=1}^{m^n}P\left(E_j\right)\cdot\left(n_j-1\right). \qquad (2.14)$$

Подставив (2.13) в (2.14), получаем

$$\overline{n}_{\text{л.р}}=\sum_{j=1}^{m^n}2^{-nH(M)-NH(K)}n_j\cdot\left(n_j-1\right)=2^{-nH(M)-NH(K)}\left(\sum_{j=1}^{m^n}n_j^2-\sum_{j=1}^{m^n}n_j\right). \qquad (2.15)$$

Заметим, что сумма $\sum_{j=1}^{m^n}n_j$ равна общему числу линий, идущих от смысловых сообщений ко всем криптограммам. Эта величина определяется формулой

$$\sum_{j=1}^{m^n}n_j=2^{nH(M)+NH(K)}. \qquad (2.16)$$

Подставив (2.16) в (2.15), получаем

$$\overline{n}_{\text{л.р}}=2^{-nH(M)-NH(K)}\sum_{j=1}^{m^n}n_j^2-1. \qquad (2.17)$$

Из курса математики известно, что если сумма некоторых величин постоянна, то сумма квадратов этих величин минимизируется при равен-

стве слагаемых. Поэтому в нашем случае (см. формулу (2.16)) при равенстве слагаемых получаем, что

$$\tilde{n}_j = \frac{2^{nH(M)+NH(K)}}{2^{n\log_2 m}},$$

и тогда

$$\tilde{n}_j^2 = \frac{\left(2^{nH(M)+NH(K)}\right)^2}{\left(2^{n\log_2 m}\right)^2}. \qquad (2.18)$$

Подставляя (2.18) в (2.17) (с неравенством), получаем

$$\bar{n}_{\text{РЕ}} \geq 2^{nH(M)+NH(K)} \cdot 2^{-n\log_2 m} - 1 = 2^{NH(K)-nD} - 1,$$

где $D = \log_2 m - H(M)$, что и требовалось доказать.

2.5. Пояснения к теореме Шеннона–Хеллмана

Если среднее число ложных расшифровок весьма велико, то можно надеяться, что при любом вычислительном ресурсе даже полный перебор ключей не позволит гарантированно определить правильный ключ, и тогда такую криптосистему можно считатьв некотором смысле безусловно стойкой.

Напротив, если число ложных расшифровок равно нулю, то это означает, что в среднем мы их вообще не получим, и если на каком-то ключе при дешифровании получено смысловое сообщение, то этот ключ и является истинным.

Найдем длину криптограммы $n = n_{\text{РЕ}}$, при которой $\bar{n}_{\text{л.р}} = 0$. Так будет, если $2^{NH(K)-nD} = 1, NH(K) = nD$, и тогда $n = \frac{NH(K)}{D}$. Данную длину криптограммы мы и принимаем за расстояние единственности шифра:

$$n_{\text{РЕ}} = \frac{NH(K)}{D} = \frac{NH(K)}{\log_2 m - H(M)}.$$

Частный случай.

Пусть ключ K – двоичный и равновероятный. Тогда $H(K) = 1$ и

$$n_{\text{РЕ}} = \frac{N}{\log_2 m - H(M)}.$$

Если, кроме того, $m = 2$ (двоичный алфавит сообщения и криптограммы), то

$$n_{\text{РЕ}} = \frac{N}{1 - H(M)}.$$

Итак, если производится криптоанализ некоторого шифра и длина криптограммы превосходит расстояние единственности, то можно быть уверенным, что тотальный перебор ключей позволит гарантированно установить истинный ключ шифрования. Таким образом, при условии $n_{\text{РЕ}} < \infty$ невозможно построить криптосистему, которая была бы стойкой, если вычислительный ресурс криптоанализа не ограничен. Поэтому для массового, непривилегированного пользователя необходимо вообще отказаться от требования безусловной стойкости криптосистем.

Определение. КС называется *вычислительно стойкой*, если наилучший возможный алгоритм дешифрования, без знания ключа, требует значительно большего времени или оборудования, чем может быть в распоряжении криптоаналитика.

Однако в настоящее время невозможно указать наилучший алгоритм криптоанализа для большинства современных шифров, и поэтому математически строгое построение вычислительно стойких КС относится к открытым проблемам.

Определение. КС называется *доказуемо стойкой* если взлом сводится к решению предположительно стойкой и строг определенной математической задачи. (Такие КС, как будет показано далее, существуют.)

3. ВЫЧИСЛИТЕЛЬНО СТОЙКИЕ ШИФРЫ

3.1. Способы построения вычислительно стойких блоковых шифров и их криптоанализ

Напомним, чтоблоковым шифром называется такая криптосистема, в которой каждый новый блок открытого сообщения преобразуется в блок криптограммы по одному и тому же правилу, определяемому алгоритмом шифрования и ключом. По такому же принципу выполняется и процедура *дешифрования.*

Напомним, что, согласно принципу Керхгоффа, алгоритмы шифрования и дешифрования полностью известны *криптоаналитику*. Неизвестен лишь ключ, который используется как для шифрования, так и для дешифрования.

Одинаковые блоки сообщений M_i и M_j всегда преобразуются в одинаковые блоки криптограмм E_i и E_j. Если это свойство является нежелательным, то используют другую модификацию того же самого блокового алгоритма шифрования (разд. 3.1.12).

3.1.1. Принцип построения блоковых шифров

Общие принципы построения блоковых шифров были определены К. Э. Шенноном. Они состоят в том, что в алгоритме блокового шифра необходимо использовать:

а) *подстановки* (нелинейные преобразования коротких частей (подблоков блокового шифра);

б) *перестановки* символов в блоках;

в) *итерирование* операций а) и б) (т. е.многократное повторение их с разными ключами).

Рассмотрим эти процедуры в отдельности.

Подстановки. *Это нелинейные преобразования блоков в блоки такой же длины.*

Пример. Блок длины $n = 3 : x_1\ x_1\ x_1 \to y_1 y_2 y_3$ (рис. 3.1).

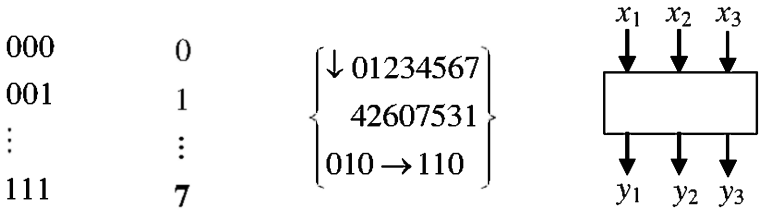


Рис. 3.1. Подстановка в подблоках длины 3

Нелинейные преобразования подстановок часто задаются фиксированной таблицей, которая может зависеть от части ключа. (Очевидно, что для шифров нельзя ограничиться только линейными преобразованиями, так как иначе был бы возможен простой способ нахождения ключа при помощи решения линейных систем уравнений.) Такая таблица должна иметь размерность 2×2^n, что при больших величинах n оказывается нереализуемым. Именно поэтому подстановки применяются к подблокам с длиной значительно меньшей, чем длина n блока шифра, а для устранения зависимости между символами различных подблоков, которая может присутствовать в исходном сообщении, используются преобразования перестановок.

Перестановки. Это преобразование, которое показывает, на какую позицию в выходном блоке должен попасть стоящий на определенной позиции символ входной последовательности.

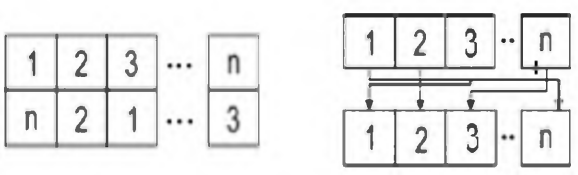


Рис. 3.2. Перестановка элементов блока

3.1.2. Схема Фейстеля

Для упрощения процедур шифрования и дешифрования во многих блоковых шифрах (например, в DES или в ГОСТ) используется *схема Фейстеля*, основанная на следующих принципах:

а) каждый текущий *i*-й блок делится на две равные части – левая L_i и правая R_i, где i – параметр итерации (раунда);

б) способ формирования следующих «половинок» блоков из предыдущих определяется, как показано на рис. 3.3, где k_i – ключ *i*-го раунда.

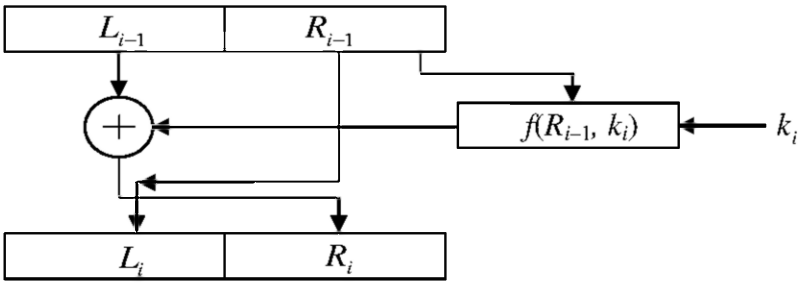


Рис. 3.3. Один раунд схемы Фейстеля

Представим это преобразование в аналитической форме:

$$L_i = R_{i-1}, \quad R_i = L_{i-1} \oplus f\left(R_{i-1}, k_i\right),$$

где $f(\cdot)$ – нелинейная функция от двух аргументов, в которой нелинейность определяется, например, таблицей.

Итерации в схеме Фейстеля *описываются следующими последовательными шагами, преобразующими сообщение* M *в криптограмму*E*:*

$$\begin{aligned} M &= \left(L_0, R_0\right) \\ &\left(L_1, R_1\right) \leftarrow \left\{ \begin{matrix} k_1 \\ \vdots \\ k_r \end{matrix} \right\} \leftarrow k \\ &\left(L_r, R_r\right) \\ E &= \left(L_r, R_r\right) \end{aligned}$$

Раундовые ключи формируются из одного секретного ключа k с помощью детерминированного алгоритма.

Схема Фейстеля обладает двумя преимуществами по сравнению с общей схемой блокового шифра.

Первое преимущество схемы Фейстеля – обратимость процедуры шифрования, хотя функция $f(\cdot)$ в схеме не обязательно является обратимой.

Рассмотрим более подробно выполнение процедуры шифрования и дешифрования по схеме Фейстеля. Один раунд шифрования схемы Фейстеля может быть представлен в следующей символической форме:

$$E_i = \Phi_i(M_i), \Phi_i = T \cdot V_i,$$

где T – перестановка половин блоков, т. е. $T(L,R) = (R,L)$,

$$V_i(L,R) = (L \oplus f(R,k_i), R).$$

Тогда для полного шифра:

$$E = \Phi_r, \Phi_{r-1}, \ldots, \Phi_2, \Phi_1(M),$$

где r – число раундов.

Дешифрование выполняется обратными операциями:

$$M = (\Phi_r \cdot \Phi_{r-1} \cdot \ldots \cdot \Phi_1)^{-1}(E) = \Phi_1^{-1} \cdot \Phi_2^2 \cdot \ldots \cdot \Phi_r^{-1}(E) =$$

$$= V_1 T V_2 T \ldots V_r T(E) = V_1 \Phi_2 \Phi_3 \ldots \Phi_r T(E),$$

поскольку, как легко проверить, $\Phi_i^{-1} = V_i T$.

Видно, что для дешифрования по схеме Фейстеля можно использовать ту же схему, что и для шифрования, но с измененным на обратный порядком следования раундовых ключей.

Второе преимущество схемы Фейстеля. Обе половины блока постоянно меняются местами и поэтому, несмотря на кажущуюся несимметричность, обе половины исходного блока шифруются с одинаковой стойкостью.

Существует множество вариантов шифров, построенных на основе схемы Фейстеля, которые отличаются видом нелинейной функции, числом раундов, размерами блока шифрования и способом выработки раундовых ключей по исходному ключу. Однако современные блоковые шифры предпочитают строить несколько на других принципах, чем принципы, на которых построена схема Фейстеля, сохраняя при этом основные принципы Шеннона. Такой класс шифров получил название *подстановочно-перестановочных* шифров (ППШ).

3.1.3. Подстановочно-перестановочные шифры [5]

Далее будем рассматривать упрощенный учебный пример ППШ, а в дальнейшем покажем возможность построения реально стойкого шифра, основанного на тех же принципах, но с другими параметрами.

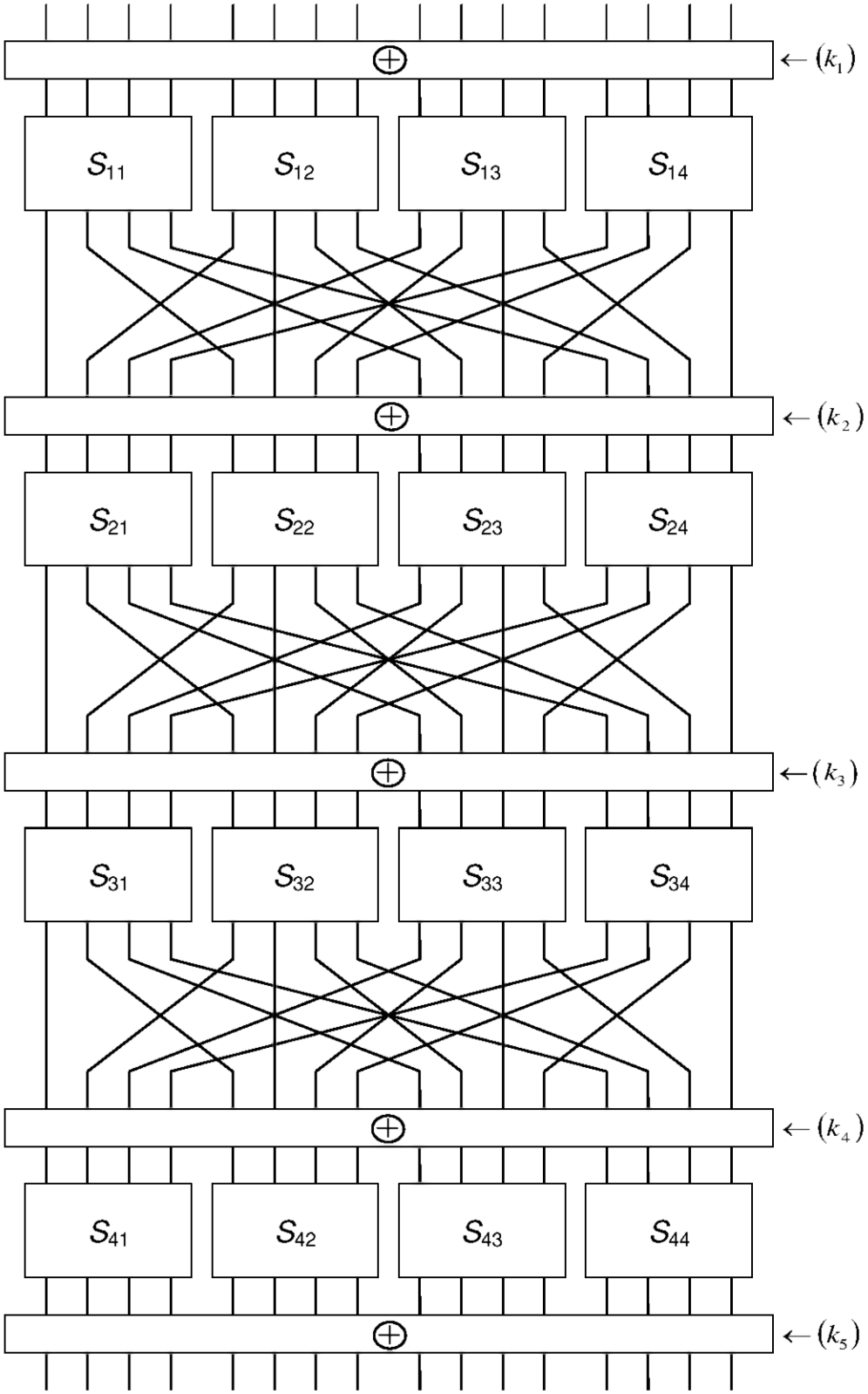


Рис. 3.4. Учебный ППШ

Из схемы, представленной на рис. 3.4, видно, что такой шифр имеет 4 итерации, причем каждая из них включает в себя сложение по модулю 2 с раундовым ключом, нелинейное преобразование, выполняемое в четырех *S*-блоках,и перестановку символов, определяемую направлениями линий. В последнем раунде перестановка не используется, но используется 5-й раундовый ключ. Длина блока учебного ППШ равна 16, а общее число бит раундовых ключей – 80.

Все *S*-блоки выполняют одинаковое табличное преобразование, не зависящее от ключа и задаваемое табл. 3.1 (в 16-ричной системе).

Таблица 3.1

Преобразования в *S*-блоках

Вход	0	1	2	3	4	5	6	7	8	9	10(A)	11(B)	12(C)	13(D)	14(E)	15(F)
Выход	E	4	D	1	2	F	B	8	3	A	6	C	5	9	0	7

Преобразования перестановок на каждом раунде также одинаковы, не зависят от ключа и задаются табл. 3.2.

Таблица 3.2

Преобразования перестановок

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

3.1.4. Основные методы криптоанализа блоковых шифров

Рассмотрим (с той или иной степенью подробности) следующие методы криптоанализа блоковых шифров:

1) полный (*тотальный*) перебор ключей;
2) *линейный* криптоанализ;
3) *дифференциальный (разностный)* криптоанализ;
4) криптоанализ по методу *максимального правдоподобия*;
5) криптоанализ на основе *решения алгебраических (булевых) уравнений*;
6) криптоанализ аппаратно реализованного шифра на основе *внесения ошибок* в процедуру шифрования.
7) криптоанализ аппаратно реализованного шифра на основе *вычисления потребляемой мощности.*

3.1.5. Полный перебор ключей

При данном методе криптоанализа, криптоаналитик перебирает все допустимые ключи, пытаясь дешифровать криптограмму достаточно большой длины n, равной или превосходящей расстояние единственности

$n_{\text{РЕ}}$, т. е. при условии, что $n \geq n_{\text{РЕ}}$. Тогда первый ключ, который даст смысловой результат при дешифровании, и принимается за истинный. Однако возможности данного метода ограничены временем перебора всех или (хотя бы в среднем) половины ключей. Если N – длина двоичного ключа, то при $N > 128$ такой криптоанализ оказывается невозможным ни при каких вычислительных ресурсах. Действительно, полное количество ключей тогда оказывается равным $2^{128} \cong 3{,}4 \cdot 10^{38}$. (В рассматриваемом учебном ППШ имеется 80 бит ключа, и тогда $2^{80} \approx 1{,}2 \cdot 10^{24}$, что также нереализуемо на ПК.)

Однако известна более эффективная методика поиска ключей, которая имеет название метода криптоанализа на основе использования таблиц поиска (так называемых **радужных таблиц**) [42].

3.1.6. Линейный криптоанализ

Линейный криптоанализ блокового шифра был предложен Мацуи в 1994 г. Основная идея его состоит в использовании (существующих для любых нелинейныхпреобразований) *скрытых линейных уравнений*, связывающих некоторые биты входа и выхода.

При выполнении данной атаки предполагается использовать много открытых текстов и соответствующих им криптограмм (т. е. это атака второго типа).

Общее уравнение линейных связей имеет вид

$$X_{i_1} \oplus X_{i_2} \oplus \ldots \oplus X_{i_n}, \oplus Y_{j_t} \oplus Y_{j_2} \oplus \ldots \oplus Y_{j_l} = 0, \qquad (3.1)$$

где X_i – i-й бит входа; Y_j – j-й бит выхода.

Такие уравнения будут существовать не всегда, а лишь для некоторой части входных сообщений. Поэтому мы можем говорить только о некоторой *вероятности* их выполнения. Для определения номеров $i_1, \ldots, i_n$; $j_1, \ldots, j_l$ влинейном уравнении (3.1) необходимо выбрать такие их них, которые обеспечивают наибольшую вероятность выполнения (3.1) по всем возможным входам.

Для решения этой задачи для полного шифра будет использован следующий подход, состоящий из двух этапов:

1) сначала исследуются отдельныеS-блоки (так как только они имеют нелинейность) и находится их наилучшая аппроксимация линейными уравнениями;

2) затем производится объединение всех S-блоков для установления полной связи входа всего шифра с его выходом.

Рассмотрим решение задачи на первом этапе. Проведем анализ возможностей по линейной аппроксимации S-блока для учебного шифра, представленного ранее таблицей с четырьмя входами и четырьмя выходами для каждого S-блока (табл. 3.1 и рис. 3.5).

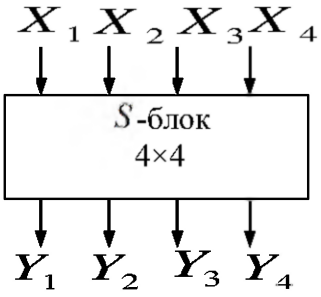


Рис. 3.5. Структура *S*-блока ППШ

Рассмотрим в качестве первой попытки аппроксимации следующее уравнение:

$$X_2 \oplus X_3 \oplus Y_1 \oplus Y_3 \oplus Y_4 = 0,$$

что эквивалентно равенству $X_2 \oplus X_3 = Y_1 \oplus Y_3 \oplus Y_4$.

Перебирая 16 возможных входов $X_1 X_2 X_3 X_4$, мы можем наблюдать частоту выполнения данного равенства, используя табл. 3.1, определяющую преобразование входов *S*-блоков в их выходы. Результаты такого расчета приведены в табл. 3.3.

Таблица 3.3

Примеры расчета линейных аппроксимаций *S*-BOX-а

X_1	X_2	X_3	X_4	Y_1	Y_2	Y_3	Y_4	$X_2 \oplus X_3$	$Y_1 \oplus Y_3 \oplus Y_4$	$X_1 \oplus X_4$	Y_2	$X_3 \oplus X_4$	$Y_1 \oplus Y_4$
0	0	0	0	1	1	1	0	0	0	0	1	0	1
0	0	0	1	0	1	0	0	0	0	1	1	1	0
0	0	1	0	1	1	0	1	1	0	0	1	1	0
0	0	1	1	0	0	0	1	1	1	1	0	0	1
0	1	0	0	0	0	1	0	1	1	0	0	0	0
0	1	0	1	1	1	1	1	1	1	1	1	1	0
0	1	1	0	1	0	1	1	0	1	0	0	1	0
0	1	1	1	1	0	0	0	0	1	1	0	0	1
1	0	0	0	0	0	1	1	0	0	1	0	0	1
1	0	0	1	1	0	1	0	0	0	0	0	1	1
1	0	1	0	0	1	1	0	1	1	1	1	1	0
1	0	1	1	1	1	0	0	1	1	0	1	0	1
1	1	0	0	0	1	0	1	1	1	1	1	0	1
1	1	0	1	1	0	0	1	1	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1	0	1	0
1	1	1	1	0	1	1	1	0	0	0	1	0	1

Из табл. 3.3 видим, что в 12 случаях из 16 равенство выполняется. Тогда можно определить так называемый «перекос вероятности», т. е. отклонение ее от $\frac{1}{2}$: $\frac{12}{16} - \frac{1}{2} = \frac{1}{4}$.

Для другого уравнения $X_1 \oplus X_4 = Y_2$ аналогично получаем перекос вероятности, равный 0, а для уравнения $X_3 \oplus X_4 = Y_1 \oplus Y_4$ перекос будет следующий: $\frac{2}{16} - \frac{1}{2} = -\frac{3}{8}$. В последнем случае скорее уместно рассматривать аппроксимацию вида $X_3 \oplus X_4 \oplus Y_1 \oplus Y_4 = 1$, но это не имеет принципиального значения, поскольку в дальнейшеммы будем вычислять перекос вероятности по абсолютной величине.

В табл. 3.4 представлены результаты проверки всех видов комбинаций (записанных в 16-ричной форме), связывающих входы и выходы *S*-блока, при различных входных последовательностях. На пересечении дается количество выполненных уравнений минус 8. Тогда для получения перекоса надо разделить на 16 значения, приведенные в табл. 3.4.

Пример 3.1.1. $x_3 \oplus x_4 \to 0011 \to \{3\}$; $Y_1 \oplus Y_4 \to 1001 \to \{9\}$.

На пересечении получаем –6, и тогда величина перекоса вероятности будет следующая: $-\frac{6}{16} = -\frac{3}{8}$.

Рассмотрим решение задачи на втором этапе. (*Анализ линейной аппроксимации для полного шифра.)* Для этого необходимо выбрать линейные комбинации, соответствующие максимальным (по абсолютной величине) значениям их перекосов вероятностей и проследить, как эти комбинации сочетаются друг с другом для заданной структуры шифра.

Таблица 3.4

Полный набор линейных аппроксимаций для *S*-блока

		Выходные суммы															
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**A**	**B**	**C**	**D**	**E**	**F**
Входные суммы	**0**	+8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	–2	–2	0	0	–2	+6	+2	+2	0	0	+2	+2	0	0
	2	0	0	–2	–2	0	0	–2	–2	0	0	+2	+2	0	0	–6	+2
	3	0	0	0	0	0	0	0	0	+2	–6	–2	–2	+2	+2	–2	–2
	4	0	+2	0	–2	–2	–4	–2	0	0	–2	0	+2	+2	–4	+2	0
	5	0	–2	–2	0	–2	0	+4	+2	–2	0	-4	+2	0	–2	–2	0
	6	0	+2	-2	+4	+2	0	0	+2	0	–2	+2	+4	–2	0	0	–2
	7	0	–2	0	+2	+2	-4	+2	0	–2	0	+2	0	+4	+2	0	+2
	8	0	0	0	0	0	0	0	0	–2	+2	+2	–2	+2	–2	–2	–6
	9	0	0	–2	–2	0	0	–2	–2	–4	0	–2	+2	0	+4	+2	–2
	A	0	+4	–2	+2	–4	0	+2	–2	+2	+2	0	0	+2	+2	0	0
	B	0	+4	0	-4	+4	0	+4	0	0	0	0	0	0	0	0	0
	C	0	–2	+4	–2	–2	0	+2	0	+2	0	+2	+4	0	+2	0	–2
	D	0	+2	+2	0	–2	+4	0	+2	–4	–2	+2	0	+2	0	0	+2
	E	0	+2	+2	0	–2	–4	0	+2	–2	0	0	–2	–4	+2	–2	0
	F	0	–2	–4	–2	–2	0	+2	0	0	–2	+4	–2	–2	0	+2	0

Пример 3.1.2. Выберем следующую аппроксимацию для *S*-блоков:

$S_{12}: X_1 \oplus X_3 \oplus X_4 = Y_2$ с вероятностью $\left(\frac{3}{4}\right)$; $S_{22}: X_2 = Y_2 \oplus Y_4$ с вероятностью $\left(\frac{1}{4}\right)$;

$S_{32}: X_2 = Y_2 \oplus Y_4$ с вероятностью $\left(\frac{1}{4}\right)$; $S_{34}: X_2 = Y_2 \oplus Y_4$ с вероятностью $\left(\frac{1}{4}\right)$.

Далее будем использовать следующие обозначения:

U_i – входная последовательность для *S*-блока на *i*-м раунде;

V_i – выходная последовательность для *S*-блока на *i*-м раунде;

U_{ij} – *j*-й бит блока U_i;

V_{ij} – *j*-й бит блока V_i;

k_i – блок ключей на входе*i*-го раунда;

k_{ij} – *j*-й бит блока k_i.

Из приведенной на рис.3.6 схемы видим, что $U_1 = P \oplus k_1$, где P – открытое сообщение.

Используя аппроксимацию для S_{12}, получаем уравнение для выхода 1-го раунда

$$V_{16} = U_{15} + U_{17} + U_{18} = \left(P_5 \oplus k_{1,5}\right) \oplus \left(P_7 \oplus k_{1,7}\right) \oplus \left(P_8 \oplus k_{1,8}\right) \text{ с вероятностью } \left(3/4\right). \quad (3.2)$$

Для 2-го раунда получаем

$$V_{2,6} \oplus V_{2,8} = U_{2,6} \text{ с вероятностью } \left(1/4\right). \quad (3.3)$$

Поскольку $V_{2,6} \oplus V_{1,6} = U_{2,6}$, мы будем иметь

$$V_{2,6} \oplus V_{2,8} = V_{1,6} \oplus k_{2,6} \text{ с вероятностью } \left(1/4\right). \quad (3.4)$$

Подставляя (3.2) в (3.4), получим

$$V_{26} \oplus V_{2,8} \oplus P_5 \oplus P_7 \oplus P_8 \oplus k_{1,5} \oplus k_{17} \oplus k_{18} \oplus k_{2,6} = 0. \quad (3.5)$$

Взаимосвязи аппроксимируемых блоков показаны на рис. 3.6.

Для расчета вероятности выполнения равенства (3.5) используем лемму [5].

Лемма. *Пусть* $x_1, x_2, \ldots, x_n$ *– случайные независимые двоичные переменные, причем заданы вероятности* $P\left(x_i = 0\right) = \frac{1}{2} + \varepsilon_i$. *Тогда*

$$P\left(\left(x_1 \oplus x_2 \oplus \ldots \oplus x_n\right) = 0\right) = \frac{1}{2} + 2^{n-1} \prod_{i=1}^{n} \varepsilon_i .$$

Используя эту лемму, получаем вероятность выполнения равенства (3.5):

$$P\left((3.5)\right) = \frac{1}{2} + 2 \cdot \left(\frac{3}{4} - \frac{1}{2}\right) \cdot \left(\frac{1}{4} - \frac{1}{2}\right) = \frac{3}{8}.$$

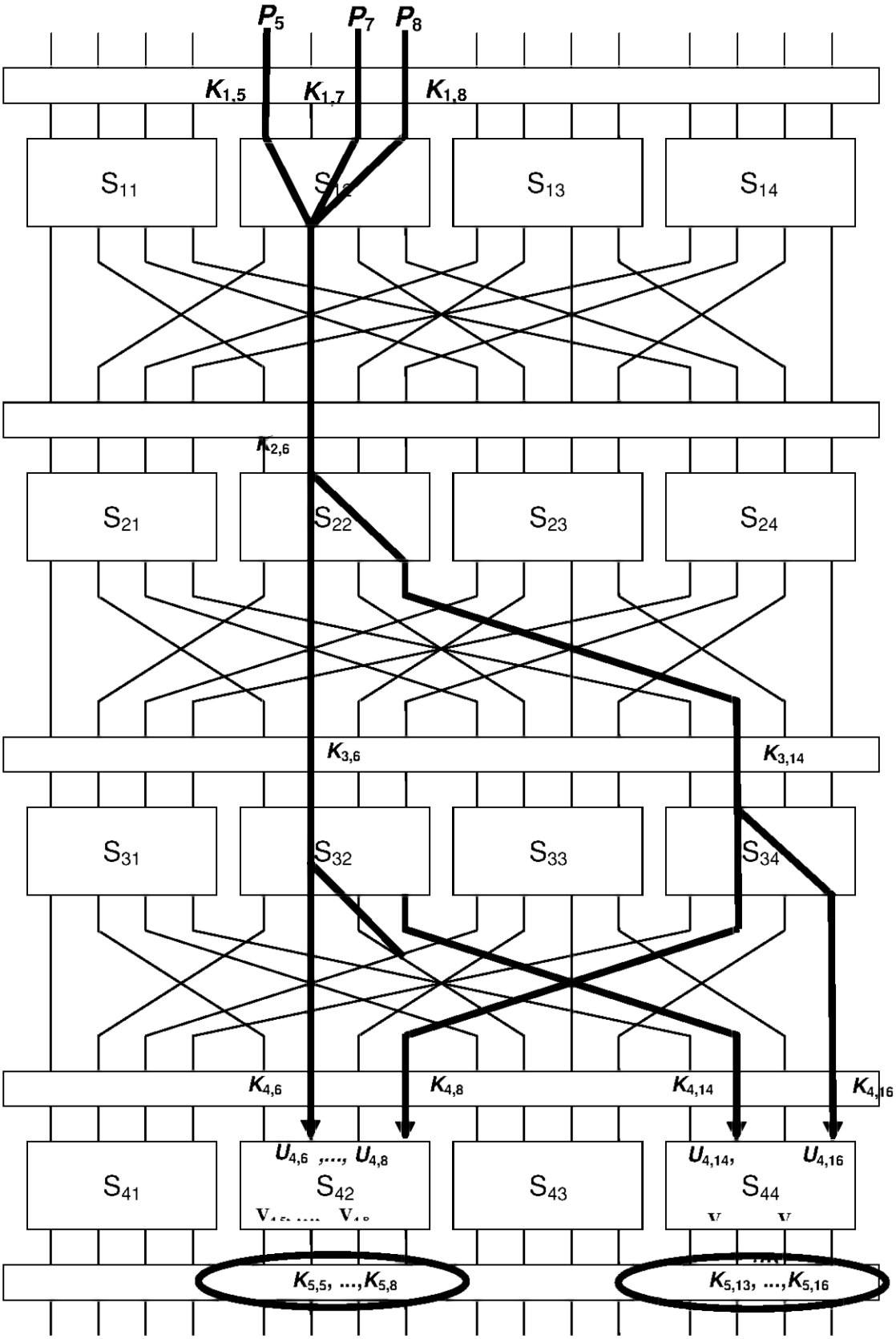


Рис. 3.6. Линейная аппроксимация ППШ

Аналогичным образом получим соотношение для 3-го раунда, используя линейную аппроксимацию S_{32}:

$$V_{3,6} \oplus V_{3,8} = U_{3,6} \text{ с вероятностью (1/4);} \quad (3.6)$$

$$V_{3,14} \oplus V_{3,16} = U_{3,14} \text{ с вероятностью (1/4).} \quad (3.7)$$

Из схемы шифра видно, что:

$$U_{3,6} = V_{2,6} \oplus k_{3,6}; \quad (3.8)$$

$$U_{3,14} = V_{2,8} \oplus k_{3,14}. \quad (3.9)$$

Объединяя (3.6)–(3.9), получаем:

$$V_{3,6} \oplus V_{3,8} \oplus V_{3,14} \oplus V_{3,16} \oplus V_{2,6} \oplus k_{3,6} \oplus V_{2,8} \oplus k_{3,14} = 0 \text{ с вероятностью } \frac{1}{2} + 2 \cdot \left(\frac{1}{4} - \frac{1}{2}\right)^2 = \frac{5}{8} \quad (3.10)$$

Подставим в (3.10) $V_{2,6} \oplus V_{2,8}$ из (3.5):

$$V_{3,6} \oplus V_{3,8} \oplus V_{3,14} \oplus V_{3,16} \oplus P_5 \oplus P_7 \oplus P_8 \oplus k_{1,5} \oplus k_{1,7} \oplus k_{1,8} \oplus k_{2,6} \oplus k_{3,6} \oplus k_{3,14} = 0. \quad (3.11)$$

Из схемы шифра, представленной на рис. 3.6, для входов 4-го раунда получаем:

$$U_{4,6} = V_{3,6} \oplus k_{4,6};\ U_{4,8} = V_{3,14} \oplus k_{4,8};\ U_{4,14} = V_{3,8} \oplus k_{4,14};\ U_{4,16} = V_{3,16} \oplus k_{4,16}. \quad (3.12)$$

Подставляя уравнения (3.12) в (3.11), окончательно получаем:

$$U_{4,6} \oplus U_{4,8} \oplus U_{4,14} \oplus U_{4,16} \oplus P_5 \oplus P_7 \oplus P_8 \oplus \sum k = \begin{cases} 1, \text{если } \sum k = 1; \\ 0, \text{если } \sum k = 0, \end{cases} \quad (3.13)$$

где $\sum k = k_{1,5} \oplus k_{1,7} \oplus k_{1,8} \oplus k_{2,6} \oplus k_{3,6} \oplus k_{3,14} \oplus k_{4,6} \oplus k_{4,8} \oplus k_{4,14} \oplus k_{4,16}$.

Использование леммы дает вероятность выполнения (3.13):

$$\frac{1}{2} + 2^3 \cdot \left(\frac{3}{4} - \frac{1}{2}\right) \cdot \left(\frac{1}{4} - \frac{1}{2}\right)^3 = \frac{15}{32} \text{ с перекосом } \left(-\frac{1}{32}\right).$$

Из равенства (3.13) видно, что оно принимает значения 0 или 1, в зависимости от значения суммы ключей $\sum k$. Этот факт позволяет определить ключи последнего 5-го раунда методом перебора, не обращая пока внимания на другие ключи. Делается это следующим образом:

1) задается множество входных и выходных сообщений P_i, E_i, где $i = 1, 2, \ldots, T$;

2) по известной криптограмме E_i вычисляются V_4 для различных ключей. В действительности нас интересуют только компоненты этого вектора: $V_{4,5}, \ldots, V_{4,8} \ldots; V_{4,13} \ldots V_{4,16}$;

3) по известной структуре S-блоков вычисляются входы 4-го раунда: $U_{4,5}, \ldots, U_{4,8}; U_{4,13}, \ldots, U_{4,16}$;

4) рассчитывается количество выполнений равенства (3.13) для различных входных сообщений и каждого из всех возможных ключей $k_{5,5}\ldots k_{5,8};k_{5,13},k_{5,16}$;

5) в качестве истинного ключа выбирается тот ключ, который дает наибольшее отклонение количества выполнений равенства (3.13), – от $T/2$. Если ключ в п.5 определен верно, то аналогичным образом определяется набор оставшихся ключей $k_{5,1}\ldots k_{5,4};k_{5,9},k_{5,12}$;

6) определив с большой вероятностью все ключи 5-го раунда, мы сможем рассчитать в точности вход 4-го раунда по известной криптограмме. Далее повторяя алгоритм, описанный в пп. 1–5, где вместо криптограмм E_i используются входы 4-го раунда U_4, вычисляются ключи 4-го раунда и так далее – вплоть до нахождения ключей 1-го раунда.

Важно отметить, что в данном случае сложность криптоанализа оценивают прежде всего числом T необходимых для его выполнения пар – открытых текстов (P) и криптограмм (E).

Пример. Для T = 10000 в табл. 3.5 приведены результаты криптоанализа по описанному алгоритму для истинного ключа (2,4) и других ключей, представленных в 16-ричной системе. Видно, что для данного эксперимента наиболее вероятный ключ действительно соответствует тому, который реально использовался.

Таблица 3.5

Экспериментальные результаты линейного криптоанализа для ППШ

Извлекаемый подключ [$K_{5,5},\ldots, K_{5,8}; K_{5,13},\ldots, K_{5,16}$]	|Перекос |	Извлекаемый подключ [$K_{5,5},\ldots, K_{5,8};K_{5,13},\ldots, K_{5,16}$]	| Перекос |
1,C	0,0031	2,A	0,0044
2,D	0,0078	2,B	0,0186
1,E	0,0071	2,C	0,0094
1,F	0,0170	2,D	0,0053
2,0	0,0025	2,E	0,0062
2,1	0,0220	2,F	0,0133
2,2	0,0211	3,0	0,0027
2,3	0,0064	3,1	0,0050
2,4	**0,0336**	3,2	0,0075
2,5	0,0106	3,3	0,0162
2,6	0,0096	3,4	0,0218
2,7	0,0074	3,5	0,0052
2,8	0,0224	3,6	0,0056
2,9	0,0054	3,7	0,0048

В теории линейного криптоанализа доказывается [5], что если ε – это перекос выполнения линейного равенства для всего шифра, типа (3.13), то

необходимое число известных пар (сообщение, криптограмма) будет примерно $T = \frac{1}{\varepsilon^2}$.

Для того чтобы противостоять атаке линейного криптоанализа, необходимо *S*-блоки строить с высокой степенью нелинейности, когда вероятности их аппроксимаций линейными равенствами минимальны, а также задать такую структуру блокового шифра, которая приводила бы к максимальному числу *активных S*-блоков на каждом из раундов.

Пути решения этих задач рассматриваются в разд. 3.1.10, посвященном разработке стойких блоковых шифров.

3.1.7. Дифференциальный (разностный) криптоанализ блоковых шифров

Дифференциальный криптоанализ (ДК), предложенный в 1991 г. Бихамом и Шамиром, использует аномально повышенные вероятности появления некоторых разностей криптограмм для определенных разностей между открытыми сообщениями.

Обозначим через $\overline{X} = X_1 X_2 ... X_n$ вход и через $\overline{Y} = Y_1 Y_2 ... Y_n$ выход некоторого блокового шифра. Зафиксируем последовательности $\overline{X}', \overline{X}''$ и соответствующие им $\overline{Y}', \overline{Y}''$. Определим входные и выходные разности следующим образом: $\Delta\overline{X} = \overline{X}'' \oplus \overline{X}'$; $\Delta\overline{Y} = \overline{Y}'' \oplus \overline{Y}'$.

В случае идеального шифра выходные разности были бы равновероятными, т. е. выполнялось бы соотношение $\Pr(\Delta\overline{Y}) = \frac{1}{2^n}$ для всех входных разностей $\Delta\overline{X}$.

Дифференциальный криптоанализ (ДК) основан на гипотезе о существовании определенных выходных разностей, которые имеют повышенные или пониженные вероятности (т. е. отличающиеся от $\frac{1}{2^n}$ в большую или в меньшую сторону). ДК, также как и ЛК, распадается на два этапа:

1) анализ разностей для каждого S-блока;

2) анализ разностей для всего шифра.

Рассмотрим оба эти этапа по очереди.

Анализ разностей для S-блока. В табл. 3.6 показаны некоторые разности ΔX и соответствующие им разности ΔY для различных X, где $\Delta Y = Y + Y'$, $Y = S(X), Y' = S(X')$, $X' = X \oplus \Delta X$, $S(\cdot)$–преобразование, выполняемое *S*-блоком.

Таблица 3.6

Примеры пар разностей для *S*-блока

X	Y	ΔY		
		$\Delta X = 1011$	$\Delta X = 1000$	$\Delta X = 0100$
0000	1110	0010	1101	1100
0001	0100	0010	1110	1011
0010	1101	0111	0101	0110
0011	0001	0010	1011	1001
0100	0010	0101	0111	1100
0101	1111	1111	0110	1011
0110	1011	0010	1011	0110
0111	1000	1101	1111	1001
1000	0011	0010	1101	0110
1001	1010	0111	1110	0011
1010	0110	0010	0101	0110
1011	1100	0010	1011	1011
1100	0101	1101	0111	0110
1101	1001	0010	0110	0011
1110	0000	1111	1011	0110
1111	0111	0101	1111	1011

Из табл. 3.6 видно, что $\Delta Y = 0010$ появляется в 8 случаях из 16, когда $\Delta X = 1011$, т. е. вероятность ее появления равна $\frac{1}{2}$, вместо $\frac{1}{16}$, которая была бы для чисто случайного выбора. Если $\Delta X = 0100$, то $\Delta Y = 0010$ вообще не появляется. Таким образом, различные виды разностей ΔY появляются с различной частотой.

Распределение значений выходных разностей для *S*-блока и различных входных разностей представлено в табл. 3.7, где все эти разности записаны в 16-ричной системе. Видно, что наибольшее число раз (8) появляется выходная разность $\Delta Y = 2$ при входной разности $\Delta X = B$.

Таблица 3.7

Распределение всевозможных разностей *S*-блока учебного шифра

Входная разность	Выходная разность															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	2	0	0	0	2	0	2	4	0	4	2	0	0
2	0	0	0	2	0	6	2	2	0	2	0	0	0	0	2	0
3	0	0	2	0	2	0	0	0	0	4	2	0	2	0	0	4
4	0	0	0	2	0	0	6	0	0	2	0	4	2	0	0	0
5	0	4	0	0	0	2	2	0	0	0	4	0	2	0	0	2
6	0	0	0	4	0	4	0	0	0	0	0	0	2	2	2	2
7	0	0	2	2	2	0	2	0	0	2	2	0	0	0	0	4
8	0	0	0	0	0	0	2	2	0	0	0	4	0	4	2	2
9	0	2	0	0	2	0	0	4	2	0	2	2	2	0	0	0
A	0	2	2	0	0	0	0	0	6	0	0	2	0	0	4	0
B	0	0	8	0	0	2	0	2	0	0	0	0	0	2	0	2
C	0	2	0	0	2	2	2	0	0	0	0	2	0	6	0	0
D	0	4	0	0	0	0	0	4	2	0	2	0	2	0	2	0
E	0	0	2	4	2	0	0	0	6	0	0	0	0	0	2	0
F	0	2	0	0	6	0	0	0	0	4	0	2	0	0	2	0

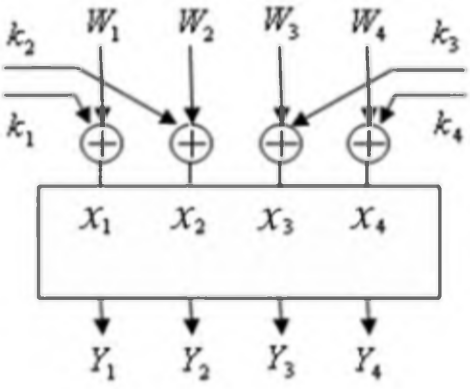


Рис. 3.7. Нахождение разности при наличии раундовых ключей

Прежде чем перейти ко второму этапу, отметим важное свойство этого метода криптоанализа: *раундовые ключи не влияют на входные разности*. Действительно, из схемы, приведенной на рис. 3.7, следует справедливость следующего соотношения:

$$\Delta\overline{W_i} = (\overline{X}' \oplus \overline{k_i}) \oplus (\overline{X}'' \oplus \overline{k_i}) = (\overline{X}' \oplus \overline{X}'') = \Delta X ,$$

поскольку раундовый ключ k_i сохраняется постоянным, и тогда $\overline{k_i} \oplus \overline{k_i} = 0$.

Расчет разностных характеристик для полного шифра. Будем использовать следующие разностные пары для *S*-блоков, через которые проходят жирные линии на рис.3.8:

$$S_{(1,2)} \quad \Delta X = B \quad \rightarrow \quad \Delta Y = 2 \quad \text{с вероятностью} \quad \left(\frac{8}{16}\right);$$

$$S_{(2,3)} \quad \Delta X = 4 \quad \rightarrow \quad \Delta Y = 6 \quad \text{с вероятностью} \quad \left(\frac{6}{16}\right);$$

$$S_{(3,2)} \quad \Delta X = 2 \quad \rightarrow \quad \Delta Y = 5 \quad \text{с вероятностью} \quad \left(\frac{6}{16}\right);$$

$$S_{(3,3)} \quad \Delta X = 2 \quad \rightarrow \quad \Delta Y = 5 \quad \text{с вероятностью} \quad \left(\frac{6}{16}\right).$$

Для всех остальных блоков входные разности будут нулевыми, что на выходе даст также нулевые разности. Используя те же обозначения для входов и выходов раундов, что и при линейном криптоанализе, но с индексом Δ (разность), мы получаем следующие выражения для соответствующих разностей:

$$\Delta P = \Delta U_1 = [0000\,1011\,0000\,0000];$$

$$\Delta V_1 = [0000\,0010\,0000\,0000], (\Delta Y = 2) \quad \text{с вероятностью} \quad \left(\left(\frac{8}{16}\right) = \frac{1}{2}\right).$$

После перестановок в первом слое, получаем

$$\Delta U_2 = [0000\,0000\,0100\,0000], (\Delta Y = 2) \quad \text{с вероятностью} \quad \left(\frac{1}{2}\right).$$

Далее используем ранее выбранную выходную пару для блока $S_{2,3}$

$$\Delta V_2 = [0000\,0000\,0110\,0000].$$

После перестановок получаем для следующего слоя

$$\Delta U_3 = [0000\,0010\,0010\,0000] \quad \text{с вероятностью} \quad \left(\frac{1}{2} \cdot \frac{6}{16} = \frac{3}{16}\right).$$

Переходя к блокам $S_{3,2}$, $S_{3,3}$, получаем

$$\Delta V_3 = [0000\,0101\,0101\,0000] \quad \text{с вероятностью} \quad \left(\frac{6}{16}\right)^2.$$

Для слоя ΔU_4 получаем

$$\Delta U_4 = [0000\,0110\,0000\,0110] \quad \text{с вероятностью} \quad \left(\frac{8}{16} \cdot \frac{6}{16} \cdot \left(\frac{6}{16}\right)^2 = \frac{27}{1024}\right).$$

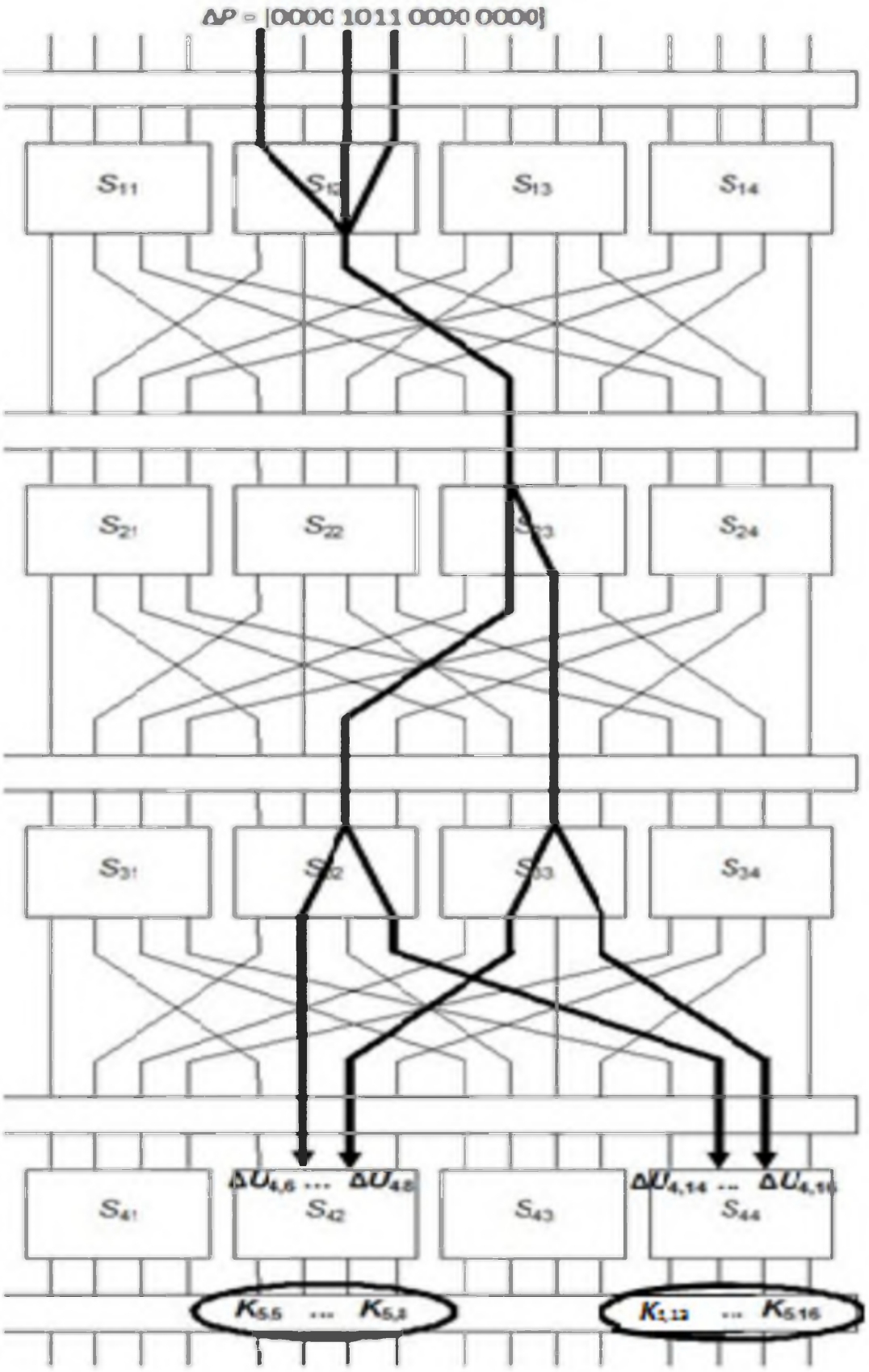


Рис. 3.8. Расчет разностей для полного шифра

Теперь можно провести криптоанализ полного шифра в целях определения некоторой части ключа 5-го раунда. Для этого перебирается множество всевозможных пар входных сообщений с определенными входными разностями. (В нашем примере эта разность равна $[0000\,1011\,0000\,0000]$.) Далее для каждого входного сообщения находятся криптограммы и по ним выходы блоков S_{42}, S_{44} при переборе всех возможных ключей $k_{5,5},...,k_{5,8}$; $k_{5,13},...,k_{5,16}$. Выходы этих блоков пересчитываются на их входы по табл. 3.1, и находятся разности ΔU_4 для каждого из ключей. Затем рассчитывается количество совпадений этих разностей с найденной ранее *высоковероятной* разностью ΔU_4 (в нашем примере $\Delta U_4 = [0000\,0110\,0000\,0110]$), и в качестве истинного ключа выбирается тот ключ, который дает наибольшее количество таких совпадений.

Результаты эксперимента для 5000 пар сообщений/криптограмм и для такого же истинного ключа (2,4), как и в эксперименте с ЛК, заимствованные из [5], представлены в табл. 3.8.

Таблица 3.8

Экспериментальные результаты
дифференциального криптоанализа учебного ППШ

Извлекаемый подключ $[K_{5,5},\ldots, K_{5,8}; K_{5,13},\ldots, K_{5,16}]$	Вероятность	Извлекаемый подключ $[K_{5,5},\ldots, K_{5,8};K_{5,13},\ldots, K_{5,16}]$	Вероятность
1,C	0,0000	2,A	0,0032
1,D	0,0000	2,B	0,0022
1,E	0,0000	2,C	0,0000
1,F	0,0000	2,D	0,0000
2,0	0,0000	2,E	0,0000
2,1	0,0136	2,F	0,0000
2,2	0,0068	3,0	0,0004
2,3	0,0068	3,1	0,0000
2,4	**0,0244**	3,2	0,0004
2,5	0,0000	3,3	0,0004
2,6	0,0068	3,4	0,0000
2,7	0,0068	3,5	0,0004
2,8	0,0030	3,6	0,0000
2,9	0,0024	3,7	0,0008

Из табл. 3.8 видно, что истинный ключ в данном эксперименте определяется верно.

При помощи аналогичного алгоритма находится другая пара подключей для 5-го раунда,и затем, используя точное знание ключей 5-го раунда, можно переходить к вскрытию ключей 4-го и всех остальных раундов.

Сложность данного криптоанализа определяется также, как и линейного, количеством необходимых для него пар открытых сообщений и криптограмм.

Приближенное количество таких требуемых пар определяется соотношением [5]

$$N_D \approx \frac{C}{P_D}, \tag{3.14}$$

где C – некоторая постоянная; P_D – вероятность появления наиболее вероятной разности для всего шифра.

В свою очередь, эту вероятность можно оценить как

$$P_D \le \prod_{i=1}^{\gamma} \beta_i, \tag{3.15}$$

где β_i – вероятность появления наиболее вероятной разности для i-го раунда; γ – полное число раундов блокового шифра.

Из (3.14) и (3.15) видно, что для разработки шифров, стойких к дифференциальному криптоанализу, необходимо в них использовать S-блоки, которые имеют *минимальные вероятности появления высоковероятностных разностей*, а также шифры, обладающие *разветвленной структурой*. Для построения таких шифров (с *доказуемой стойкостью* к дифференциальному и линейному криптоанализу) нам потребуется дополнительный математический аппарат, описывающий свойства булевых функций и конечных полей.

3.1.8. Булевы функции в криптографических преобразованиях

Определение 3.1.1. Отображение $GF(2)^n \to GF(2)^m$ называется *векторной булевой функцией* (БФ), если его компоненты являются обычными (скалярными) булевыми функциями, где $GF(2)^{n(m)}$ – множество всех двоичных последовательностей длиной $n(m)$. *Скалярная булева функция* – это отображение $GF(2)^n$ в 0 или 1. Векторную БФ можно записать так:

$$\bar{Y} = f(\bar{x}) = (y_1,..., y_m) = (f_1(\bar{x}), f_2(\bar{x}),..., f_m(\bar{x})),$$

где $\bar{x} = (x_1, x_2,..., x_n)$.

Определение 3.1.2. Последовательность значений скалярной булевой функции $f(\bar{x})$, состоящая из элементов 0 и 1 и имеющая вид $f(\beta_0)f(\beta_1)...f(\beta_{2^n-1})$, где β_i пробегает все значения из $GF(2)^n$, расположенные в *лексикографическом* порядке, называется таблицей истинности булевой функции $f(\bar{x})$.

Пример. $n = 2$.

$\bar{x}$	00	01	10	11
$f(x)$	0	1	1	0

Определение 3.1.3. Сопряженной булевой функцией $f^*(\bar{x})$ к $f(\bar{x})$ называется функция, принимающая значения ±1 и имеющая вид

$$f^*(\bar{x}) = (-1)^{f(x)}. \tag{3.16}$$

Пример. $n = 2$.

$\bar{x}$	00	01	10	11
$f(\bar{x})$	0	1	1	0
$f^*(\bar{x})$	+1	–1	–1	+1

Очевидно, что любая БФ может быть представлена таблицей истинности. Однако возникает вопрос – можно ли любую БФ описать в «более сжатом виде», т.е. в виде формулы, содержащей операции над входными двоичными переменными? Ответ на этот вопрос оказывается положительным и даже не однозначным. Так, можно доказать, что любая БФ, отличная от тождественного нуля, может быть представлена в виде совокупности операций «И», «ИЛИ», «НЕТ», и такое представление называется *нормальной дизъюнктивной формой.*

В теории булевых функций доказывается также, что любая БФ может быть представлена в виде так называемой алгебраической нормальной формы –АНФ (иначе говоря, в виде *полиномов Жегалкина*):

$$f(\bar{x}) = \gamma_0 + \sum_{i=0} \gamma_i x_i \oplus \sum_{i_1 \le} \sum_{i_2} \gamma_{i_1 i_2} x_{i_1} x_{i_2} \oplus \ldots \oplus \gamma_{1,2,\ldots,n} x_1 x_2 \ldots x_n, \qquad (3.17)$$

где все коэффициенты $\gamma_{i_1 i_2 \ldots i_s}$ равны 0 или 1.

Существует способ получения алгебраической нормальной формы по известной таблице истинности. Он реализуется при помощи следующего алгоритма [6]:

1) составляется таблица истинности из 2^n значений булевой функции f в лексикографическом порядке. Обозначим ее S_f и представим в виде столбца:

$$S_f = \begin{pmatrix} f(\beta_0) \\ \ldots \\ f(\beta_{2^n-1}) \end{pmatrix};$$

2) выписывается столбец $\bar{\gamma}_f$ из 2^n неизвестных пока коэффициентов АНФ в лексикографическом порядке: $\bar{\gamma}_f = \begin{pmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \end{pmatrix}$;

3) строится матрица A_n следующим рекуррентным образом:

$$A_1 = [1];$$

$$A_n = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \otimes A_{n-1} = \begin{bmatrix} A_{n-1} & 0 \\ A_{n-1} & A_{n-1} \end{bmatrix},$$

где $\otimes$ – знак *тензорного* произведения матриц;

4) находятся коэффициенты АНФ следующим образом:

$$\bar{\gamma}_f = A_n \bar{S}_f. \qquad (3.18)$$

Пример. Пусть БФ для $n = 2$ задана следующей таблицей истинности:

00	01	10	11
0	1	0	0

Тогда $\overline{S}_f = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$.

Подставляя этот столбец в (3.18), получаем

$$\begin{pmatrix} \gamma_0 \\ \gamma_1 \\ \gamma_2 \\ \gamma_3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}.$$

Тогда в соответствии с представлением (3.17) получим

$$f(x_1, x_2) = x_1 \oplus x_1 x_2.$$

Сложность вычисления АНФ по таблице истинности является, как видно, полиномиальной по размеру матрицы A_n, но экспоненциальной по порядку булевой функции *n*.

Определение 3.1.4. Булева функция $f(\overline{x})$ называется *аффинно-линейной*, если она представима в следующем виде:

$$f(\overline{x}) = C \oplus \sum_{i=1}^{n} \alpha_i x_i = C \oplus (\overline{\alpha}, \overline{x}), \tag{3.19}$$

где $(\overline{\alpha}, \overline{x})$ называется скалярным произведением в поле $GF(2)$; $C \in (0,1)$, $\alpha_i \in (0,1)$.

В частном случае, если $C = 0$, то функция называется просто *линейной*. В криптографических приложениях востребованы нелинейные функции, причем чем больше нелинейность этих функций, тем лучше. Поэтому необходимо ввести определенный показатель степени нелинейности БФ. Это потребует определения новых понятий.

Определение 3.1.5. Преобразованием Уолша–Адамара (ПУА) от БФ $f(\overline{x})$ называется следующая функция векторного параметра $\overline{\alpha} \in GF(2)^n$:

$$U_{\overline{\alpha}}(f) = \sum_{\overline{x} \in GF(2)^n} f(\overline{x}) \cdot (-1)^{(\overline{\alpha}, \overline{x})}. \tag{3.20}$$

Заметим, что значения ПУА принадлежат области вещественных чисел (сложение под знаком суммы выполняется в обычной арифметике, а не по модулю 2, хотя скалярное произведение в показателе производится по (3.19) с использованием сложения по модулю 2).

Для сопряженной булевой функции ПУА имеет следующий вид:

$$U_{\bar{\alpha}}(f^*) = \sum_{\bar{x} \in GF(2)^n} (-1)^{(f(\bar{x}) \oplus (\bar{\alpha}, \bar{x}))}. \quad (3.21)$$

ПУА имеет обратные преобразования (восстанавливающие исходную БФ или сопряженную к ней), которые задаются следующим образом:

$$f(\bar{x}) = \frac{1}{2^n} \sum_{\alpha \in GF(2)^n} U_{\bar{\alpha}}(f) \cdot (-1)^{(\bar{\alpha}, \bar{x})}; \quad (3.22)$$

$$f^*(\bar{x}) = \frac{1}{2^n} \sum_{\alpha \in GF(2)^n} U_{\bar{\alpha}}(f^*) \cdot (-1)^{(\bar{\alpha}, \bar{x})}. \quad (3.23)$$

В теории булевых функций [6] доказывается следующее равенство:

$$\underbrace{U_{\bar{\alpha}}(f)}_{\substack{\text{столбец} \\ 2^n}} = \underbrace{H_n}_{\substack{\text{матрица} \\ 2^n \times 2^n}} \cdot \underbrace{\left| S(f^*) \right|}_{\substack{\text{столбец} \\ 2^m}}, \quad (3.24)$$

где таблица истинности $S(f^*)$ для сопряженной булевой функции выглядит следующим образом:

$$\begin{pmatrix} (-1)^{f(\beta_0)} \\ (-1)^{f(\beta_1)} \\ \vdots \\ (-1)^{f(\beta_{2^n-1})} \end{pmatrix}.$$

В (3.24) матрица H_n – это *матрица Адамара*, которая задается рекурентно следующим образом:

$$H_0 = [1]; \; H_n = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \otimes H_{n-1} = \begin{bmatrix} H_{n-1} & H_{n-1} \\ H_{n-1} & -H_{n-1} \end{bmatrix},$$

где $\otimes$ – символ тензорного умножения матриц.

Определение 3.1.6. *Расстоянием Хэмминга* (обозначается $D_H(f,g)$) между двумя скалярными булевыми функциями одной размерности называется число позиций, в которых отличаются их таблицы истинности, т.е. $D_H(f,g) = W_H(S(f) \oplus S(g))$, где $W_H(...)$ – вес Хэмминга двоичной последовательности, т.е. число единиц, которое она содержит.

Определение 3.1.7. *Коэффициентом корреляции* (обозначается $R(f,g)$) между двумя булевыми функциями называется величина:

$$R(f,g) = \frac{1}{2^n} \sum_{\bar{x} \in GF(2)^n} f^*(\bar{x}) \cdot g^*(\bar{x}). \quad (3.25)$$

Расстояние Хэмминга и коэффициент корреляции связаны между собой следующим образом:

$$R(f,g) = 1 - \frac{1}{2^{n-1}} D_H(f,g). \quad (3.26)$$

Определение 3.1.8. *Нелинейностью* БФ называется следующая величина:

$$N(f) = \min_{f_{aff}} D_H(f, f_{aff}), \tag{3.27}$$

где f_{aff} – аффинно-линейные функции той же размерности.

Из этого определения видно, что нелинейность БФ тем больше, чем больше расстояние Хэмминга от нее до ближайшей аффинно-линейной функции.

Утверждение 3.1.1. Нелинейность БФ рассчитывается так [6]:

$$N(f) = 2^{n-1} - \frac{1}{2} \max_{\bar{\alpha} \in GF(2)^n} \left| U_{\bar{\alpha}}(f^*) \right|. \tag{3.28}$$

Если булева функция является векторной, т. е. это отображение $GF(2)^n \to GF(2)^m$, то *нелинейностью векторной* БФ будем называть

$$\tilde{N}(f) = \min_{\substack{\bar{b} \in GF(2)^m \\ \bar{b} \neq 0}} N(\bar{b}, \bar{f}), \tag{3.29}$$

где $(\bar{b}, \bar{f}) = b_1 f_1 \oplus b_2 f_2 \oplus \ldots \oplus b_m f_m$.

Определение 3.1.9. *Автокорреляцией (АКФ)* булевой функции $f(\bar{x})$ называется следующая функция, зависящая от векторного параметра $\bar{\alpha}$:

$$R_{\bar{\alpha}}(f) = \frac{1}{2^n} \sum_{\bar{x} \in GF(2)^n} f^*(x) \cdot f^*(\bar{x} \oplus \bar{\alpha}) = \left(S^*\left(f\right), S^*\left(f\left(\bar{x} \oplus \bar{\alpha}\right)\right)\right). \tag{3.30}$$

Предположим, что на вход булевой функции поступает случайная векторная величина $\bar{x}$, координаты которой равновероятны и взаимно независимы. Тогда для любого двоичного вектора $\bar{\alpha}$ будет справедливо следующее соотношение [6]:

$$\Pr\left(f^*\left(\bar{x}\right)\right) \neq f^*(\bar{x} \oplus \bar{\alpha}) = \frac{1}{2} - \frac{1}{2} R_{\bar{\alpha}}\left(f\right). \tag{3.31}$$

Из (3.31) видно, что если $\bar{\alpha}$ определяет некоторую ошибку в исходном векторе $\bar{x}$, то вероятность совпадения сопряженных БФ, вычисленных для исходного и искаженного векторов, будет тем более близка к $\frac{1}{2}$, чем меньше величина $R_{\bar{\alpha}}$ (значение автокорреляционной функции). Поэтому можно ожидать, что чем ближе к нулю значения АКФ БФ для всех векторов $\bar{\alpha} \neq 0$, тем больше будет стойкость шифра, использующего *S*-блоки с данными БФ, к *разностному* криптоанализу.

Можно показать, что устойчивость шифра к *линейному* криптоанализу будет тем больше, чем больше нелинейность векторных БФ, описывающих *S*-блоки, входящие в состав этого шифра.

Для оценки устойчивости к другим статистическим методам криптоанализа оказываются полезными следующие дополнительные свойства БФ.

Определение 3.1.10. Булева функция $f\left(\bar{x}\right)$ называется *сбалансированной*, если ее таблица истинности содержит равное число 0 и 1 (это чис-

ло равно 2^{n-1} для БФ $f(\bar{x})$ порядка n). В терминах ПУА сбалансированная булева функция определяется следующим соотношением: $U_{\bar{0}}(f) = 2^{n-1}$ или $U_{\bar{0}}(f^*) = 0$.

Определение 3.1.11. Булева функция $f(\bar{x})$ порядка n называется *корреляционно нечувствительной (корреляционно иммунной) степени* $l < n$, если $U_{\bar{\alpha}}(f^*) = 0$, для всех $\bar{\alpha}$ веса Хэмминга от 1 до l.

Наличие корреляционно нечувствительной степени l означает, что если на вход булевой функции подаются равновероятные и независимые двоичные символы, то значение БФ $Y = f(\bar{x})$ будет статически независимым от любого набора, состоящего не более чем из l компонент входного аргумента $\bar{x}$.

Определенные выше свойства БФ широко использовались в исследованиях по разработке *S*-блоков, обеспечивающих стойкость шифра к различным методам криптоанализа. Применительно к линейному и разностному криптоанализу были получены результаты, позволяющие достаточно строго оптимизировать построение *S*-блоков и структуры всего шифра. Однако для того чтобы их изложить, нам потребуется развить некоторый дополнительный математический аппарат («Теорию конечных полей»), который будет также эффективно использован и в других разделах курса («Потоковые шифры» и «Аутентификация сообщений»), а также во второй части книги («Основы криптографии с открытым ключом»).

3.1.9. Элементы теории конечных полей

Определение 3.1.12. *Конечным полем* $GF(q)$, *или полем Галуа порядка* q ,называют конечное произвольное множество элементов с заданными между ними операциями сложения, умножения и деления. Эти операции обладают следующими свойствами:

1. $\forall a,b \in GF(q), a+b \in GF(q)$;
2. $\forall a,b \in GF(q), a \cdot b \in GF(q)$;
3. $a+b=b+a$;
4. $a \cdot b = b \cdot a$;
5. $(a+b)+c = a+(b+c) = a+b+c$;
6. $a \cdot (b+c) = a \cdot b + a \cdot c$;
7. $\exists$ элемент «O» $\in GF(q), a+O=a, \forall a \in GF(q)$;
8. $\exists$ элемент «$-a$» $\in GF(q)$, такой, что $a+(-a)=O, \forall a \in GF(q)$;
9. $\exists$ элемент «e» $\in GF(q), a \cdot e = a, \forall a \in GF(q)$;

10. $\forall a \in GF(q),\ a \neq 0,\ \exists a^{-1} : a \cdot a^{-1} = e.$

Определение 3.1.13. *Характеристикой* «*p*» конечного поля $GF(q)$ называют наименьшее натуральное число, такое что: $e \cdot p = \underbrace{e+e+e+...+e}_{p} = 0.$

Утверждение 3.2. Характеристика любого конечного поля всегда будет простым числом.

Доказательство. Предположим, что это не так, т. е.

$$\underbrace{e+e+e+...+e}_{p},$$

где $p = p' \cdot p''$. Тогда

$$\underbrace{e+e+e+...+e}_{p'} + \underbrace{e+e+e+...+e}_{p'} + ... + \underbrace{e+e+e+...+e}_{p'} = 0.$$

Обозначим через *a*частные суммы, содержащие p'слагаемых, что дает равенство

$$\underbrace{a+a+a+...+a}_{p''} = 0. \qquad (3.32)$$

Так как $a \neq 0$, существует элемент a^{-1}. Умножив обе части (3.32) на a^{-1}, получаем

$$\underbrace{e+e+e+...+e}_{p''} = 0,$$

что противоречит определению характеристики поля, поскольку $p'' < p$.

Легко видеть, что в любом конечном поле $GF(q)$ характеристики «*p*», существует *простое подполе* $GF(p)$, включенное в $GF(q)$.

Действительно, рассмотрим множество $0, 1, 1+1, ..., \underbrace{1+1+1+...+1}_{p-1}$.

Множество образует поле, удовлетворяющее всем определенным выше свойствам $GF(p)$. Выберем в качестве e обычную единицу (1), в качестве элементов поля – числа 0, 1, 2, 3, …, $(p-1)$, а все действия будем рассматривать как математические операции по $\bmod p$, т.е. вычисление остатка отделения на *p*результатов обычных арифметических операций.

Хотя это лишь один пример простого поля, но можно показать, что любые простые поля изоморфны такому полю, которое обозначают обычно Z_p. (Определение изоморфизма будет дано далее).

Рассмотрим некоторые примеры простых полей.

Пример 3.1.6. Пусть $p = 2$, тогда $GF(2) = \{0,1\}$, причем все операции выполняются по $\bmod 2$:

$$(0+0=0 \quad 0+1=1 \quad 1+0=1 \quad 1+1=0)$$

Пример 3.1.7. Пусть $p=5$, тогда $GF(5)=\{0,1,2,3,4\}$, причем все операции выполняются по $\mathrm{mod}5$ (табл. 3.9а, 3.9б).

Таблица 3.9а
Сложение в поле $GF(5)$

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

Таблица 3.9б
Умножение в поле $GF(5)$

×	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

Вернемся теперь к общему случаю конечного поля $GF(q)$, где q– любое (не обязательно простое) число. Тогда в поле существует конечный базис, т. е. минимальное число линейно независимых элементов $\alpha_1,\alpha_2,...,\alpha_n$, $\alpha_i \in GF(q)$. Линейные комбинации элементов базиса с коэффициентами из простого поля $GF(p)$, очевидно, образуют все элементы поля $GF(q)$, т.е.:

$$\forall \alpha \in GF(q) \quad : \quad \alpha = \sum_{i=1}^{n} \alpha_i C_i; \quad C_i \in GF(p).$$

Отсюда следует, что общее число элементов поля $GF(q)$ будет равно $q = p^n$, где p – характеристика поля (простое число), n – натуральное число.

Таким образом, фактически доказана теорема, что *всякое конечное поле может содержать число элементов, равное только целой неотрицательной степени простого числа.*

Так, например, число элементов поля может быть:

$$q = 2,3,4,5,7,8,9,11,13,...$$

и не может быть: $q = 6,10,12,15,...$

Конструкция конечного поля. Рассмотрим множество всех последовательностей длины n, каждая позиция которой принимает любое значение из множества $(0,...,p-1)$. Тогда общее число последовательностей будет, очевидно, равно $q = p^n$.

Пример3.1.8. Поле $GF(2^3)$. Тогда $n=3$ и получаем следующие элементы поля $GF(2^3)$ в виде 8 двоичных последовательностей:

000, 001, 010-α, 011, 100, 101-β, 110, 111-γ.

Определим сложение и вычитание на этом множестве последовательностей как покомпонентное сложение по модулю p, т. е.:

$$\alpha + \beta = 010 \oplus 101 = 111 = \gamma.$$

Ноль в таком поле – это нулевая последовательность000.

Однако для задания умножения и деления на множестве этих последовательностей нам потребуется дополнительное определение.

Определение 3.1.14. Многочлен $f(x)$ с коэффициентами из поля $GF(p)$ называется *неприводимым* в поле $GF(p)$, если он не может быть представлен как произведение двух и более многочленов с коэффициентами из этого же поля.

Пример 3.1.9. Многочлен $x^2+1=(x+1)\cdot(x+1)=x^2+\underbrace{1x+1x}_{0}+1$ – приводимый многочлен; x^3+x+1 – неприводимый.(Заметим, что при перемножении многочленов действия над их коэффициентами должны выполняться по правилам действий в поле $GF(p)$.)

Далее будем отождествлять последовательности длины n с многочленами, коэффициенты которых соответствуют номерам позиций (значениям разрядов последовательностей):

$$00...0\to 0;\quad 00...1\to 1;\quad 00...10\to x...11...1\to x^{n-1}+x^{n-2}+...+1.$$

Так, данные о соответствии последовательностей и многочленов для поля $GF(2^3)$ приведены в табл. 3.10.

Таблица 3.10

Соответствие последовательностей и многочленов в поле $GF(2^3)$

№ п/п	Последовательность	Многочлен
0	000	0
1	001	1
2	010	x
3	011	x+1
4	100	x^2
5	101	x^2+1
6	110	x^2+x
7	111	x^2+x+1

Определим операции умножения между элементами поля $GF(p^n)$ как перемножение соответствующих этим элементам многочленов с приведением результатов по модулю любого неприводимого многочлена $f(x)$ степени n. *Приведенный по модулю* $f(x)$ многочлен равен остатку от деления этого многочлена на $f(x)$.

Пример 3.10. Рассмотрим поле $GF(2^3)$, неприводимый многочлен $f(x)=x^3+x+1$ и перемножим элементы поля:

$$\alpha=110\Rightarrow x^2+x;$$

$$\beta = 111 \Rightarrow x^2 + x + 1;$$

$$\alpha \cdot \beta = x^4 + \cancel{x^3} + \cancel{x^2} + \cancel{x^2} + \cancel{x^2} + x$$

$$= \begin{array}{l|l} x^4 + x & x^3 + x \cdot \\ x^4 + x^2 + x & \overline{\qquad\qquad} \\ \hline x^2 & x \end{array}$$

$$\alpha\beta = x^2 = 100$$

Легко проверить, что такое определение сложения, вычитания и умножения между элементами поля соответствует всем аксиомам, которые предъявляются к конечным полям. Можно выполнить и деление на ненулевой элемент поля, что эквивалентно умножению на обратный элемент поля. Найдем далее метод вычисления такого обратного элемента.

Для многочленов можно доказать утверждение, аналогичное тому, которое доказывается в теории чисел [7], а именно: пусть $a(x), b(x)$ – многочлены над полем $GF(p)$. Тогда их *общий наибольший делитель* (gcd) может быть представлен в следующем виде:

$$\gcd\big(a(x), b(x)\big) = u(x) \cdot a(x) + v(x) \cdot b(x), \qquad (3.33)$$

где $u(x), v(x)$ – некоторые многочлены над полем $GF(p)$. (Напомним, что $\gcd\big(a(x), b(x)\big)$ – это, по определению, многочлен наибольшей степени, который делит $a(x)$ и $b(x)$.)

Из этого утверждения вытекает следствие, аналогичное тому, которое доказывается далее в теории чисел. Пусть $a(x), b(x) = f(x)$ над $GF(p)$ такие, что: $\gcd\big(a(x), f(x)\big) = 1$. Тогда из (3.33) следует, что существует единственный многочлен

$$u(x) : a(x) \cdot u(x) = 1, \ \mathrm{mod}\big(f(x)\big). \qquad (3.34)$$

Из (3.34) видно, что $u(x) = a(x)^{-1}$, т. е. $u(x)$ – это обратный элемент к элементу поля $a(x)$. Поскольку в конечном поле многочлен $f(x)$ всегда выбирается неприводимым, тогда $\gcd\big(a(x), f(x)\big) = 1$ и все ненулевые элементы поля имеют обратные элементы, которые находятся по расширенному алгоритму Евклида [7]. Деление будет выполняться следующим образом:

$$\frac{a(x)}{b(x)} = a(x) \cdot b(x)^{-1}.$$

Построенные выше конструкции полей называют *полями классов вычетов* по модулю неприводимого многочлена.

В теории конечных полей доказывается важное утверждение, что любые конечные поля $GF(p^n)$ изоморфны полю классов вычетов по модулю любого неприводимого над $GF(p)$ многочлена степени n.

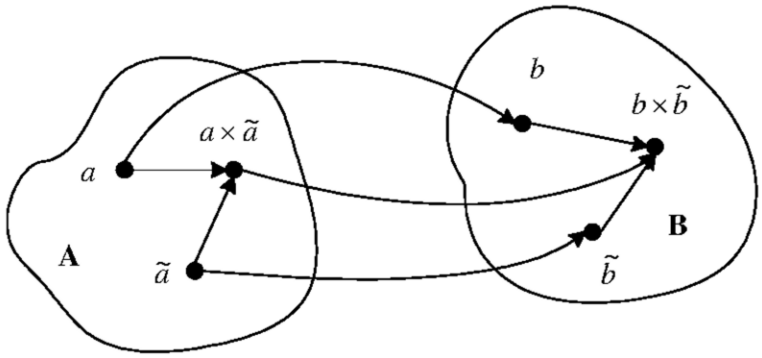


Рис. 3.9. Изоморфизм конечных полей

Изоморфизм – это соответствие между элементами двух множеств A и B, которое сохраняет основные операции (рис.3.9). Изоморфные поля можно считать совпадающими с точностью до порядка обозначения их элементов.

Основные свойства конечных полей

1. Пусть $a,b\in GF(p^n)$. Тогда $(a+b)^p=a^p+b^p$.

Доказательство:

$$(a+b)^p=a^p+C_p^1a^{p-1}b+C_p^2a^{p-2}b^2+...+b^p,$$

где $C_p^i=\dfrac{p!}{i!(p-i)!}$, $i=1,2,...,n$. Из этого выражения видно, что все *биномиальные коэффициенты* содержат в качестве множителя число p, откуда следует, что они равны нулю по модулю p.

Определение 3.1.15. *Порядком e* элемента конечного поля $\alpha\in GF(q)$ называется наименьшее целое положительное число, такое что $\alpha^e=1$. Очевидно, что порядок любого элемента конечного поля всегда будет конечен.

2. В поле $GF(q)$ порядок*е* любого элемента α делит $q-1$.

Доказательство. Докажем сначала, что всегда $\alpha^{q-1}=1$. Для доказательства рассмотрим произведение всех ненулевых элементов поля $\alpha_1\cdot\alpha_2\cdot...\cdot\alpha_{q-1}$. Умножим каждый из этих элементов на α. Получаем $(\alpha_1\cdot\alpha)\cdot(\alpha_2\cdot\alpha)\cdot...\cdot(\alpha_{q-1}\cdot\alpha)$. Очевидно, что данная операция будет лишь перестановкой строки $\alpha_1\cdot\alpha_2\cdot...\cdot\alpha_{q-1}$, т. е.: $\alpha_1\cdot\alpha_2\cdot...\cdot\alpha_{q-1}=(\alpha_1\cdot\alpha)\cdot(\alpha_2\cdot\alpha)\cdot...\cdot(\alpha_{q-1}\cdot\alpha)$. Отсюда получаем, что $\alpha_1\cdot\alpha_2\cdot...\cdot\alpha_{q-1}=\alpha^{q-1}(\alpha_1\cdot\alpha\cdot...\alpha_{q-1})\Rightarrow\alpha^{q-1}=1$, а это и требовалось доказать.

Пусть e – порядок элемента α, т. е. $\alpha^e=1$. Предположим, что $e\nmid q-1$ (т. е. e не делит $q-1$). Таким образом, $q-1=be+r$, где $r<e$.

Рассмотрим теперь $\alpha^r=\alpha^{q-1-be}=\alpha^{q-1}\cdot(\alpha^{be})^{-1}=1\cdot1$, а это приводит к противоречию, так как r не может быть порядком, поскольку $r<e$, а e –

это, по определению, такое минимальное число, что $\alpha^e = 1$. Следовательно, утверждение неверно и $e \mid q-1$.

Можно доказать следующие свойства, относящиеся к порядкам элементов поля (пп. 3–5).

3. Пусть e_1 – порядок элемента поля α_1, e_2 – порядок элемента поля α_2, $\gcd(e_1, e_2) = 1$. Тогда порядок произведения $\alpha_1 \cdot \alpha_2$ равен $e_1 \cdot e_2$.

4. Если $Od(a) = e$, то $Od(\alpha^k) = \frac{e}{d}$, где $d = \gcd(e,k)$; $Od(...)$ – обозначение порядка элемента поля.

Определение 3.1.16. Элемент α, принадлежащий конечному полю $GF(q)$, называется *примитивным*, если его порядок равен $q-1$.

Ясно, что степени примитивного элемента $\alpha^1, \alpha^2, \alpha^3, ..., \alpha^{q-1} = 1$ образуют все элементы поля, за исключением нуля.

5. Каждое конечное поле $GF(q)$ содержит хотя бы один примитивный элемент [8, теорема 6.2].

Существование примитивных элементов конечного поля позволяет доказать многие его свойства и существенно упростить вычисления в конечных полях. Так, если поле задано в виде степеней примитивного элемента, то таблица умножения становится вообще не нужной.

Действительно, пусть $\beta, \gamma \in GF(q)$, где $\beta = \alpha^i$, $\gamma = \alpha^j$. Тогда $\beta \cdot \gamma = \alpha^{i+j}$ ($i + j$ надо приводить по модулю $q-1$).

Пример 3.1.11. Рассмотрим поле $GF(2^3)$. Пусть $f(x) = x^3 + x + 1$ – неприводимый многочлен, образующий это поле. Тогда если α – корень уравнения $\alpha^3 + \alpha + 1$, который является примитивным элементом, то все элементы поля можно представить в виде степеней α (табл. 3.11).

Таблица 3.11

Представление элементов поля $GF(2^3)$ степенями примитивного элемента при $f(x) = x^3 + x + 1$

№ п/п	Представление в виде последовательности	Представление степенью примитивного элемента
0	000	0
1	001	$\alpha^0 = 1$
2	010	$\alpha^1 = \alpha$
3	100	α^2
4	011	α^3
5	110	α^4
6	111	α^5
7	101	α^6

Обратный элемент в конечном поле может быть найден следующим образом: пусть $\beta \in GF(q)$, тогда $\beta^{-1} = \beta^{q-2}$.

Доказательство. Найдем произведение $\beta \cdot \beta^{q-2} = \beta^{q-1} = 1$ (по свойству 2), что и доказывает утверждение.

Можно показать, что все операции в конечном поле $GF(q)$ требуют $O\left(\log^3 q\right)$ битовых операций. Возведение в *k*-ю степень любого элемента $a \in GF(q)$ требует $O\left(\log^k \log^3 q\right)$ битовых операций при использовании *быстрого алгоритма* с представлением числа*k* в двоичном виде[9].

Изоморфизм конечных полей становится более понятным с помощью табл. 3.12.

Выберем другой неприводимый многочлен $f(x) = x^3 + x^2 + 1$, образующий поле $GF(2^3)$.Тогда для него можно будет составить другую таблицу представления элементов поля $GF(2^3)$ степенями примитивного элемента γ, который является корнем многочлена $x^3 + x^2 + 1$.

Таблица 3.12

Представление элементов поля $GF(2^3)$ степенями примитивного элемента при $f(x) = x^3 + x^2 + 1$

№ п/п	Представление в виде последовательности	Представление степенью примитивного элемента
0	000	0
1	001	1
2	010	γ
3	100	γ^2
4	101	γ^3
5	111	γ^4
6	011	γ^5
7	110	γ^6

Изоморфизм поля $GF(2^3)$, определеного табл. 3.11 и поля $GF(2^3)$, определеного табл. 3.12, задается соотношением $\alpha \leftrightarrow \gamma^3$. Отображение первого поля во второе показано на рис. 3.10.

Действительно, $\alpha^6 = \gamma^{18} = \gamma^7 \cdot \gamma^7 \cdot \gamma^4 = \gamma^4$. (Аналогично проверяются и другие соответствия.)

Определение 3.1.17. Неприводимый многочлен $p(x)$ над полем $GF(p)$ называется *примитивным многочленом*, если его корень (принадлежащий $GF(p^n)$), является примитивным элементом этого поля.

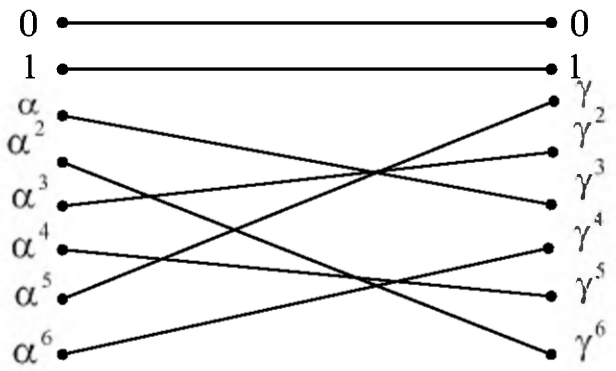


Рис. 3.10. Изоморфизм поля $GF\left(2^3\right)$ на неприводимом многочлене $x^3 + x + 1$ и поля $GF\left(2^3\right)$ на неприводимом многочлене $x^3 + x^2 + 1$

Свойства многочленов над конечными полями **([8])**

1. Многочлен $x^{p^n} - x$ равен произведению всех нормализованных неприводимых над $GF(p)$ многочленов, степени k которых делят n. *(Нормализованный многочлен* – это многочлен, у которого коэффициент при старшем элементе равен 1.)

2. Неприводимые над $GF(p)$ многочлены не имеют в поле $GF(p^n)$ кратных (совпадающих) корней.

3. Если $f(x)$ – неприводимый многочлен степени n над $GF(p)$ и $\beta \in GF(p^n)$ является корнем многочлена, т. е. $f(\beta)=0$, то элементы $\beta^p, \beta^{p^2}, ..., \beta^{p^{n-1}}$ определяют всю совокупность корней этого многочлена.

4. Все корни неприводимого многочлена имеют один и тот же порядок.

Эквивалентное определение примитивного многочлена. Неприводимый многочлен $f(x)$ степени n над полем $GF(p)$ будет примитивным тогда и только тогда, когда

$$f(x) \mid x^{p^n-1}-1,\ f(x) \nmid x^{n'}-1,\ \forall n' < p^n-1.$$

Оценим количество всех неприводимых и примитивных многочленов степени n над полем $GF(p)$. Обозначим через $I_p(n)$ количество всех неприводимых нормированных многочленов степени n над полем $GF(p)$. Тогда [9]

$$I_p(n)=\frac{1}{n}\sum_{d \mid n}\mu(d)\,p^{n/d},$$

где $\mu(d)$ – *функция Мёбиуса*,

$$\mu(d)=\begin{cases}1, \text{если } d=1;\\ 0, \text{если } d \text{ содержит кратные простые множители;}\\ (-1)^k, \text{если } d - \text{это произведение } k \text{ простых чисел.}\end{cases}$$

Пример 3.1.12. Если d = 20, то $\mu(20)=0$, так как $20=2^2\cdot 5$, а если d = 10, то $\mu(10)=1$, так как $10=2\cdot 5$. При больших величинах n существует следующая асимптотическая оценка для $I_p(n)$:

$$I_p(n) \approx \frac{p^n}{n},$$

т. е. мы видим, что при больших n имеется достаточно много неприводимых многочленов.

Найдем теперь количество всех примитивных многочленов степени n над полем $GF(p)$, обозначив его через $N_p(n)$. Тогда [9]

$$N_p(n)=\frac{\varphi\left(p^n-1\right)}{n},$$

где φ – *функция Эйлера*. По определению $\varphi(n)$ равна количеству целых неотрицательных чисел b: $0 \le b < n, \gcd(b,n)=1$, где $\gcd(b,n)$ означает наибольший общий делитель чисел b и n. Легко проверить, что если $n=p$, т.е. n простое число, то $\varphi(p)=p-1$.

Для некоторых n и $p=2$ число 2^n-1 может оказаться простым. В этом случае эти числа называются *числами Мерсенна* (M_n). Можно проверить, что $M_n=(2,3,5,7,...,31,...,11213)$. Для чисел Мерсенна (согласно свойству функции Эйлера)

$$N_2(n)=\frac{2^n-2}{n}.$$

Видно, что для больших n количество примитивных многочленов также весьма велико.

Тесты на нахождение неприводимых и примитивных многочленов. Для нахождения неприводимых и примитивных полиномов можно использовать таблицы [8,9], однако в некоторых криптосистемах неприводимые и примитивные полиномы выбираются в качестве ключей, и тогда общий метод генерирования такого ключа состоит в том, чтобы случайным образом сгенерировать полином $f(x)$ степени nнад полем $GF(p)$, а затем проверить его неприводимость или примитивность. Если требуемое свойство не выполнилось, то генерируется следующий полином, и так до тех пор, пока не получится положительный результат. Так как доля неприводимых и примитивных полиномов достаточно велика, то число попыток будет конечно (и не слишком велико).

Для проверки полиномов $f(x)$ степени n над полем $GF(p)$ на неприводимость используют следующее их свойство [9]: чтобы заданный полином был неприводимым, необходимо и достаточно выполнение условия

$$\gcd\left(f(x),x^{p^i}-x\right)=1,\ 1\le i\le\left[\frac{n}{2}\right].$$

Однако, как будет показано в разд. 3.2, для построения стойких потоковых шифров необходим выбор примитивных полиномов над полем $GF(2)$, когда их степени $n\ge 80$, и поэтому использование такого алгоритма тестирования оказывается вычислительно нереализуемым даже для двоичных ЛРР, для которых p=2. Чтобы упростить вычисления, используется модификация данного алгоритма с приведением степеней многочленов по модулю $f(x)$.

Алгоритм тестирования для больших величин 2^n выглядит тогда следующим образом [9]:

1. Установить $u(x)\leftarrow x$.

2. Для i от 1 до$[n/2]$ выполнить шаги 2.1–2.3.

2.1. Вычислить $u(x)\leftarrow u(x)^2 \bmod f(x)$.

2.2. Вычислить $d(x)=\gcd(f(x),u(x)-x)$.

2.3. Если $d(x)\ne 1$, то $f(x)$ – приводимый полином.

3. Если$d(x)=1$ для всехi, то $f(x)$ – неприводимый полином.

Для того чтобы выполнить шаг 2.2 алгоритма, используется следующая модификация *алгоритма Евклида* для нахождения общего наибольшего делителя целых чисел.

Если $g(x), h(x)$ два полинома над полем $GF(2)$, то их наибольший общий делитель ($\gcd(g(x), h(x))$) находится повторением следующих шагов:

1. Пока $h(x) \neq 0$, выполнить шаг 1.1.

1.1. Установить $r(x) \leftarrow g(x) \bmod h(x),\ g(x) \leftarrow h(x),\ h(x) \leftarrow r(x)$.

2. Выдать $\gcd = (g(x))$.

Для нахождения примитивных полиномов используют следующий тест на примитивность [9]. Пусть задан неприводимый многочлен $f(x)$ степени n над полем $GF(p)$. Допустим, что существует разложение $p^n - 1$ на простые множители. Обозначим их через $r_1, r_2, ..., r_t$. Если выполняется условие $x^{\frac{(p^n-1)}{r_i}} \bmod f(x) \neq 1$, при $i = 1, 2, ..., t$, то $f(x)$ будет примитивным. Если разложение на множители числа $p^n - 1$ не известно, то не известен эффективный алгоритм для тестирования примитивных полиномов.(Если n – число Мерсенна, то любой неприводимый полином будет и примитивным.)

Для возведения x в большую степень по модулю $f(x)$ используется следующий алгоритм, который является обобщением алгоритма быстрого возведения в степень по модулю заданного числа, изучаемого во второй части книги. Пусть необходимо вычислить $g(x)^k \bmod f(x)$, где показатель степени k имеет представление в виде двоичного разложения $k = \sum_{i=0}^{t} k_i 2^i$.

Тогда выполняются следующие шаги:

1. Установить $s(t) \leftarrow 1$. Если $k = 0$, тогда выдать как результат $(s(x))$.
2. Установить $G(x) \leftarrow g(x)$.
3. Если $k_0 = 1$, тогда установить $s(x) \leftarrow g(x)$.
4. Для i от 1 до t выполнить шаги 4.1, 4.2.

4.1. Установить $G(x) \leftarrow G(x)^2 \bmod f(x)$.

4.2. Если $k_i = 1$, то установить $s(x) \leftarrow G(x) \cdot s(x) \bmod f(x)$.

5. Выдать как результат $s(x)$.

3.1.10. Разработка блоковых шифров, доказуемо стойких к линейному и разностному криптоанализу

В работах по криптографии [35] было показано, что если в качестве входов x и выходов $S(x)$ для S-блоков, имеющих одинаковую длину n, рас-

сматривать элементы конечного поля $GF(2^n)$ и выбрать преобразование $S(x)=x^{-1}$, где x^{-1} – *обратный элемент* к элементу x в конечном поле $GF(2^n)$, то вероятности дифференциальной аппроксимации P_D и корреляции $P_L=2\varepsilon$ (где ε – перекос для линейной аппроксимации) минимизируются и принимают значения:

$$P_D=2^{2-n}; \tag{3.35}$$

$$P_L=2^{1-\frac{n}{2}}. \tag{3.36}$$

Доказывается также, что нелинейность такого *S*-блока будет тогда равна:

$$N(f)=n-1. \tag{3.37}$$

Поскольку строгое доказательство, изложенное в [35], того факта, что $S(x)=x^{-1}$ – это оптимальное преобразование (с точки зрения защиты от линейного и дифференциального криптоанализа), является весьма сложным, приведем лишь некоторые пояснения.

Действительно, обозначим входную разность при дифференциальном криптоанализе через α, а выходную – через β. Тогда при использовании в *S*-box преобразования $S(x)=x^{-1}$ мы получим следующее уравнение для нахождения множества входов x, обеспечивающих при входной разности α выходную разность β:

$$(x+\alpha)^{-1}-x^{-1}=\beta.$$

В предположении, что $x\neq 0$ и $x\neq -\alpha$, это уравнение будет эквивалентно следующему:

$$\beta x^2+\alpha\beta x-\alpha=0.$$

Данное уравнение имеет в поле $GF(2^n)$ самое бо̀льшее 2 решения. Если $x=0$ или $x=-\alpha$ являются решениями первого уравнения, то это будет, только, когда $\beta=\alpha^{-1}$.

В этом случае последнее уравнение преобразуется к виду

$$x^2+\alpha x-\alpha^2=0,$$

оно может дать два дополнительных решения к исходному уравнению.

Действительно, возводя в квадрат обе части последнего уравнения и делая подстановку $x^2=\alpha x+\alpha^2$ (что справедливо для поля $GF(2^n)$ характеристики 2), получим

$$x(x^3+\alpha^3)=0,$$

которое не будет иметь других решений, если $\gcd(3,2^n-1)=1$ или (что одно и то же) когда n – нечетное. Если же n – четное, тогда 3 делит 2^n-1 и появляются еще 2 дополнительных решения $x=\alpha^{1+d}$ и $x=\alpha^{1+2d}$, где $d=\frac{1}{3}\left(2^n-1\right)$.

Таким образом, в самом плохом для разработчиков шифра случае исходное уравнение имеет 4 решения. Поэтому вероятность появления фиксированной выходной разности будет не больше, чем $\frac{4}{2^n} = 2^{2-n}$, что совпадает с (3.35).

Однако для того чтобы показать устойчивость блокового шифра к линейному и дифференциальному криптоанализу, необходимо учесть и структуру всего шифра. В частности, использование простых перестановок, которые были представлены в схеме учебного шифра, а также которые встречаются во многих типах реальных блоковых шифров, не является наилучшим способом построения блоковых шифров, устойчивых к линейному и дифференциальному криптоанализу. Действительно, корреляция и вероятность разностной аппроксимации для всего шифра подчиняются следующему неравенству [11]:

$$P_{\text{ш}(L,D)} \le P_{L(D)}^{\frac{(r\beta+2)}{2}}, \qquad (3.38)$$

где P_L, P_D – корреляция и вероятность разностной аппроксимации для входящих в шифрS-блоков; r – число раундов полного шифра; $\beta_{L(D)}$ – *коэффициент ветвления шифра* относительно линейной или разностной аппроксимации, который для простых перестановок всегда будет равен 2.

Для того чтобы увеличить коэффициент ветвления и тем самым повысить устойчивость шифра как к линейному, так и к дифференциальному криптоанализу, необходимо заменить простые перестановки между раундами на более сложные преобразования.

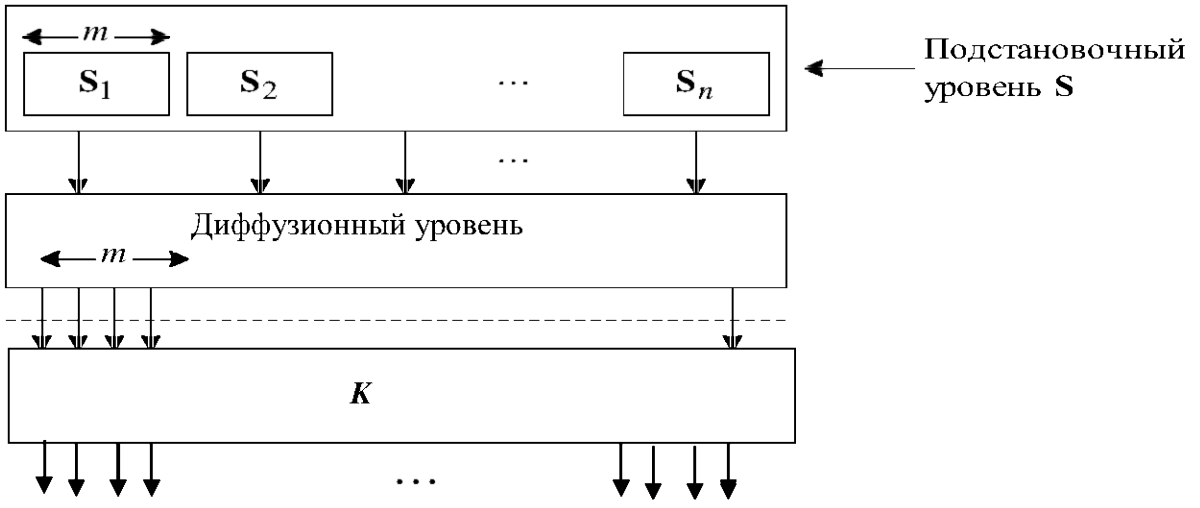


Рис. 3.11. SD-преобразование

Рассмотрим для этого блоковый шифр, построенный на повторении так называемых SD-*преобразований* и показанный на рис.3.11. В аббревиатуре SD буква S означает *подстановочный* (substitution) уровень, а буква D – *рассредоточивающий* (diffusion) уровень. В этом случае вход и выход SD-преобразования, длины которых равны длине блока шифра, разбиваются на n подблоков длины m каждый. Заметим, что здесь буква n использу-

ется не для обозначения длины блокового шифра, как раньше (теперь эта длина будет равна nm).Подблоки длиной m рассматриваются теперь как элементы поля $GF(2^m)$, над которыми и производится преобразование D.

Для тогочтобы увеличить коэффициент ветвления β, необходимо вместо простых перестановок использовать более общее *линейное преобразование*, задаваемое при помощи некоторой матрицы $B_{n\times n}$, и специальным образом выбрать саму эту матрицу. Доказывается [10], что максимальное значение коэффициента ветвления, равное $\beta_{L(D)} = n+1$, где n – число S_i-блоков, достигается в том случае, когда эта матрица $B_{n\times n}$ задается следующим образом.

Пусть имеется укороченный код *Рида–Соломона* (над полем $GF(2^m)$ с параметрами $(2n, n, n+1)$, где $2n$ – длина кода, n – число информационных символов, $n+1$ – величина минимального кодового расстояния) [8]. Порождающая матрица этого кода всегда может быть представлена в следующем *каноническом* виде: $\left[I_n B_{n\times n}\right]$, где I_n – единичная $n\times n$ матрица:

$$\begin{pmatrix} 1 & 0 & 0 & \dots \\ 0 & 1 & 0 & \dots \\ 0 & 0 & 1 & \dots \\ \vdots & \vdots & \vdots & 1 \end{pmatrix},$$

$B_{n\times n}$ – нетривиальная $n\times n$подматрица порождающей матрицы с элементами из поля $GF(2^m)$.Тогда в качестве наилучшего преобразования для диффузного уровня необходимо выбрать именно матрицу $B_{n\times n}$.

Итак, достаточно просто построить блоковый шифр, который будет устойчив как к линейному, так и к дифференциальному криптоанализу. (Такой шифр называется *доказуемо стойким* по отношению к таким видам криптоанализа.) Для этого можно выбрать структуру шифра, которая показана на рис.3.11 с неопределенными пока параметрами: m – размерность S-блоков; n – число S-блоков; r – число раундов шифра.

Если после этого в качестве преобразований S взять обращение элементов в поле $GF(2^m)$, а в качестве преобразований D выбрать умножение на матрицу $B_{n\times n}$, взятую как нетривиальную часть порождающей матрицы укороченного кода Рида–Соломона, то для такого блокового шифра получаем из (3.38) следующие границы для вероятностей успешного осуществления линейного и дифференциального криптоанализа:

$$P_{\text{ш}(L)} \le 2^{\frac{\left(1-\frac{m}{2}\right)(r(n+1)+2)}{2}}; \qquad (3.39)$$

$$P_{ш(D)} \le 2^{\frac{(2-m)(r(n+1)+2)}{2}}. \qquad (3.40)$$

Тогда количество пар «открытый текст/криптограмма», требуемых для анализа, будет примерно равно $\frac{1}{P_{ш(L)}}$ для дифференциального и $\frac{1}{P^2_{ш(D)}}$ для линейного криптоанализа.Задаваясь теперь максимально допустимым количеством таких пар (с точки зрения как организационной, так и вычислительной сложности), можно подобрать параметры блокового шифра m,n,r, которые будут удовлетворять этим требованиям.

Пример. Выберем m=8, n=8, r=5. Тогда по (3.39), (3.40) получаем $P_{ш(L)} \le 2^{-69}$, $P_{ш(D)} \le 2^{-138}$,что обеспечивает достаточно высокую стойкость по отношению как к линейному, так и к дифференциальному криптоанализу.

Мы рассмотрели достаточно общий метод построения хороших блоковых шифров, однако он обладает существенными недостатками:

1) данный метод не является наиболее экономным с точки зрения обеспечения максимальной скорости шифрования как при программной, так и при аппаратной реализациях;

2) структура шифра с выбором параметров по (3.39),(3.40) не гарантирует еще его стойкости относительно других методов криптоанализа, которые будут кратко рассмотрены далее.

3.1.11. Другие методы криптоанализа

Криптоанализ по методу максимального правдоподобия. Рассмотрим последовательно две возможные атаки: при неизвестном открытом сообщении и при известном открытом сообщении.

Первая атака

E – известно, M – не известно, K – не известно.

Предполагается, что криптоаналитик знает распределение вероятностей на сообщении $P(M)$, и поэтому он может, зная структуру шифра, определить условное распределение вероятностей $P\left(E \mid K, P(M)\right)$ как функцию неизвестного ключа. Тогда анализ по методу максимального правдоподобия будет состоять в нахождении такого ключа $\tilde{K}$, который обеспечит максимум этой условной вероятности, т. е. $\tilde{K} = \arg\max_K P\left(E \mid K, P(M)\right)$.Очевидно, что при нахождении $\max$ невозможно использовать переборный метод, поскольку невозможно в обозримое время перебрать все ключи.Поэтому в ряде работ, например в [12], предлагается использовать более эффективные методы нахождения максимумов, в частности градиентный метод.

Вторая атака

E – известно, *M* – известно, *K* – не известно.

В этом случае для решения задачи поиска ключа по методу максимального правдоподобия необходимо задать неопределенную пока вероятность распределения на ключе:

$$P(K)=\prod_{i=1}^{N}P(K_i);\quad P(K_i=1)=P_i;\quad P(K_i=0)=1-P_i.$$

Теперь можно попытаться найти это распределение по методу максимального правдоподобия:

$$\tilde{P}(K)=\arg\max_{P(K)}P\left(E\mid P(K),M\right). \tag{3.41}$$

Наконец, по найденному из (3.41) вектору вероятности ключа $\tilde{P}(K)=\left(\tilde{P}_1,\tilde{P}_2,....,\tilde{P}_N\right)$ производится его квантование для нахождения бит ключа:

$$K_i=\begin{cases}1 & \tilde{P}\left(K_i=1\right)\geq 1/2;\\ 0 & \tilde{P}\left(K_i=1\right)<1/2.\end{cases}$$

Однако при выполнении криптоанализа по данному методу, возникают две существенные проблемы:

1) как аналитически записать функцию под аргументом максимума в (3.41) для конкретного заданного блокового шифра;

2) как найти экстремум функции в (3.41).

Первая задача решается сравнительно просто, поскольку при рассмотрении шифров типа ППШ, зная сообщение, легко записать распределение вероятности после сложения с первым ключом на выходе*S*-блока для 1-го раунда (рис.3.4).

Действительно, так как структура *S*-блока известна, все его выходы можно представить в виде булевых функций от входа, причем в данном случае удобно представление в*дизъюнктивной нормальной форме* (т. е. когда выход формируется из входов при помощи сочетания операций «и», «или», «нет»).

После получения такого представления необходимо вместо булевой переменной на входе подставить вероятность *p*, рассчитанную для символа 1 в этой переменной, а операции «и», «или», «нет» заменить соответственно на операции умножения, сложения и вычисления обратной вероятности $(1-p)$.

Далее, производя вычисления аналогичным образом для всех последующих раундов, можно дойти до вычисления вероятности криптограммы в (3.41). Таким образом, первая задача может быть успешно решена даже для достаточно сложных шифров.

При решении второй задачи необходимо использовать *градиентные итеративные* методы, которые, к сожалению, не всегда сходятся к правильному решению, т. е. к истинному ключу. Экспериментально показано [12], что такой метод допускает, например, вскрытие криптосистемы DES, если в ней используется только 3-4 раунда, вместо 15, положенных по стандарту.

Таким образом, можно сделать общий вывод, что криптоанализ, основанный на методе максимального правдоподобия, не позволяет вскрыть мощные блоковые шифры, но имеет определенные перспективы своего применения в будущем, особенно в сочетании с другими методами криптоанализа.

Криптоанализ по методу решения систем нелинейных уравнений.

Как было отмечено ранее, блоковый шифр может быть описан при помощи булевых функций, связывающих биты открытого сообщения, биты криптограммы и биты неизвестного ключа. Эти булевы функции можно представить в виде полиномов Жегалкина, т. е. в виде уравнений, содержащих произведения и суммы этих переменных по модулю 2. Тогда, составив систему уравнений для полного шифра при нескольких известных блоках открытых сообщений и соответствующих им криптограмм, полученных с использованием одного и того же ключа, можно попытаться решить эту систему относительно бит ключа. Однако в отличие от линейной системы уравнений, для которой существуют регулярные методы решения с полиномиальной сложностью относительно ее размеров, общих методов решения нелинейных систем уравнений (даже в том случае, когда это решение заведомо существует) не известно. Заметим также, что для усложнения описания системы линейных уравнений в современных шифрах стараются избегать задания нелинейных преобразований в «явном виде» (т. е. при помощи аналитических соотношений), заменяя их или дополняя табличными описаниями (см. шифры DES и ГОСТ, разд. 3.1.14).

В [13] рассматривались различные частные подходы к решению такой нелинейной системы уравнений, однако для больших длин ключа и сложных шифров не удается решить задачу по его нахождению в обозримое время. (Заметим, что линейный и дифференциальный криптоанализ, посуществу, представляют собой специальные вероятностные методы решения такой системы уравнений.)В ряде последних исследований [14] предлагается использовать специально разработанный алгоритм, который эффективно работает в том случае, когда число уравнений значительно больше числа переменных, а также когда каждое уравнение является «редким», т. е. в него входит относительно мало слагаемых.

Для того чтобы выполнить первое из этих условий, используется тот факт, что иногда уравнение, описывающее каждый *S*-блок, порождает несколько дополнительных уравнений относительно координат этого *S*-блока. Так, например, в шифре AES, который будет рассматриваться по-

дробно в разд. 3.1.14, в качестве преобразований, выполняемых*S*-блоками, используется обращение элементов в конечном поле $GF(2^m)$, где m – длина входа и выхода для *S*-блока. (Напомним, что такой выбор *S*-блока обеспечивает наилучшую стойкость шифра по отношению к линейному и дифференциальному криптоанализу.) Тогда, если обозначить через x иу вход и выход *S*-блока соответственно, дополнительные уравнения, описывающие этот блок, появляются при умножении обеих частей основного уравнения на x и y: $x^{-1} = y \Rightarrow x \cdot y = 1 \Rightarrow x = x^2 y,\ \ y = x \cdot y^2$.

Казалось бы, эти уравнения не дадут ничего нового, однако если записать их в двоичных координатах, то они оказываются линейно независимыми от основного уравнения, и при построении всего множества таких дополнительных уравнений, описывающих полный шифр, решение подобной системы значительно упрощается.

Существование такого подхода позволяет сделать вывод, что при оценке защищенности блоковых шифров необходимо учитывать их стойкостьне только к линейному и дифференциальному анализу, но и по отношению к данной атаке, основанной на решении нелинейной системы уравнений. Для повышения устойчивости каждого *S*-блока к такому виду криптоанализа в [14] предлагается при его разработке минимизировать следующий параметр *S*-блока:

$$\Gamma = \left(\frac{t}{m}\right)^{\left(t/r\right)},$$

гдеm – размерность *S*-блока; t – максимальное число ненулевых членов в уравнениях, описывающих этот блок; r – число уравнений, описывающих этот блок с учетом дополнительных уравнений.

Криптоанализ, основанный на физических атаках.Существует два типа физических атак на аппаратно реализованный шифр: пассивные и активные. Первый тип использует электромагнитное излучение в окружающее пространство или небольшие флюктуации в цепях питания устройства шифрования (дешифрования), которые в той или иной степени отражают выполнение операций, реализующих эти процедуры.

Рассмотрим для определенности так называемую *утечку по цепям питания*, когда контролируется потребляемая мощность. Эта мощность квантуется по времени и амплитуде и запоминается в ПК в течение шифрования большого количества сообщений на заданном, но не известном ключе, содержащемся в невскрываемоммодуле.

На ПК рассчитывается целевая функция разностного криптоанализа мощности (DPA) следующего вида:

$$T = \frac{1}{L}\sum_{i=1}^{L}\left(\tilde{\alpha}(i) - \frac{1}{2}\right)S(i),$$

где L – общее количество используемых пар «сообщение/криптограмма»; $\tilde{\alpha}(i)$ –оценканекоторого двоичного бита, вырабатываемого алгоритмом шифрования при фиксированном выборечастисекретногоключа и известной криптограмме, полученной на правильном ключе; $S(i)$ – отсчет сигнала источника питания, соответствующий по времени iвыработке оцениваемого символа $\tilde{\alpha}(i)$ при использовании правильного ключа.

Модель отсчета $S(i)$ можно определить следующим образом:

$$S(i) = a_1 + \varepsilon_1(i), \text{ если } \alpha(i) = 1;$$
$$S(i) = a_0 + \varepsilon_0(i), \text{ если } \alpha(i) = 0,$$

где a_1, a_0 – любые вещественные числа; $\varepsilon_1(i), \varepsilon_0(i)$ – случайные (шумовые) величины; $\alpha(i)$ – истинное значение рассматриваемого бита на i-м отсчете при выборе правильного ключа.

Если выполнены условия:

1) $\tilde{\alpha}(i)$ – хорошая оценка $\alpha(i)$;

2) для вычисления $\tilde{\alpha}(i)$ достаточно выполнить обозримый перебор части секретного ключа;

3) $a_0 \neq a_1$;

4) отсчеты шума $\varepsilon_0(i), \varepsilon_1(i)$ слабо зависимы по «i», – то можно с высокой надежностью определить правильный ключ при помощи обозримого перебора части ключа по максимуму значения целевой функции Т. Впервые такая атака была изучена в [36], причем в дальнейшем при помощи данного метода было взломано более 100 различных аппаратно реализованных шифров, включая 4 чиповые смарт-карты.

В [37] подробно рассмотрена возможность взлома аппаратно реализованного шифра ГОСТа таким методом.

Важно отметить, что такие простые методы защиты от подобной атаки, как экранирование или зашумление цепей питания, оказываются неэффективными и приводят лишь к некоторому увеличению времени криптоанализа.

Схемный (алгоритмический) метод защиты от атак DPA (и различных ее модификаций) подробно описан в [38].

Второй тип атак относится к классу активных, когда на аппаратно реализованный шифр оказывается внешнее физическое воздействие.Сущность этой атаки [15] состоит в том, что в алгоритм шифрования принудительно вносятся ошибки, приводящие к инвертированию бит на определенных местах, например тогда, когда ошибка вводится в один из битов на входе S-блока последнего раунда перед сложением с раундовым ключом.

Если теперь наблюдать результаты шифрования до и после внесения ошибок, то иногда удается определить некоторые биты ключа последнего

раунда. Повторяя эту процедуру по отношению к другим битам алгоритма, можно получить постепенное вскрытие всего секретного ключа. Хотя эта процедура весьма трудоемка и требует значительных технологических ухищрений, однако и ее нельзя полностью исключать из рассмотрения. Поэтому современные шифры строятся с учетом и подобной атаки [16]. (Более подробно она будет рассмотрена во второй части книги.)

Метод, основанный на использовании «слабых» ключей.

Различают две основные разновидности понятия «слабый» ключ:

-неудачно сгенерированный случайный ключ;

-плохо разработанная система формирования раундовых ключей из основного ключа.

В первом случае неудачный выбор ключа может позволить криптоаналитику провести тот или иной вид криптоанализа значительно быстрее, чем это делается в общем случае. Так, в частности, для алгоритма шифрования DES(он будет рассмотрен в разд. 3.1.14) существует 4 слабых ключа $\tilde{K}$, которые для 2^{32} из2^{64} возможных входных блоковM дают вообще отсутствие всякого шифрования, т. е. выполнение условия $f(\tilde{K},M)=M$ [9]. Конечно, при случайном и равновероятном выборе ключей вероятность попадания на такие слабые ключи будет ничтожна мала, однако при отклонении от этого условия такие ситуации становятся возможными.

Во втором случае неудачный выбор алгоритма генерирования раундовых ключей может намного облегчить криптоаналитику задачу нахождения основного ключа. Однако рассмотрение данного вопроса выходит за рамки книги «Основы криптографии».

3.1.12. Модификации блоковых шифров

Модификация в виде электронной кодовой книги (ECB)

До сих пор рассматривалась лишь одна из *модификаций* (или, иначе говоря, *мод* блоковых шифров) в виде так называемой *электронной кодовой книги*, при которой каждый блок сообщения шифровался независимо от других блоков в блок криптограммы на одном и том же ключе (рис. 3.12).

Данный способ шифрования имеет следующие свойства:

1) одинаковые блоки сообщения будут всегда шифроваться одинаковыми криптограммами (при условии, конечно, сохранения прежних ключей). Это, на самом деле, отрицательное свойство, поскольку оно дает некоторую информацию перехватчику. Действительно, если уже было известно, что некоторым блоком криптограммы зашифровано определенное сообщение (скажем, имя пользователя), то, встретив ту же самую криптограмму в следующий раз, перехватчик знает, что ее отправил тот же самый пользователь;

2) блоки шифруются независимо, и это позволяет осуществить подмену информации без факта обнаружения такой подмены законным пользователем, имеющим правильный ключ. Для пояснения этого отрицательного свойства рассмотрим следующий пример. Пусть, как это иногда бывает принято, выполняется денежный перевод из одного банка в другой в зашифрованном виде. Формат этого перевода (не обязательно в точности соответствующий действительности, но отражающий суть проблемы), представлен ниже.

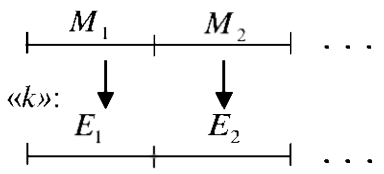


Рис. 3.12. ECB-мода блокового шифра

Время отправки	Банк отправитель	Банк получатель	Имя вкладчика	Номер счета	Сумма вклада
6 байт	12 байт	12 байт	48 байт	16 байт	8 байт

Для выполнения подлога злоумышленник сначала переводит небольшую сумму денег на свой счет. Далее, зная формат, показанный выше, он запоминает 4-ю и 5-ю части своего перевода, перехватывает перевод, выполнявшийся банком для другого лица, подменяет 4-ю и 5-ю его части своими, запомненными раньше, и направляет измененное полное сообщение к другому банку. Вследствие такой махинации деньги (в неизвестном пока количестве и от неизвестного клиента банка) будут переведены на его (вполне известный…) счет. Оставляя пока в стороне проблему технической сложности подобной махинации, можно констатировать факт, что с криптографической точки зрения такой метод шифрования не защищен от подмены сообщений, поскольку при дешифровании не будет обнаружено никаких нарушений смысла сообщений. Для устранения перечисленных выше недостатков используют другие модификации блоковых шифров.

Модификация с зацеплением блоков (CBC-мода)

В этом случае формирование последовательных блоков криптограмм $i = 1, 2,\ldots$ выполняется по следующему алгоритму:

$$E_i = f_K\left(E_{i-1} \oplus M_i\right),\ i = 1, 2,..., \tag{3.42}$$

где $E_0 = IV$ – начальный вектор (некоторый открытый блок, согласованный заранее на передаче и приеме); E_i, M_i – блоки i-й криптограммы и сообщения; $f_K(\cdot)$ – преобразование шифрования, выполняемое на ключе K; $\oplus$ – операция побитного сложения по mod 2. Легко проверить, что восстановление последовательных блоков сообщений M_i для этой моды можно получить, используя следующий алгоритм:

$$M_i = E_{i-1} \oplus g_K(E_i), \tag{3.43}$$

где $g_K(\cdot)$ – преобразование дешифрования, выполняемое на том же ключе K.

Схематически шифрование представлено на рис. 3.13.

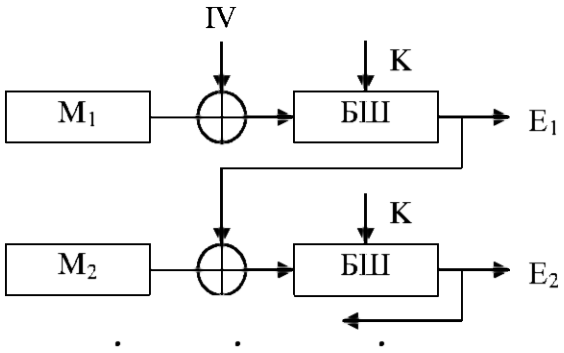


Рис. 3.13. CBC-мода блокового шифра:
БШ – блоковый шифратор

Свойства CBC-модификации:

1) изменение начального вектора *IV* приводит к различной последовательности криптограмм для одного и того же ключа, что даже при повторении блоков сообщений повышает их секретность;

2) перестановка или замена блоков криптограммы приводят к ошибкам дешифрования и, следовательно, обнаруживаются по нарушению смыслового содержания сообщений;

3) единичная ошибка в блоке криптограммы E_i, как видно из алгоритма дешифрования (3.43), приводит к искажениям текущего M_i и последующего M_{i+1} блоков сообщения. В блоке M_i это может привести к большому *размножению ошибок*, и фактически блок M_i оказывается недешифруемым даже при единичной ошибке в блоке криптограммы E_i. (Заметим, что такое же свойство справедливо и для модификации в виде кодовой книги.) В следующем же блоке сообщения M_{i+1} ошибки произойдут в тех же самых местах, где они произошли в блоке E_i и, очевидно, без всякого их размножения. В дальнейшем при отсутствии ошибок в блоках криптограммы все последующие блоки сообщения: M_{i+2},…, – восстановятся верно;

4) для правильного дешифрования сообщения необходимо знать границы блоков. (Это требование (*синхронизации по блокам*), очевидно, справедливо и для модификации в виде кодовой книги.) Хотя атака с подменой блоков, продемонстрированная ранее для модификации ECB, в этом случае не проходит, однако существуют более изощренные атаки, которые могут нарушить *целостность данных* в широком понимании этого термина [9]. Кроме того, если какие-либо два блока криптограмм E_i и E_j совпадают, то в соответствии с алгоритмом дешифрования (3.43) получаем

$$E_{i-1} \oplus E'_{j-1} = M_i \oplus M_j . \qquad (3.44)$$

Поскольку сообщения являются, как правило, избыточными, то уравнение (3.44) позволяет иногда восстановить M_i или M_j. Для предотвращения такой неожиданной атаки необходимо обеспечить маловероятность совпадения криптограмм, а для этого на каждом сеансе связи следует использовать чисто случайные векторы инициализации *IV*, которые могут затем быть переданы в открытом виде для выполнения процедуры дешифрования.

Модификация с обратной связью по криптограмме (CFB-мода)

Недостаток 1-й и 2-й модификаций заключается в том, что при их использовании нельзя начать дешифрование текущего блока сообщения, пока не принят целиком соответствующий ему блок криптограммы. В некоторых приложениях это не удобно, так как требуется дешифровать порции данных меньшие, чем размер блока. Данный недостаток устраняется при использовании CFB-модификации.

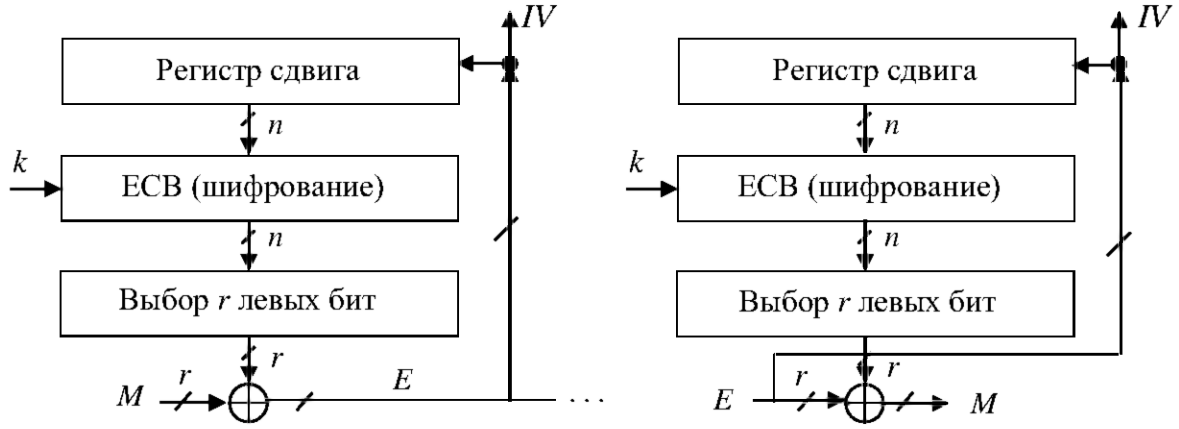


Рис. 3.14. CFB-мода блокового шифра:
а) схема шифрования; б) схема дешифрования

Алгоритм шифрования и дешифрования для CFB-модификации удобнее представить в виде схемы, показанной на рис. 3.14. Здесь регистр сдвига – это одноканальная линия задержки длиной n, и на каждом такте работы этой схемы производится сдвиг влево ее содержимого на r разрядов, причем данные, выходящие из крайнего левого разряда, теряются. ECB-шифрование означает, что используется любой блоковый алгоритм шифрования в режиме электронной кодовой книги. *IV* – это начальный вектор (как и в режиме ECB), который является открытым и должен выбираться случайным на каждом новом сеансе связи. Операция $\oplus$ означает суммирование по mod 2.

Свойства CFB-модификации:

1) изменение начального вектора *IV* для одного и того же ключа приводит к различной последовательности блоков криптограмм даже при по-

вторении блоков сообщения, что повышает секретность сообщений в целом;

2) перестановка или замена блоков криптограммы злоумышленником приводят к ошибкам при дешифровании и, следовательно, обнаруживаются авторизованным пользователем;

3) если происходят ошибки в r-битовом блоке криптограммы, то они будут распространяться на следующие n/r таких блоков сообщений после дешифрования (пока ошибочные биты не выйдут из регистра сдвига), что приводит к размножению ошибок в дешифрованном сообщении на этом интервале;

4) свойство *самосинхронизации.* Если при шифровании оказываются неизвестными границы блоков, например за счет вставок или выпадений двоичных символов, возникающих в процессе передачи их по каналам связи, то это приведет к неправильному дешифрованию не более чем в течение времени передачи n-бит (т. е. пока ошибка не покинет регистр сдвига дешифрования). Данная модификация иногда называется модификацией с самосинхронизацией, поскольку после ошибки синхронизации правильное дешифрование восстанавливается через определенное время без проведения специальной процедуры *ресинхронизации*;

5) в данной модификации алгоритм блокового шифрования используется только в *функции шифрования*. Это важно, например, для случая, когда алгоритм дешифрования является секретным, а алгоритм шифрования – открытым, как это может оказаться в криптосистемах с открытым ключом (см. часть II настоящей книги).

Модификация с обратной связью по шифрующей гамме (OFB)

Алгоритмы шифрования и дешифрования для данной модификации показаны на рис. 3.15., где используются такие же преобразования, как и для модификации CFB, с той лишь разницей, что обратная связь (обновляющая вход блокового шифратора) поступает не как выходная криптограмма полного алгоритма, а как выход внутреннего блокового шифра.

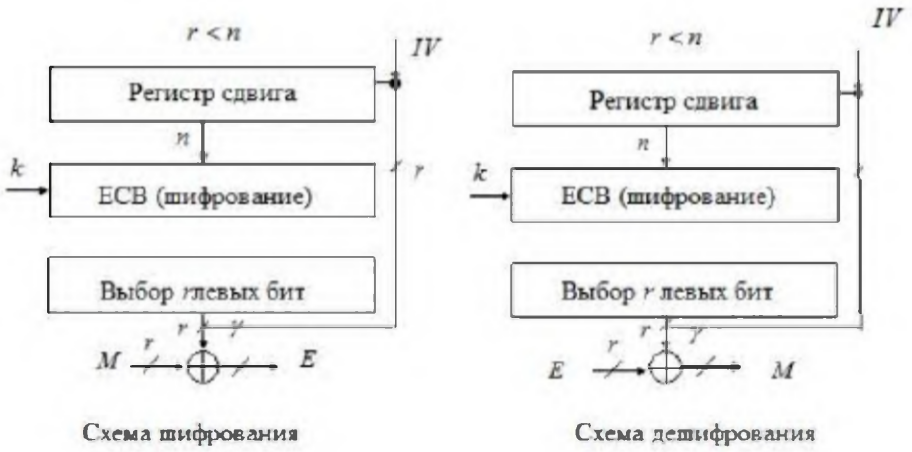


Рис. 3.15. OFB-мода блокового шифра

Видно, что такая модификация блокового шифра, по существу, образует *потоковый шифр* (в смысле определения, данного в разд. 1.3). Однако в отличие от потоковых шифров на основе линейных рекуррентных регистров, которые будут рассматриваться в разд. 3.2, такой потоковый шифр можно назвать *потоковым шифром на основе блокового шифра.*

Свойства OFB модификации:

1) изменение начального вектора *IV* приводит к различной последовательности блоков криптограмм, шифруемых одним и тем же ключом, даже при повторении блоков сообщения, что повышает секретность сообщения в целом;

2) *шифрующая гамма* (γ) не зависит от открытого сообщения, что приводит к отсутствию размножения ошибок после дешифрования, если такие ошибки возникли в криптограмме;

3) для правильного дешифрования сообщений необходимо синхронизировать криптограммы по шифрующей гамме с точностью до бита. Пропадание или вставка битов криптограммы приводит к невозможности правильного дешифрования на всем сеансе связи, если синхронизация не будет восстановлена дополнительными средствами;

4) при каждом новом сеансе связи начальный вектор *IV* должен обязательно выбираться различным, так как в противном случае возможно *простое дешифрование сообщения без знания ключа*. Действительно, пусть это не так, т. е. начальный вектор *IV* не изменяется, и тогда $E_1 = M_1 \oplus \gamma(k)$, $E_2 = M_2 \oplus \gamma(k)$. В этом случае криптоаналитик вычисляет $E_1 \oplus E_2 = M_1 \oplus \gamma(k) \oplus M_2 \oplus \gamma(k) = M_1 \oplus M_2$, что легко поддается дешифрованию с использованием известной избыточности сообщений. Для того чтобы избежать проявления этого «фатального» свойства, необходимо либо каждый раз изменять *IV* по определенному детерминированному закону, известному также и при дешифровании, либо выбирать чисто случайное *IV* на передаче и посылать его по открытому каналу связи на прием, где дешифруется сообщение;

5) в [9] показано, что если обратная связь выбирается полной, т. е. $r = n$, то длина периода гаммы оказывается $\approx 2^{n-1}$ (где n – длина блокового шифра), если же $r < n$, то период гаммы может оказаться $\approx 2^{\frac{n}{2}}$. Поскольку повторение гаммы приводит (см. свойство 4) к возможности дешифрования сообщений без знания ключа, необходимо предусмотреть, чтобы ее период был всегда больше обозримого времени или времени действия одного ключа.

Отметим, что известна еще одна модификация блокового шифрования – *режим счетчика* [9]. В этом случае в схеме (рис. 3.15) отсутствует обратная связь с выхода гаммы ко входу регистра сдвига. Регистр сдвига заме-

няется счетчиком, который генерирует (j + 1)-й входной блок ECB по правилу $I_{j+1} = I_j + 1$, где «+» – обычная арифметическая операция сложения. Положительным свойством такой модификации по сравнению с OFB является наличие периода выходной гаммы, равного $2^n - 1$, независимо от параметра r. Кроме того, i-й блок криптограммы не обязательно должен быть расшифрован, чтобы расшифровать (i + 1)-й блок криптограммы.

Рекомендации по выбору модификаций блокового шифрования:

ECB – выбирается тогда, когда требуется наибольшая простота и максимальная скорость шифрования и нет острой необходимости в защите от навязывания ложной информации;

CBC – это типичная модификация для шифрования данных при программной реализации (достаточно быстрая и защищенная от некоторых атак навязывания);

CFB – целесообразно использовать при аппаратной реализации шифрования и передаче криптограмм в реальном времени по каналам связи, где возможны ошибки синхронизации, но уровень помех достаточно мал;

OFB – целесообразно использовать при аппаратной реализации шифрования и передаче криптограмм в реальном времени по каналам связи со значительным уровнем помех.

3.1.13. Многократное шифрование

В разд. 3.1.14 будет рассмотрен шифр DES, который имеет всего 56 бит ключа и поэтому допускает криптоанализ в реальном времени путем перебора всех ключей с использованием специализированных устройств. Однако DES достаточно популярен в различных прикладных программах и имеет дешевую аппаратную реализацию. Именно поэтому желательно было бы повысить его стойкость, не изменяя существенно алгоритм шифрования и дешифрования. Вполне естественной является попытка увеличения стойкости слабого алгоритма за счет применения многократного шифрования. В этом случае открытое сообщение M сначала шифруется на одном ключе K_1, затем полученная криптограмма E_1 повторно шифруется на втором ключе K_2, образуя новую криптограмму E_2, и так далее вплоть до финальной криптограммы $E = E_l$,полученной после однократного шифрования на ключе K_l криптограммы E_{l-1}. Дешифрование финальной криптограммыE производится очевидным образом с использованием ключей, взятых в обратном порядке: $K_l, K_{l-1}, ..., K_1$. Все ключи в этом случае генерируются независимо друг от друга.

Если производится атака при известных tпарах «открытое сообщение/криптограмма», то можно попытаться найти ключ при однократном

шифровании, используя следующую стратегию: выбрать наугад ключ и проверить, согласуется ли он со всеми парами. Если да, то принять его как *подсказку*. Подсказка, которая не является действительным ключом, называется *ложной подсказкой*.

В [9] доказывается, что при использованииL-кратного шифрования и наличии t пар«сообщение/криптограмма», среднее число ложных подсказок приблизительно равно 2^{Ls-tn}, где n – длина блока шифра, аs – количество бит ключа.

Рассмотрим теперь *двукратное* шифрование(т. е.повторное шифрование полученной криптограммы на новом ключе), схема которого показана на рис. 3.16.

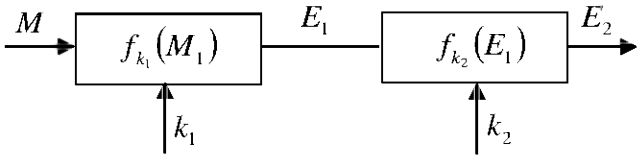


Рис. 3.16. Двукратное шифрование

Ясно, что при использовании такой схемы с алгоритмом шифрования DES, перебор 2^{112} ключей будет невозможным при использовании любых вычислительных средств. Однако оказывается, что такой «наивный» метод повышения стойкости не дает желаемого результата, так как существует более эффективный метод криптоанализа для двукратного шифрования, чем перебор всех пар ключей. Называется этот метод «*встречей в середине*».

Описание метода криптоанализа двукратного шифрования – «встреча в середине».Пусть известны две пары «сообщение/криптограмма»: M_1, M_2 и E_1, E_2. Тогда для двукратного шифрования будут очевидно справедливы следующие соотношения:

$$E_1 = f_{k_2}\left(f_{k_1}(M_1)\right); \tag{3.45}$$

$$E_2 = f_{k_2}\left(f_{k_1}(M_2)\right), \tag{3.46}$$

где f – функция шифрования, а k_1, k_2 – различные ключи.

Обозначим через A результат внутреннего шифрования, т. е. $A = f_{k_1}(M_1)$. Тогда из (3.45) следует, что $g_{k_2}(E_1) = A$, где $g(.)$ – функция дешифрования. Поместим в память множество значений криптограмм $f_{k_1}(M_1)$, а также множество расшифровок $g_{k_2}(E_1)$ для всех возможных ключей алгоритма шифрования DES $k_1, k_2 = 1...2^{56}$ (табл. 3.13).

Таблица 3.13

Дешифрование по методу «встречи в середине»

$f_{k_1}(M_1)$	$f_1(M_1) \dots f_i(M_1) \dots f_{2^{56}}(M_1)$
$g_{k_2}(E_1)$	$g_1(E_1) \dots g_j(E_1) \dots g_{2^{56}}(E_1)$

Очевидно, что объем требуемой для этого памяти будет равен $V_{\text{пам}} = 2 \cdot 2^{56} = 2^{57}$. Всегда найдется хотя бы одна такая пара ключей *i* и *j* (для верхней и нижней строки соответственно), что $f_i(M_1) = g_j(E_1)$. (Действительно, как следует из (3.45), это будет заведомо верно для ключей k_1, k_2, выбранных для получения криптограммы E_1 по сообщению M_1.) Далее надо в табл. 3.13выбрать ключ $k_1 = i$, а ключ $k_2 = j$. Для того чтобы убедиться, что данная параключей не является ложной, проверяется выполнение равенства (3.46) на данных ключах,и если оно не выполняется, то ищется следующая пара ключей k_1, k_2, которая обеспечивает равенство значений в верхней и нижней строках табл. 3.13 и т.д.

В [9] доказывается, что для шифра DES вероятность попадания на ложные ключи при выполнении для них (3.45), (3.46) оказывается равной $2^{-16} \cong 0{,}000015$, что говорит о практической достаточности использования всего двух пар «сообщение/криптограмма» для правильного нахождения ключей шифрования. Действительной проблемой может показаться поиск совпадающих значений в верхней и нижней строках табл. 3.13, однако эта задача может быть решена при запоминании ее верхней строки с *сортировкой* по значениям $f_i(M_1)$. Тогда по мере получения каждого из значений $g_j(E_1)$ их можно будет быстро сравнивать с сортированными значениями первой строки.

Таким образом, получаем пессимистический вывод о том, что *двукратный* DESможет быть подвергнут успешному криптоанализу с использованием порядка 2^{56} операций и 2^{57} ячеек памяти, что практически то же самое, что требуется для переборного криптоанализа при использовании обычного шифра DES. Поэтому переход к двойномуDES не имеет смысла и даже вреден, так как при этом увеличивается расход ключей и уменьшается скорость шифрования. (Этот вывод, очевидно, будет справедлив и для любого другого блокового шифра.) Для действительного повышения стойкости шифра (и в частности, шифра DES) необходимо использовать *трехкратное шифрование.*

Трехкратное шифрование с двумя различными ключами. Для того чтобы обеспечить стойкость шифра при переборе ключей, но уменьшить расход ключевого материала, используется алгоритм двукратного шифрования и однократного дешифрования, в котором, однако, один и тот же ключ встречается лишь дважды. Точнее говоря, этот алгоритм имеет следующий вид:

$$E = f_{k_1}(g_{k_2}(f_{k_1}(M))). \qquad (3.47)$$

Дешифрование при известных ключах k_1, k_2 выполняется очевидным образом:

$$M = g_{k_1}(f_{k_2}(g_{k_1}(E))). \qquad (3.48)$$

В [44] отмечается, что использование функции дешифрования $g(\cdot)$ вместо функции шифрования $f(\cdot)$ не повышает стойкость шифра, а объясняется лишь особенностями разработки стандарта фирмой IBM.

В[9] доказывается, что применительно к шифру DES криптоанализ методом перебора ключей и при атаке с t известными парами «сообщение/криптограмма» требует порядка $2^{120-\log t}$ операций при используемом для криптоанализа объеме памяти t. Легко видеть, что даже при мало реальном выборе числа элементов памяти $t = 2^{40}$ требуемое число операций оказывается 2^{80}, т. е. практически нереализуемым.

Использование более изощренного криптоанализа со *специально выбранными* 2^{56} открытыми сообщениями и соответствующими им криптограммами, позволяет найти ключи k_1, k_2 после выполнения порядка 2^{56} операций и задействования такого же примерно объема памяти. Хотя требуемый вычислительный ресурс в последнем случае значительно меньше, чем в предыдущем, однако выполнение условия по получению 2^{56} пар «сообщение/криптограмма» для сообщений, *специально выбранных* криптоаналитиком, представляется нереальным. Поэтому можно сделать общий вывод, что алгоритм трехкратного шифрования с двойными ключами для DES является стойким относительно наилучших переборных методов криптоанализа.

Заметим, что переход к трехкратному алгоритму шифрования DES с использованием трех различных ключей повышает, безусловно, его стойкость по отношению к переборному методу криптоанализа [9], однако это вряд ли целесообразно, поскольку приводит и к значительному увеличению расхода ключевого материала.

3.1.14. Примеры практически используемых блоковых шифров

DES *(Федеральный стандарт шифрования США (FIPS 46-2), часто используемый в различных прикладных программах и открыто распространяемый в сети Интернет)*

Структура шифра. DES – это блоковый шифр с одинаковой длиной входных и выходных двоичных блоков, равной 64 бита. Он основан на структуре Фейстеля с использованием 16 раундов преобразований, в каждом из ко-

торых выполняется нелинейное преобразование, зависящее от раундового ключа. Нелинейное преобразование выполняется с использованием 8 различных *S*-блоков, имеющих 6 двоичных входов и 4 двоичных выхода и заданных различными для каждого из *S*-блоков таблицами. Перестановки бит в промежуточных блоках раундов задаются одинаковыми и также определяются соответствующей таблицей. Раундовые ключи формируются из основного секретного ключа длиной 56 бит с использованием алгоритма, выполняющего его циклические сдвиги на число бит, зависящее от номера раунда. Дешифрование (согласно свойству схемы Фейстеля) выполняется тем же алгоритмом, что и шифрование, но при использовании раундовых ключей в обратном порядке.

Схема алгоритма шифрования подробно описана в [18].

Стойкость шифра. Шифр DES появился в качестве стандарта в середине 70-х годов прошлого века и был в то время первым и единственным шифром с полностью опубликованным алгоритмом шифрования и дешифрования, где неизвестным оставался только ключ. Это позволило безлицензионно использовать данный метод или пытаться «взломать» данный шифр всем желающим.

Метод криптоанализа данного шифра, основанный на полном переборе всех 2^{56} ключей, требует (как легко подсчитать) на современном персональном компьютере порядка 2300 лет. Однако использование параллельных вычислений и специализированных ЭВМ позволяет найти секретный ключ при атаке только с одной известной криптограммой, практически в «реальном времени», хотя разработка и даже эксплуатация таких спецвычислителей доступна лишь государственным и большим корпоративным структурам.

Алгоритм DES достаточно устойчив как к линейному, так и к дифференциальному криптоанализу. Как отмечено в [9], линейный криптоанализ оказывается успешным с вероятностью 10 % при использовании 2^{38} пар «сообщение/криптограмма» и при выполнении 2^{50} элементарных операций. При дифференциальном криптоанализе необходимо иметь 2^{47} специально выбранных пар «сообщение /криптограмма» и выполнить около 2^{47} элементарных операций. Получить в распоряжение криптоаналитика такое количество сообщений (еще и специально подобранных) и соответствующих им криптограмм практически совершенно нереально. Поэтому более практичным оказывается метод полного перебора ключей. Однако если при шифровании использовался усеченный алгоритм (скажем, состоявший из 3–5 раундов), что может объясняться желанием увеличить скорость шифрования, то методы линейного и дифференциального криптоанализа будут обладать значительным преимуществом перед полным перебором.

Таким образом, хотя известна «народная теорема» о том, что шифр DES является нестойким, но это не совсем так. В действительности его вполне можно использовать при шифровании сообщений не слишком высокого уровня конфиденциальности, когда разумно надеяться, что к их дешифрованию не подключатся государственные структуры или корпорации,

обладающие большим финансовым потенциалом. В тех же случаях, когда эти условия отсутствуют, целесообразно использовать так называемые «усиленные» алгоритмы DES, например тройной DES с двойным ключом.

Скорость шифрования. При программной реализации на современном персональном компьютере скорость шифрования или дешифрования по алгоритму DES имеет порядок 100–200 kB/s. При аппаратной реализации она может быть увеличена в 3–4 раза.

ГОСТ 28147–89. (*Бывший советский (ныне Российский) Федеральный стандарт, рекомендованный к обязательному использованию в организациях, взаимодействующих с государственными учреждениями).*

Структура шифра. ГОСТ – это блоковый шифр с длиной входных и выходных блоков, равной 64 бита, основанный на структуре Фейстеля. Число раундов равно 32, а длина ключа составляет типично 256 бит, однако имеется дополнительный *долговременный ключ* длиной 512 бит.

В каждом раунде выполняются нелинейные преобразования с использованием восьми *S*-box, имеющих 4 входа и 4 выхода и задаваемых таблицами (структура таблиц не задана самим стандартом), и сложением с раундовыми ключами по $\bmod 2^{32}$. (Последняя операция аналогична той, которая используется в известном шифре IDEA [9].) Раундовые ключи длиной 32 бита формируются по заданному стандартом алгоритму из основного ключа длиной 256 бит, который для этого делится на 8 блоков по 32 бита каждый. Один *i*-й раунд схемы шифрования ГОСТ показан на рис. 3.17.

Видно, что левая половина выхода *i*-го раунда L_i (также как и в DES) равна правой половине входа раунда R_{i-1}.

Правая часть выхода *i*-го раунда R_i получается следующим образом: правая половина входа *i*-го раунда R_i складывается по $\bmod 2^{32}$ с 32-битовым раундовым ключом K_i. (Раундовые ключи, используемые на первых 8 раундах, получаются путем простого разбиения 256-битового секретного ключа на восемь 32-битовых блоков. Раундовые ключи с 9-го по 16-й и с 17-го по 24-й являются повторением ключей с 1-го по 8-й раунд, а раундовые ключи с 25-го по 32-й раунд являются ключами с 1-го по 8-й раунд, но взятыми в обратном порядке.) Далее 32-битовая комбинация R_{i-1} разбивается на восемь 4-битовых *S*-box, каждый из которых поступает на вход соответствующего *S*-box.

Выходы всех *S*-box объединяются, образуя 32-битовую комбинацию, над которой производится преобразование перестановки символов (циклический сдвиг на 11 разрядов влево). Наконец, эта комбинация складывается по $\bmod 2$ с 32-битовой комбинацией левой половины L_{i-1}, образуя правую половину R_i на *i*-м раунде.

Процесс дешифрования выполняется аналогичным образом с той лишь разницей, что раундовые ключи используются в обратном порядке.

Стойкость шифра. Очевидно, что криптоанализ методом полного перебора ключей оказывается для этого шифра невозможным, поскольку он потребует необозримо большого времени при использовании любых вычислительных ресурсов. (Так, на типовом персональном компьютере необходимое время составляет порядка 10^{63} лет). Не известно также успешных нападений на этот шифр с использованием линейного или дифференциального криптоанализа. Однако в отличие от шифра DES количество публикаций по криптоанализу ГОСТ невелико, что, в частности, объясняется неопределенностью в задании таблиц, описывающих *S*-блоки. (Этот шифр можно считать шифром с «частично общедоступным алгоритмом».)

Преимуществом алгоритма ГОСТ, повышающим его стойкость (впрочем, как и шифра IDEA), считается использование в нем, помимо табличных операций и операций по mod 2, также операций по $\bmod 2^{32}$.

Таким образом, можно надеяться, что шифр ГОСТ при его программной реализации является стойким относительно любых известных к настоящему времени атак.

Скорость шифрования. При программной реализации на типовом персональном компьютере скорость шифрования имеет порядок 1 Mb/s, а при аппаратной реализации («Криптон-9») – до10 Mb/s.

AES (**AdvancedEncryptionStandard** – *Американский стандарт шифрования FIPS-197*)

Данный алгоритм шифрования является не только полностью общедоступным, но и принятым как наилучший среди представленных на открытом конкурсе в США в 1997 г.

Заметим, что к проектам, представлявшимся на этот конкурс, предъявлялись определенные требования. Криптоалгоритм должен был быть:

- открыто опубликованным;
- симметричным блоковым шифром, допускающим длины ключа 128, 192 и 256 бит;
- предназначенным как для аппаратной, так и для программной реализации;
- доступным для свободного использования в любых продуктах, а значит, он не мог быть запатентован.

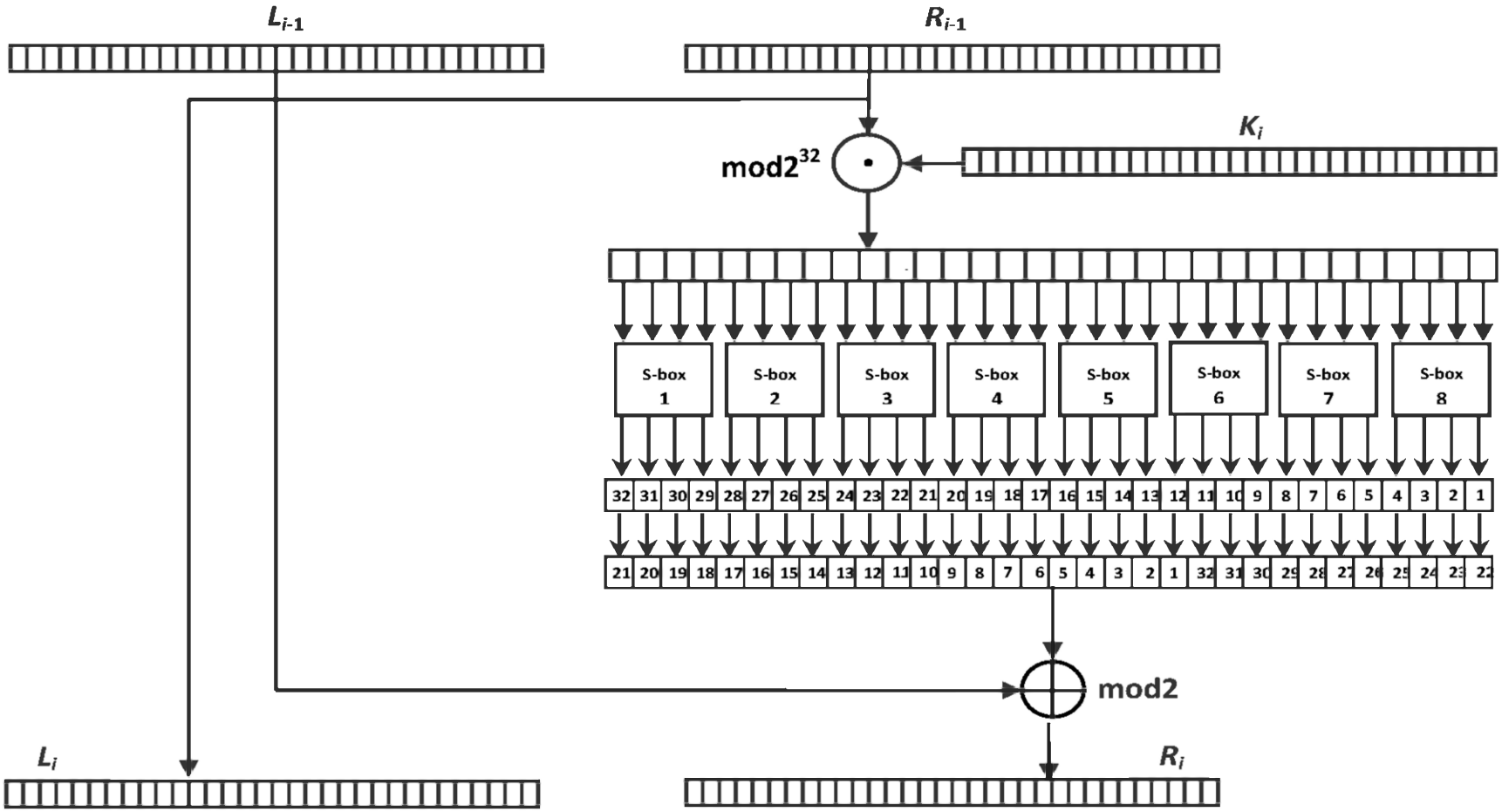


Рис. 3.17. Схема *i*-го раунда шифра ГОСТ

При оценке экспертами конкурсных проектов они подвергались изучению по следующим свойствам: стойкость, стоимость, гибкость (выполнимость в различных средах), использование для выполнения других криптографических функций, реализуемость в смарт-картах.

В 2000 г. конкурс завершился победой бельгийских разработчиков криптоалгоритма RIJNDAEL («Рэйндол»), который был так назван по начальным буквам фамилий его авторов. В 2002 г. вступил в действие новый американский стандарт FIPS-197, соответствующий алгоритму шифрования AES, который не без основания можно назвать *Криптографическим стандартом XXI века*.

Структура шифра. Данный шифр основан на принципе итерирования SD-преобразований (разд. 3.1.10) и использует так называемую архитектуру «квадрат», т. е. все преобразования производятся в рамках одного квадрата, приведенного на рис. 3.18. Текущие данные (в том числе исходное сообщение и получаемая криптограмма) записываются по одному байту (8 бит) в каждую из 16 клеток, что дает общую длину блока шифрования, равную $8\times16=128$ бит.

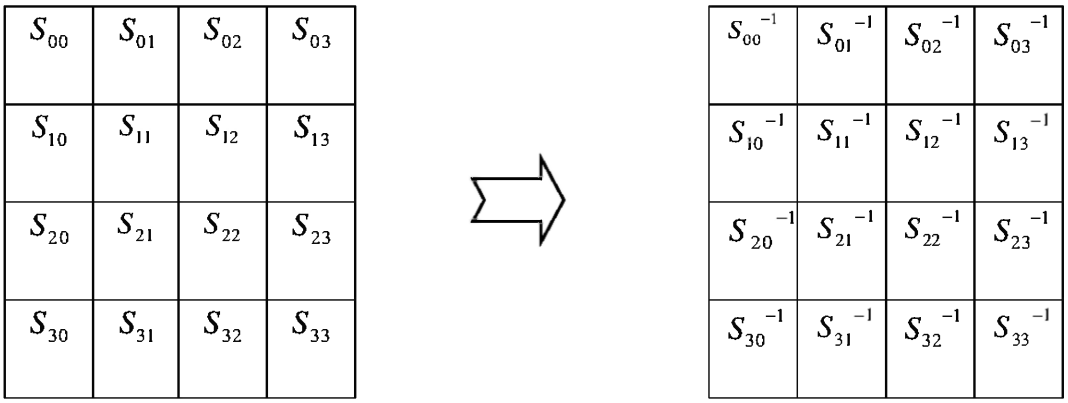


Рис. 3.18. Второе преобразование в шифре AES

Первое преобразование данного алгоритма выполняется как побитовое сложение всех элементов квадрата сообщения со 128 элементами основного ключа. Далее производится вычисление обратного элемента в поле $GF(2^8)$, задаваемом неприводимым полиномом $x^8+x^4+x^3+x+1$, что, как было показано в разд. 3.1.10, обеспечивает доказуемую устойчивость шифра по отношению к линейному и дифференциальному криптоанализу, при этом нулевой элемент поля сохраняется без преобразования (рис. 3.18).

Следующее преобразование состоит в умножении каждой клетки квадрата, представленной в виде двоичного вектор-столбца: $(b_0,b_1,\ldots,b_7)$, – на фиксированную матрицу и добавлении также фиксированного вектор-столбца, причем все операции здесь выполняются в поле $GF(2)$ (рис. 3.19):

$$\begin{pmatrix} b_0' \\ b_1' \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ b_7' \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ b_7 \end{bmatrix} \oplus \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

Используемая в этом преобразовании матрица и вектор-столбец сохраняются одинаковыми на всех раундах и не зависят от ключа. Заметим, что умножение на матрицу и добавление вектора улучшает криптографические свойства шифра для случая, когда в клетках квадрата появляются нулевые элементы.

В качестве очередного преобразования используется побайтовый циклический сдвиг массива сообщений на различное количество байт (клеток), выполняемый, как показано на рис. 3.19.

S_{00}	S_{01}	S_{02}	S_{03}	$\Rightarrow$	S_{00}	S_{01}	S_{02}	S_{03}	Отсутствие сдвига
S_{10}	S_{11}	S_{12}	S_{13}		S_{11}	S_{12}	S_{13}	S_{10}	Сдвиг на 1 клетку влево
S_{20}	S_{21}	S_{22}	S_{23}		S_{22}	S_{23}	S_{20}	S_{21}	Сдвиг на 2 клетки влево
S_{30}	S_{31}	S_{32}	S_{33}		S_{33}	S_{30}	S_{31}	S_{32}	Сдвиг на 3 клетки влево

Рис. 3.19. Преобразование сдвига в шифре AES

Следующее преобразование называется *перемешиванием столбцов.* На этом шаге каждый C-й столбец квадратной матрицы представляется как 4-мерный вектор над полем $GF(2^8)$, и далее производится умножение в этом поле, заданном неприводимым полиномом $x^8 + x^4 + x^3 + x + 1$, на определенную матрицу с элементами из этого же поля:

$$\begin{pmatrix} S'_{0C} \\ S'_{1C} \\ S'_{2C} \\ S'_{3C} \end{pmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \times \begin{pmatrix} S_{0c} \\ S_{1C} \\ S_{2C} \\ S_{3C} \end{pmatrix},$$

где элементы, показанные в этой матрице, задаются как элементы поля $GF(2^8)$ (т. е. как двоичные последовательности длины 8), что иллюстрируется следующим примером: $02 = 00000010$.

0 2

Наконец, производится сложение с раундовыми ключами, которое выполняется просто как побитное сложение всех элементов последнего квадрата с 128 элементами раундового ключа по модулю 2. После завершения одного раунда все описанные выше операции повторяются с использованием других раундовых ключей. Раундовые ключи вырабатываются из единственного секретного ключа длиной 128, 192 или 256 бит (в зависимости от выбранного режима шифрования) с помощью специальных преобразований, включающих в себя циклические сдвиги и расширения. Точный алгоритм выработки раундовых ключей можно найти в [19]. Количество раундов шифра зависит от выбранного режима его работы и изменяется в пределах от 10 до 14. В последнем раунде не выполняется перемешивание столбцов.

Для дешифрования используется последовательность обратных преобразований с обратным порядком следования раундовых ключей, что оказывается вполне возможным, поскольку все операции, выполняемые в каждом раунде, как легко убедиться, обратимы. Однако следует заметить, что в отличие от шифров, основанных на структуре Фейстеля (например, шифра DES), данный шифр должен использовать разные электронные схемы или программы для шифрования и дешифрования соответственно.

Особенности шифра:

1) AES ориентирован в основном на реализацию с 8-разрядными процессорами;

2) все раундовые преобразования выполняются в конечных полях, что допускает простую реализацию на различных платформах.

Стойкость шифра. Очевидно, что перебор всех ключей (даже при их минимальном количестве – (2^{128})) оказывается невозможным. Линейный криптоанализ и дифференциальный криптоанализ также практически невозможны вследствие выбора оптимальных процедур преобразований и, в частности, использования вычисления обратных элементов в конечном поле.

Криптоанализ на основе решения нелинейной системы уравнений над полем GF(2), описывающих шифр, в принципе возможен [14], в том числе и за счет появления дополнительных уравнений (разд. 3.1.11). Однако эта процедура требует необозримо большого вычислительного ресурса. Таким образом, в настоящее время шифр AES можно считать стойким относительно любых известных атак.

Правда, недавно появилась группа атак на шифр AES, которые оказываются эффективнее полного перебора ключей. Это так называемые «бумеранг-атаки» на связанных ключах [43]. В этом случае предполагается, что криптоаналитик помимо знания множества открытых сообщений и со-

ответствующих им криптограмм может задавать разности между используемыми ключами. Для AES-256, где 256 означает длину ключа, такая атака имеет сложность 2^{119}, что все еще остается далеко за пределами практических возможностей. Защита от атак подобного типа может состоять в увеличении количества раундов шифра или в улучшении алгоритма выработки раундовых ключей.

Скорость шифрования. При программной реализации данный алгоритм наиболее эффективно реализуется на 8- и 32-разрядных платформах. Для типичных ПК скорость шифрования может составлять порядка 1Мбит/с – 500 кбит/с. При аппаратной реализации высокие скорости шифрования (порядка 100 Мбит/с и выше) потребуют увеличения аппаратных ресурсов и, следовательно, увеличения габаритов устройства и энергопотребления.

ГОСТ Р 34.12—2015.

Настоящий стандарт содержит описание алгоритмов блочного шифрования и дешифрования и предложен для замены стандарта ГОСТ-28147-89. Необходимость разработки нового стандарта вызвана потребностью в создании блочных шифров с различными длинами блока, соответствующих современным требованиям к криптографической стойкости и эксплуатационным свойствам.

Стандартом определены два алгоритма шифрования для блоков длиной 64 бита и 128 битов. Алгоритм шифрования длиной 64 бита подобен алгоритму шифрования согласно ГОСТ-28147-89, который был рассмотрен ранее, но содержит точные задания всех параметров алгоритма.

Рассмотрим алгоритм шифрования для 128-битных длин блоков, являющийся основным. В этом алгоритме длина базового ключа 256 битов, количество раундов шифрования – 10.

Алгоритм шифрования

Схема одного раунда шифрования показана на рисунке. Он реализуется при помощи следующих шагов:

1. 128 бит информации M складываются по mod2 со 128 битами раундового ключа K_i, i=1,2..9.
2. Получившийся блок из 128 бит разбивается на 16 подблоков по 8 бит каждый.
3. Каждый подблок длины 8 бит преобразуется по следующему алгоритму:
   3.1. Каждая двоичная последовательность V_8 преобразуется в множество чисел Z_{256}, (Например, 10000101→133).
   3.2. Над ним выполняется нелинейное преобразование, заданное таблицей π': Z_{256}→ Z_{256}. Например: 0→252, 1→238,..255→182.

Нелинейные преобразования π': Z_{256}→ Z_{256}.

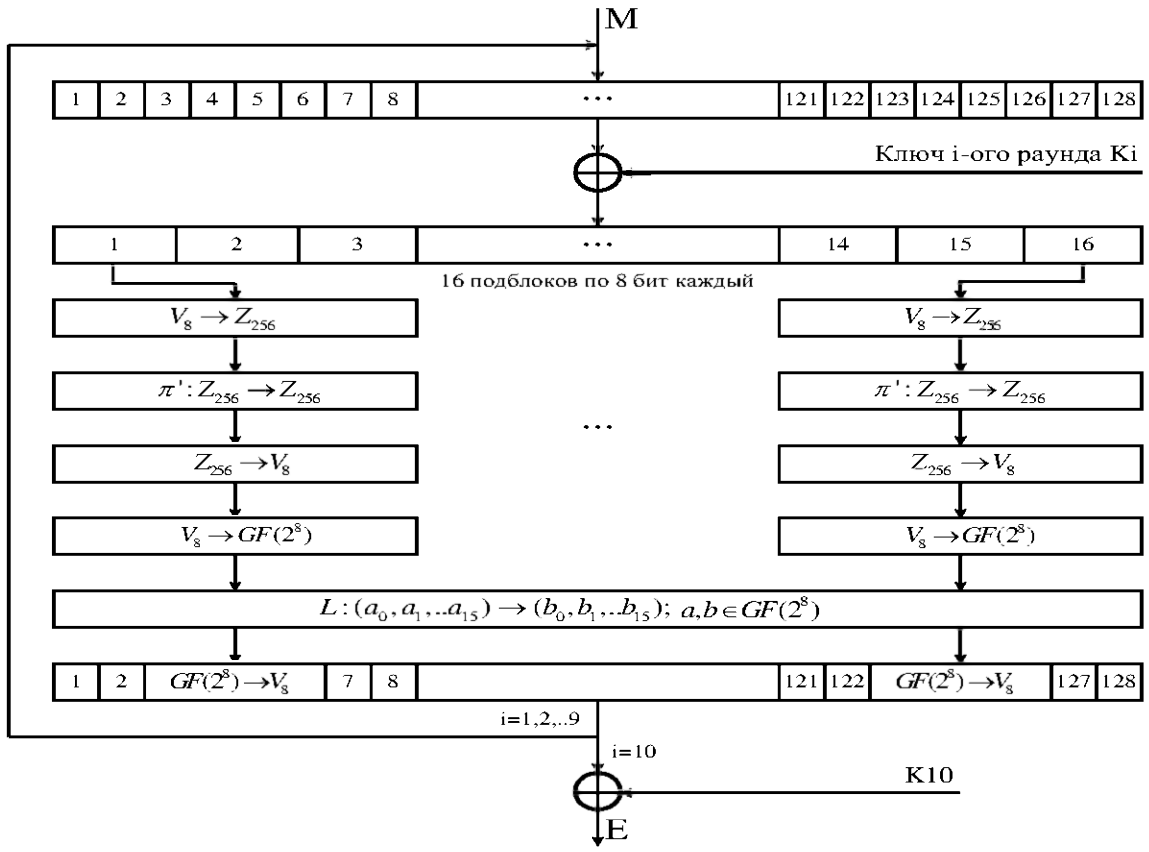
π'= (252, 238, 221, 17, 207, 110, 49, 22, 251, 196, 250, 218, 35, 197, 4, 77, 233,119, 240, 219, 147, 46, 153, 186, 23, 54, 241. 187, 20, 205, 95, 193, 249,

24, 101,90, 226, 92, 239, 33, 129, 28, 60, 66, 139, 1, 142, 79, 5, 132, 2, 174, 227, 106, 143,160, 6, 11, 237, 152, 127, 212, 211, 31, 235, 52, 44, 81, 234, 200, 72, 171, 242, 42,104, 162, 253, 58, 206, 204, 181, 112, 14, 86, 8, 12, 118, 18, 191, 114, 19, 71, 156, 183, 93, 135, 21, 161, 150, 41, 16, 123, 154, 199, 243, 145, 120, 111, 157, 158, 178,177, 50, 117, 25, 61, 255, 53, 138, 126, 109, 84, 198, 128, 195, 189, 13, 87, 223,245, 36, 169, 62, 168, 67, 201, 215, 121, 214, 246, 124, 34, 185, 3, 224, 15, 236,222, 122,148, 176, 188, 220, 232, 40, 80, 78, 51, 10, 74, 167, 151, 96, 115, 30, 0,98, 68, 26, 184, 56, 130, 100, 159, 38, 65, 173, 69, 70, 146, 39, 94, 85, 47, 140, 163,165, 125, 105, 213, 149, 59, 7, 88, 179, 64, 134, 172, 29, 247, 48, 55, 107, 228, 136,217, 231, 137, 225, 27, 131, 73, 76, 63, 248, 254, 141, 83, 170, 144, 202, 216, 133,97, 32, 113, 103, 164, 45, 43, 9, 91, 203, 155, 37, 208, 190, 229, 108, 82, 89, 166,116, 210, 230, 244, 180, 192, 209, 102, 175, 194, 57, 75, 99, 182).

3.3. Множество чисел Z_{256} вновь преобразуется в двоичную последовательность V_8 (Например, 133→10000101).

3.4. Двоичные последовательности V_8 рассматриваются как элементы поля $GF(2^8)$ (Например, 10000101→ x^7+x^2+1).

4. 16 подблоков объединяются в один блок, состоящий из шестнадцати 8-битных подблоков. Над ним производится линейное преобразование L, которое будет описано позднее.
5. Блок, состоящий из шестнадцати элементов конечного поля $GF(2^8)$, преобразуется в двоичную последовательность и отправляется в начало



алгоритма для всех раундов, кроме i=10.

6. Для 10-го раунда на выход поступает блок, полученный в пункте 5 после выполнения 9-го раунда, который складывается по mod2 с 10-ым раундовым ключом и образует криптограмму E для исходного сообщения M.

Схема раунда шифрования алгоритма ГОСТ Р 34.12-2015

Линейное преобразование L: $(a_0,a_1,..a_{15})\rightarrow(b_0,b_1,..b_{15})$ выполняется при помощи шестнадцати циклов линейного рекуррентного регистра (ЛРР) (см. следующий рисунок).

1 цикл:

$(a_0,a_1,\ldots,a_{15})\rightarrow(a_0^1,a_1^1,..,a_{15}^1)$, где $a_{14}^1 = a_{15}$, $a_{13}^1 = a_{14}$,..., $a_0^1 = a_1$, $a_{15}^1 = l(a_0,a_1,..a_{15})$.

2 цикл:

$(a_0^1,a_1^1,..,a_{15}^1)\rightarrow(a_0^2,a_1^2,\ldots,a_{15}^2)$, где $a_{14}^2=a_{15}^1$, $a_{13}^2=a_{14}^1$,..., $a_0^2=a_1^1$, $a_{15}^2=l(a_0^1,a_1^1,..a_{15}^1)$.

И так далее до 16 цикла, где:

$(a_0^{15},a_1^{15},\ldots,a_{15}^{15})\rightarrow(a_0^{16},a_1^{16},\ldots,a_{15}^{16})$, где $a_{14}^{16}=a_{15}^{15}$, $a_{13}^{16}=a_{14}^{15}$,...,$a_0^{16}=a_1^{15}$, $a_{15}^{16}=l(a_0^{15},a_1^{15},\ldots,a_{15}^{15})$.

Здесь $l(a_0^{i+1},a_1^{i+1},..a_{15}^{i+1}) = A_{15}a_{15}^i + A_{14}a_{14}^i + A_{13}a_{13}^i + A_{12}a_{12}^i + A_{11}a_{11}^i + A_{10}a_{10}^i + A_9a_9^i + A_8a_8^i + A_7a_7^i + A_6a_6^i + A_5a_5^i + A_4a_4^i + A_3a_3^i + A_2a_2^i + A_1a_1^i$.

Значения коэффициентов:

$A_7=A_9=A_0=1$, $A_1=A_{15}=148$, $A_2=A_{14}=32$, $A_3=A_{13}=133$, $A_4=A_{12}=16$, $A_5=A_{11}=194$, $A_6=A_{10}=192$, $A_8=251$. Все операции выполняются в поле $GF(2^8)$, $a^i \in GF(2^8)$, (см. раздел 3.1.9.)

Таким образом, видно, что начальное заполнение ЛРР $(a_0,a_1,..a_{15})$ подвергается поочерёдному сдвигу вправо тактовыми импульсами, а за счёт обратной связи, выполняемой преобразованием l(.), элемент ЛРР a_{15} замещается на значение l(.). После выполнения шестнадцати сдвигов, текущее состояние ЛРР, т.е. $a_0^{16},a_1^{16},..a_{15}^{16}$, считывается в однотактовый регистр как $b_0,b_1,..b_{15}$, что и даёт выход линейного преобразования.

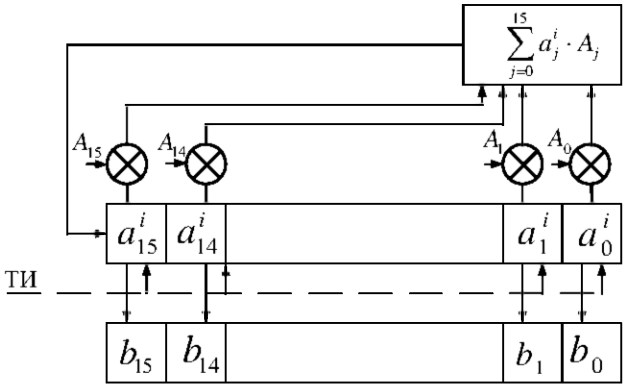


Схема линейного преобразования L алгоритма шифрования

Как видно из алгоритма шифрования, для выполнения каждого i-ого раунда требуется i-ый раундовый ключ K_i, i=1,2..10.

Алгоритм выработки десяти раундовых ключей представляет собой схему Фейстеля, где основной (базовый или мастер-ключ) задаётся как открытое сообщение $K=(K_1,K_2)$, где K_1 – левая половина и K_2 – правая половина основного ключа К, каждая из которых имеет длину 128 бит. В процессе преобразования в качестве ключей используются заранее выработанные итерационные константы C_i, i=1,2,..32.

Выработка констант описывается формулой:

$$C_j = L(V_{128}(j)),\ j=1,2,..32.$$

Данная запись означает, что сначала числа j=1,2,..32 переводятся в соответствующие двоичные последовательности длиной 128 бит. (Например: j=2, $V_{128}(2)$=000...10; j=5, $V_{128}(5)$=00...101.) Далее полученные двоичные последовательности преобразуются в константы C_j длиной 128 бит при помощи линейного преобразования L, описанного ранее и представленного на схеме, видно, что линейное преобразование не зависит от ключа.

Все раундовые ключи K_i (i=1,2..10) имеют длину 128 бит и формируются из первичного базового ключа длиной 256 бит.

Алгоритм выработки раундовых ключей описывается одной формулой:

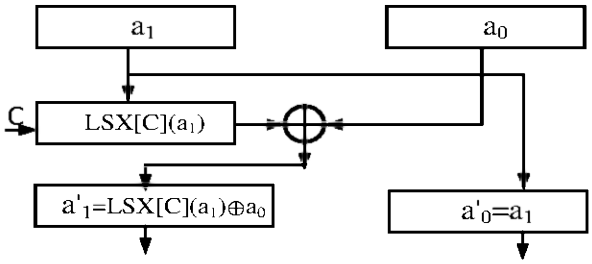
$$(K_{2i+1}, K_{2i+2})=F[C_{8(i-1)+8}]\ldots F[C_{8(i-1)+1}](K_{2i-1}, K_{2i}).$$

F[C](,) здесь означает один цикл схемы Фейстеля, а именно:

$$F[C](a_1, a_0)=(LSX[C](a_1) \oplus a_0, a_1).$$

Последнее выражение, в свою очередь, означает, что при заданном ключе С выполняется один раунд шифрования ГОСТ Р 34.12-2015 (см. схему одного раунда шифрования) от двоичного блока a_1 длиной 128 бит. (Напомним, что X означает сложение по mod2 с ключом С длиной 128 бит, S – нелинейное преобразование по таблице (см. п.3.2 алгоритма шифрования) и, наконец, L – линейное преобразование, рассмотренное выше).

Далее полученный блок криптограммы $LSX[C](a_1)$ длиной 128 бит складывается по mod2 с блоком a_0 и к полученной цепочке справа приписывается двоичная цепочка a_1 длиной 128 бит, что даёт новые половинки для схемы Фейстеля a_1' и a_0'.



Цикл в схеме Фейстеля формирования ключа

Данные преобразования представлены на следующем рисунке. В роли ключа С в этом преобразовании выступают константы длиной по 128 бит каждая, выработанные на подготовительном этапе (см. выше).

В качестве первого и второго раундовых ключей выбираются половинки $К_1$ и $К_2$ базового ключа. Начальными блоками (a_1, a_0) являются левая и правая половинки базового ключа, т.е. $a_1=К_1$, $a_0=К_2$, а ключом «С» - константа $С_1$.

Для выработки следующих раундовых ключей ($К_3$, $К_4$) преобразование F повторяется ещё 7 раз, где для каждого i-ого преобразования i=2,3..8 используется ключ в виде постоянной $С_i$, а в качестве входного сообщения (a_1', a_0') выход криптограммы предыдущего преобразования.

Для выработки следующей пары раундовых ключей ($К_5$, $К_6$) (каждый из которых имеет длину 128 бит), используется та же схема преобразования F и её повторение 8 раз, но в качестве исходного сообщения a_1, a_0 берутся предыдущие раундовые ключи $К_3$, $К_4$, а в качестве ключей шифрования используются константы $С_9$..$С_{16}$.

Данный метод распространяется для выработки всех раундовых ключей, вплоть до $К_9$, $К_{10}$.

Схема выработки пары раундовых ключей ($К_{2i+1}$, $К_{2i+2}$) показана на следующем рисунке, где $a_1(j, i)$ и $a_0(j, i)$ – означают левую и правую половинку выхода схемы Фейстеля на j-ом (j=1,2..8) раунде.

Таким образом, получаем, что при выполнении такого алгоритма для i=1,2,3,4 мы получаем 10 раундовых ключей $К_1$, $К_2$,..$К_{10}$ с использованием тридцати двух предварительно выработанных констант $С_1$,$С_2$,..$С_{32}$.

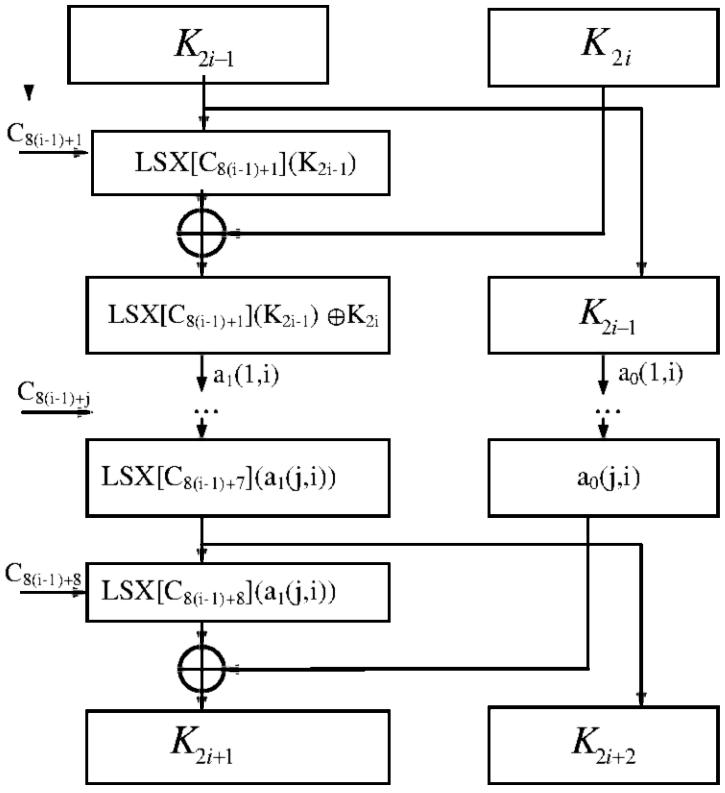


Схема выработки раундовых ключей алгоритма шифрования

Алгоритм дешифрования

Для дешифрования выполняются операции с обратным порядком выбора ключей: K_{10}, K_9,..K_1 и при выполнении обратных преобразований.

Легко видеть, что все преобразования, показанные на схеме одного раунда шифрования, имеют простые обратные. Действительно, сложение с i-ым раундовым ключом по mod2 имеет элементарную обратную операцию (повторное сложение с этим ключом), преобразование $V_8 \rightarrow Z_{256}$ имеет тривиальную обратную операцию $Z_{256} \rightarrow V_8$, табличное преобразование π': $Z_{256} \rightarrow Z_{256}$ имеет простую обратную операцию π'^{-1}: $Z_{256} \rightarrow Z_{256}$, когда находится число в таблице, и ему сопоставляется порядковый номер этого числа в таблице (пример: 221→3, 182→255), далее следует обратное преобразование $Z_{256} \rightarrow V_8$, $V_8 \rightarrow GF(2^8)$.

Единственным нетривиальным обратным преобразованием является $L^{-1}:(b_0,\ldots,b_{15}) \rightarrow (a_0,\ldots,a_{15})$. Рассмотрим его более подробно.

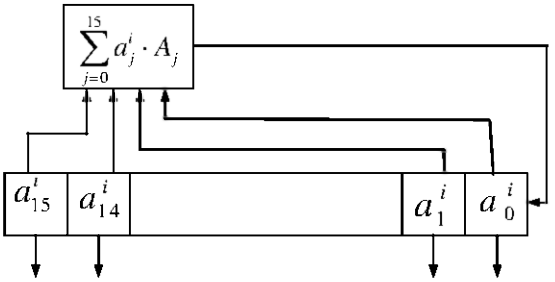
Как видно из схемы одного раунда шифрования, прямое преобразование $L^1:(a_0,\ldots,a_{15}) \rightarrow (b_0,\ldots,b_{15})$ заключается в выполнении шестнадцати сдвигов в $q=2^8$-ичном линейном рекуррентном регистре с обратной связью, выполняемой при помощи сумматора $a_0^{i+1} = \sum_{j=0}^{15} A_j a_j^i$, i=0,1,...,15, где A_j, $a_j^i \in GF(2^8)$.

Для того, чтобы выполнить обратное преобразование, т.е.

$$L^{-1}:(b_0,\ldots,b_{15}) \rightarrow (a_0^{16},..a_{15}^{16}),$$

необходимо научиться находить цепочку $(a_0^{i-1},a_1^{i-1},\ldots,a_{15}^{i-1})$ к цепочке $(a_0^i,a_1^i,\ldots,a_{15}^i)$.

Изобразим упрощённую схему линейного преобразования при дешифровании:



Упрощённая схема линейного преобразования l(a0,..a15)

Из этой схемы непосредственно видим, что $a_{15}^{i-1} = a_{14}^i$, …, $a_j^{i-1} = a_{j-1}^i,\ldots,a_1^{i-1} = a_0^i$. Остаётся найти только элемент a_0^{i-1}. Этот элемент может быть восстановлен из уравнения $a_{15}^i = \sum_{j=0}^{15} A_j a_j^{i-1}$

Действительно, из этого уравнения находим

$$a_0^{i-1} = \left(a_{15}^i - \sum_{j=1}^{15} A_j a_j^{i-1} \right) \cdot A_0^{-1} = \left(a_{15}^i - \sum_{j=1}^{15} A_j a_{j-1}^i \right) \cdot A_0^{-1} .$$

Поскольку в конечном поле GF(2^8) операция вычитания совпадает с операцией сложения, а коэффициент $A_0 = A_0^{-1} = 1$, то из последнего равенства получаем:

$$a_0^{i-1} = a_{15}^i + \sum_{j=1}^{15} A_j a_{j-1}^i .$$

В заключение отметим, что даже предварительное сравнение нового стандарта шифрования с ранее описанными в этой книге шифрами DES и ГОСТ-28147-89, показывает, что он является значительно более стойким, поскольку обладает значительно более длинным базовым ключом по сравнению с шифром DES и более сложными алгоритмами шифрования и выработки раундовых ключей по сравнению с шифром ГОСТ 28147-89.

Очевидным преимуществом нового стандарта является также задание для него (в отличие от старого стандарта) всех параметров (в том числе и нелинейных преобразований) в явном виде. Что же касается шифра AES, то в первом приближении можно считать, что он имеет примерно одинаковую стойкость с новым ГОСТ-ом, хотя линейное преобразование последнего больше соответствует оптимальному, построенному на базе укороченного кода Рида-Соломона, описанного в данной книге.

3.2. Способы построения и криптоанализ потоковых шифров, использующих линейные рекуррентные регистры сдвига

Один из способов построения потокового шифра, как моды блокового шифра с обратной связью по гамме (OFB), уже рассматривался ранее в разд. 3.1.12. В этом разделе мы будем изучать потоковый шифр, построенный на основе другой техники, а именно на основе использования так называемых *линейных рекуррентных регистров сдвига* (ЛРР), свойства которых будут представлены далее.

Напомним, что потоковым шифром называется шифр, основанный на следующих общих преобразованиях шифрования и дешифрования.

Шифрование:

$$E_i = M_i \oplus \gamma_i(k) \qquad i = 1, 2, ..., \tag{3.49}$$

где M_i – сообщение, представленное в виде двоичной последовательности; E_i – двоичная последовательность криптограммы; $\gamma_i(k)$ – двоичная последовательность *гаммы*, зависящая от ключа k.

Дешифрование:

$$M_i = E_i \oplus \gamma_i(k), \qquad i = 1, 2, ..., \tag{3.50}$$

где, как видно, используется тот же самый ключ и, следовательно, вырабатывается та же гамма, что и при шифровании.

Общие свойства потокового шифра:

1) ошибки, произошедшие в определенных позициях криптограммы, полностью сохраняются в тех же позициях после дешифрования:

$$\tilde{E} = E \oplus e \qquad \Rightarrow \qquad \tilde{M} = M \oplus e,$$

где e – образец ошибок (т. е. двоичная последовательность с единицами на позициях, где произошли ошибки, и с нулями, где ошибок не было);

2) для правильного дешифрования потокового шифра необходима синхронизация источников гаммы при шифровании и дешифровании с точностью до бита;

3) стойкость потокового шифра определяется стойкостью гаммы, которая может быть наглядно определена следующим образом: при известном алгоритме формирования гаммы, но при неизвестном ключе k знание достаточно большого отрезка этой гаммы не позволяет в обозримое время найти ее продолжение;

4) если на различных сеансах связи гамма повторяется, то это приводит к простому дешифрованию сообщения (см. аналогичное свойство OFB-моды в разд. 3.1.12).

Очевидно, что наилучшим способом построения датчика гаммы было бы использование датчика чисто случайной двоичной последовательности, скажем, на основе преобразования теплового шума или использование датчика на основе радиоактивного распада. Такой датчик гаммы привел бы к идеальному шифру. Однако, как было показано в разд. 2.1, это возможно только тогда, когда длина ключа пропорциональна длине сообщения. Для того чтобы построить датчик гаммы с «коротким ключом», необходимо использовать двоичную *псевдослучайную последовательность* (ПСП), которая по своим свойствам будет близка к чисто случайной последовательности.

Примером ПСП (ПСЧ) является последовательность чисел, вырабатываемых при помощи программы, используемой в компьютерах, но эта ПСЧ не является криптографически стойкой, т. е. не обеспечивает непредсказуемость ее продолжения по известному начальному отрезку. Поэтому рассмотрим другой метод формирования ПСП на основе использования ЛРР. Для этого изучим прежде всего основные свойства ЛРР.

3.2.1. Линейный рекуррентный регистр и его основные свойства

Линейный рекуррентный регистр (ЛРР) – это схема, или алгоритм, формирования двоичных ПСП, который задается следующим *рекуррентным уравнением*:

$$b_j = \begin{cases} a_j, & j = 0, 1, ..., n-1; \\ \sum_{i=0}^{n-1} h_i b_{i+j-n}, & j = n, n+1, ..., \end{cases} \tag{3.51}$$

где $a_0, a_1, ..., a_{n-1}$ – двоичное *начальное заполнение* ЛРР; b_j – бесконечная двоичная *выходная* последовательность ЛРР; h_i – двоичные *коэффициенты*, определяющие структуру ЛРР; n – *длина* ЛРР. (Заметим, что действия над коэффициентами и суммирование в (3.51) производятся в конечном поле $GF(2)$.)

ЛРР, заданный уравнением (3.51), является, точнее говоря, двоичным ЛРР, поскольку можно очевидным образом задать и q-ичный ЛРР, если операции в (3.51) будут производиться в поле $GF(q)$, однако для построения потокового шифра вполне достаточно использовать только двоичный ЛРР (и поэтому будем в дальнейшем опускать слово «двоичный»).

Для большей наглядности рассмотрим аппаратную реализацию ЛРР, которая показана на рис. 3.20.

Работа данной схемы достаточно очевидна: первоначально в ячейки памяти вводится двоичное начальное заполнение $a_0, a_1, ..., a_{n-1}$, затем под воздействием каждого из тактовых импульсов содержимое этих ячеек памяти сдвигается на одну ячейку вправо; одновременно с выходов ячеек их содержимое (0 или 1) поступает на входы сумматоров по mod 2 через «коммутатор обратных связей», который пропускает к сумматорам содержимое ячеек, если соответствующий им коэффициент $h_i = 1$ и не пропускает при $h_i = 0$; содержимое крайней правой ячейки одновременно поступает на выход схемы и на вход первого сумматора, а выход крайнего левого сумматора соединен со входом первой ячейки памяти. Таким образом, при ненулевом начальном заполнении на выходе схемы ЛРР будет появляться неограниченная по длине ненулевая двоичная последовательность, если число тактовых импульсов также не ограничено.

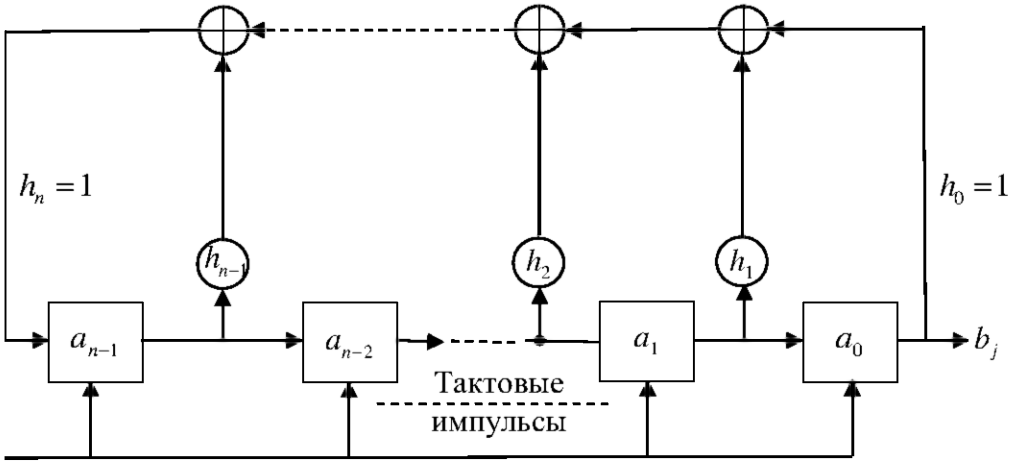


Рис. 3.20. Линейный рекуррентный регистр сдвига

Попытаемся использовать ЛРР как датчик гаммы, предполагая, что ключом является либо его начальное заполнение, либо отводы обратных

связей. Однако для доказательства того, что это возможно, необходимо изучить прежде свойства ЛРР, сравнив их со свойствами, которыми должны обладать ПСП. (Заметим, что для описания свойств ЛРР и способов построения потоковых шифров на основе ЛРР далее будут существенно использоваться элементы теории конечных полей, изложенные ранее в разд. 3.1.9.)

Для того чтобы было более удобно описывать свойства ЛРР, необходимо ввести одно новое понятие.

Определение 3.2.1. *Полиномом обратных связей* $h(x)$ двоичного ЛРР длины n называется полином следующего вида:

$$h(x) = h_n x^n + h_{n-1} x^{n-1} + \ldots + h_1 x + h_0,$$

где h_i – двоичный коэффициент, задающий ЛРР в (3.51), причем мы всегда полагаем $h_n = 1$, $h_0 = 1$.

Основные свойства ЛРР.

1. Период выходной последовательности ЛРР. Докажем сначала следующее неравенство для величины периода T любого ЛРР: $T \le 2^n - 1$, где n – длина ЛРР.

Доказательство. Действительно, общее число возможных состояний ячеек памяти ЛРР равно 2^n, но состояние из всех нулей появиться не может, и, следовательно, число реально появляющихся состояний равно $2^n - 1$. С другой стороны, каждое состояние ячеек памяти ЛРР однозначно определяет выходной символ ЛРР, и поэтому существование периода ЛРР больше чем $2^n - 1$ означало бы, что число различных ненулевых состояний ячеек памяти ЛРР было бы больше, чем $2^n - 1$, что невозможно.

Утверждение 3.2.1 [8]. Минимально возможный период ЛРР равен такому наименьшему целому числу n_0, для которого многочлен $x^{n_0} + 1$ делится на многочлен обратных связей ЛРР $h(x)$, причем этот период будет достигаться для начального заполнения, состоящего из коэффициентов многочлена $g(x) = (x^{n_0} + 1)/h(x)$ и следующих за ними $n_0 - n - 1$ нулей. (Для других начальных заполнений ЛРР период может оказаться меньше, чем n_0.) Если многочлен $h(x)$ является неприводимым над $GF(2)$, то при любом начальном заполнении ЛРР период выходной последовательности будет в точности равен n_0.

Из определения примитивного многочлена в разд. 3.1.9 следует, что если $h(x)$ – примитивный многочлен, то он должен удовлетворять следующим условиям:

$$\begin{cases} h(x) \mid 2^{2^n-1}+1; \\ h(x) \nmid x^{n'}+1, \ n' < 2^n-1. \end{cases} \tag{3.52}$$

Тогда из утверждения 3.2.1 получаем, что для ЛРР, *реализованного на примитивном полиноме*, период выходной последовательности при любом начальном заполнении будет в точности равен 2^n-1. Последовательности, генерируемые такими ЛРР, называют *последовательностями максимальной длины*.

Пример 3.2.1. Пусть ЛРР длины 4 имеет полином обратных связей $h(x)=x^4+x^3+x+1=(x+1)^2(x^2+x+1)$. Тогда из утверждения 3.2.1 получаем

$$h(x) \mid x^6+1 \Rightarrow T=6, g(x)=\frac{x^6+1}{h(x)}=x^2+x+1.$$

Поэтому для начального заполнения 0111 выходная последовательность оказывается равной 111000 и имеет период, равный 6. Если теперь выбрать $h(x)$ примитивным, например, как $h(x)=x^4+x+1$, то такой ЛРР будет иметь максимальный период выходной последовательности, равный 15, для любого начального заполнения.

Важно отметить, что выбор ЛРР на примитивном полиноме является необходимым условием при его использовании в качестве датчика в потоковом шифре, так как в противном случае гамма будет повторяться с меньшим периодом, и криптоаналитик, зная это, может дешифровать сообщение, используя свойство 4 потокового шифра.

Далее будем рассматривать только ЛРР, построенные на примитивных полиномах.

2. *Свойство «окна» для выходной последовательности ЛРР*. Пусть $\bar{b}$ – выходная последовательность ЛРР и пусть $\tilde{\bar{b}}$ подпоследовательность $\bar{b}$ длины $2^n+k-2,\ k \le n$. Тогда каждая ненулевая подпоследовательность $\tilde{\tilde{\bar{b}}}$ длины k будет присутствовать в точности 2^{n-k} раз в подпоследовательности $\tilde{\bar{b}}$ (рис. 3.21). Это свойство называется свойством «окна», поскольку подпоследовательности $\tilde{\tilde{\bar{b}}}$ получаются как бы при перемещении «окна» шириной k вдоль последовательности $\tilde{\bar{b}}$. Если $k=n$, то все ненулевые подпоследовательности $\tilde{\tilde{\bar{b}}}$ будут появляться в «окне» точно по одному разу.

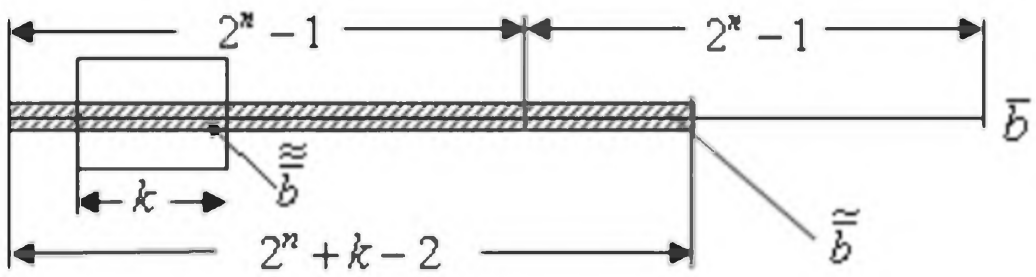


Рис. 3.21. Свойство «окна» для ЛРР

3. Свойство баланса для выходной последовательности ЛРР. Число нулей на периоде выходной последовательности ЛРР равно в точности $2^{n-1}-1$, а число единиц 2^{n-1}. Это свойство весьма просто доказывается. Действительно, если состояния ячеек памяти ЛРР представить целыми числами, то легко видеть, что нечетным числам будет соответствовать появление единиц в выходной последовательности, а четным – появление нулей, но поскольку число нечетных чисел на множестве 1, 2, …, 2^n-1 равно в точности 2^{n-1}, а число четных чисел $2^{n-1}-1$, то отсюда и следует доказательство данного утверждения. Это свойство называется свойством *баланса* поскольку при больших n оно означает, что число нулей на периоде выходной последовательности ЛРР будет практически равно числу единиц на этом периоде.

4. Свойство серий для выходной последовательности ЛРР. *Различают* серии блоков *длиной* k, *которые задаются следующим образом:* …011…10… – *и* серии пробелов *длиной* k*:* …$1\underbrace{00\ldots0}_{k}1$…

Свойство серий состоит в том, что половина всех серий на периоде выходной последовательности имеет длину 1, одна четверть серий – длину 2, $\frac{1}{8}$ серий – длину 3 и т. д., причем половина серий любой длины – это блоки, а половина серий любой длины – это пробелы [9].

5. Автокорреляционная функция выходной последовательности ЛРР. *Определим автокорреляционную функцию R(k) для* Т*-периодической выходной последовательности* b_i *ЛРР следующим образом:*

$$R(k)=\frac{A(k)-B(k)}{T}=\frac{2A(k)-T}{T},$$

где $A(k),B(k)$ – означают число совпадений и несовпадений соответственно между b_i и b_{i+k} на периоде T, причем сумма $i+k$ рассчитывается по $mod T$.

Тогда для *автокорреляционной функции* выходной последовательности ЛРР выполняется следующее соотношение [9]:

$$R(k)=\begin{cases} 1, & k=0; \\ -\dfrac{1}{2^n-1}, & 1\le k\le 2^n-1. \end{cases}$$

Из рассмотренных выше пяти свойств ЛРР следует, что ЛРР дает на выходе двоичную последовательность, которая удовлетворяет нескольким постулатам для ПСП, определенным С. Голомбом [9, 25]. Однако следующее свойство опровергает возможность использования ЛРР непосредственно в качестве датчика гаммы в потоковом шифре.

6. Предсказуемость выходной последовательности ЛРР. По известной последовательности, состоящей из $2n$ смежных выходных символов ЛРР $\underbrace{b_k, b_{k+1}, \ldots, b_{k+2n-1}}_{2n}$, можно однозначно определить его обратные связи $h_0 h_1 \ldots h_{n-1}$ и тогда предсказать все последующие элементы на выходе ЛРР, причем сложность решения такой задачи не превосходит $O\left(n^2\right)$, где n – длина ЛРР.

Действительно, используя рекуррентные уравнения (3.51) для нахождения выходных символов ЛРР $b_{k+n}, \ldots, b_{k+2n-1}$ через предыдущие символы, мы можем записать n уравнений и представить их в следующем матричном виде:

$$\begin{pmatrix} b_k & b_{k+1} & \ldots & b_{k+n-1} \\ b_{k+1} & b_{k+2} & \ldots & b_{k+n} \\ \ldots & \ldots & \ldots & \ldots \\ b_{k+n-1} & b_{k+n} & \ldots & b_{k+2n-2} \end{pmatrix} \cdot \begin{pmatrix} h_0 \\ h_1 \\ \vdots \\ h_{n-1} \end{pmatrix} = \begin{pmatrix} b_{k+n} \\ b_{k+n+1} \\ \vdots \\ b_{k+2n-1} \end{pmatrix}. \qquad (3.53)$$

Легко проверить, что строки матрицы, стоящей в левой части (3.53) линейно независимы, т. е. ее определитель не равен нулю. Тогда существует единственное решение этого уравнения относительно коэффициентов обратной связи $h_0 h_1 \ldots h_{n-1}$ со сложностью решения $O\left(n^3\right)$ [9]. Правда, для составления уравнений необходимо знать длину ЛРР n, но это можно сделать подбором. Более того, существует значительно более быстрый алгоритм решения этой задачи, известный как *алгоритм Берлекэмпа–Месси* (БМ) [9]. Данный алгоритм по не менее чем $2n$ известным последовательным выходным элементам ЛРР (и даже при неизвестной длине ЛРР) вычисляет коэффициенты обратной связи со сложностью $O\left(n^2\right)$. Это означает, что если ЛРР используется в качестве датчика гаммы, то, зная $2n$ смежных элементов сообщения и криптограммы, можно найти $2n$ элементов гаммы и вычислить ее продолжение на любую длину.

Таким образом, хотя мы и пришли к отрицательному выводу, состоящему в том, что использование только ЛРР в качестве датчика шифрующей гаммы оказывается нецелесообразным, однако было бы весьма желательно (учитывая положительные свойства ЛРР 1–5) сохранить его в качестве датчика первичной гаммы, которая затем преобразуется при помощи нелинейных операций. Такие преобразователи называются *нелинейными узлами усложнения* (НУУ). Тогда потоковый шифр приобретает вид, показанный на рис. 3.22.

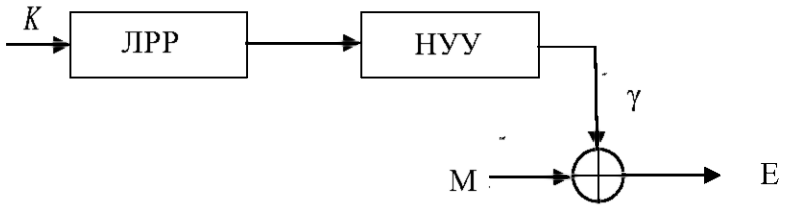


Рис. 3.22. Комбинация ЛРР и НУУ в потоковом шифре

В этом случае ключ можно вводить как начальное состояние ЛРР или/и как коэффициенты полинома обратных связей. (В обоих случаях длина ключа будет равна *n*.) Как было отмечено ранее, в качестве полиномов обратных связей необходимо всегда выбирать примитивные полиномы. При выборе ключа в виде начального заполнения этот полином является открытым, и поэтому он может быть выбран заранее с использованием методов, рассмотренных в разд. 3.1.9 (или просто взят из таблиц [8, 9]). В этом случае полное число нетривиальных ключей будет, очевидно, равно $2^n - 1$ и при выборе $n \geq 128$ криптоанализ путем их полного перебора окажется невозможным. Если же ключом является полином обратных связей, то эти полиномы также должны быть примитивными, но вдобавок и секретными, причем полное число таких ключей $N_2(n)$ для ЛРР длины *n* оказывается меньше, чем $2^n - 1$, и может быть найдено из соотношения, приведенного в разд. 3.1.9, которое для двоичного ЛРР принимает следующий вид:

$$N_2(n) = \varphi(2^n - 1)/n, \qquad (3.54)$$

где φ – функция Эйлера. Если *n* оказывается числом Мерсенна (т. е. когда $2^n - 1$ – простое число), то (3.54) преобразуется к виду

$$N_2(n) = (2^n - 2)/n. \qquad (3.55)$$

Из (3.54) и (3.55) видно, что хотя число примитивных полиномов степени *n* и меньше, чем 2^n, но оно также весьма велико, и при больших *n* число ключей будет непереборным. Поэтому для генерировании ключей в этом случае используется следующий алгоритм:

1. Физический генератор формирует чисто случайную двоичную последовательность длины *n*.

2. Полученная последовательность преобразуется в полином степени n, который тестируется на примитивность, и если тест оказывается положительным, то полином выбирается в качестве ключа.

3. Если тест на примитивность не проходит, то повторяются шаги 1, 2 и т. д., вплоть до получения положительного результата. (Поскольку число примитивных полиномов достаточно велико, то количество неудачных попыток не должно оказаться слишком большим.)

Как показано в разд. 3.1.9, тестирование полиномов на примитивность оказывается особенно простым, когда длина ЛРР является числом Мерсенна.

3.2.2. Нелинейные узлы усложнения, используемые для построения потоковых шифров

Узел перемножения. Это простейшее нелинейное преобразование вида $y = x_1 \times x_2$, где x_1, x_2, y – двоичные последовательности, а $\times$ – операция умножения в $GF(2)$, что иллюстрируется схемой, показанной на рис. 3.23.

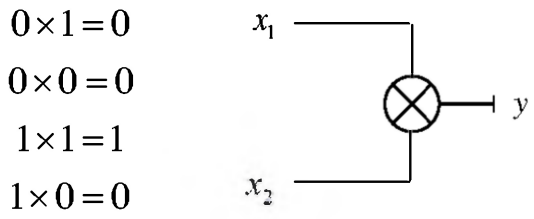


Рис. 3.23. Узел перемножения

В отличие от примеров в радиотехнике, которые показывают, что нелинейные устройства принципиально не эквивалентны линейным, в задачах преобразования двоичных последовательностей любые нелинейные преобразования выходов ЛРР, имеющего длину n, могут быть заменены одним ЛРР длины $n_{эк}$, возможно, большей, чем n, т. е. когда выполнено условие $n << n_{эк}$ (рис. 3.24).

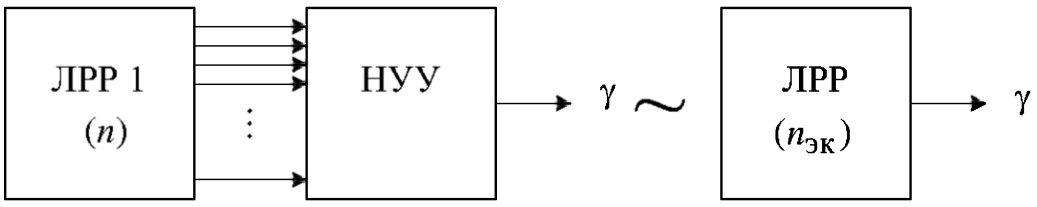


Рис. 3.24. Схема формирования гаммы на основе эквивалентного ЛРР

Очевидно, что если длина $n_{эк}$ *эквивалентного* ЛРР оказывается весьма большой, то $2n_{эк}$ элементов гаммы не могут быть перехвачены в обозримое время, а число операций, необходимых для нахождения структуры эквива-

лентного ЛРР по алгоритму БМ, равное $n_{эк}^{\ 2}$, также будет необозримо велико.

Определение 3.2.2. Пусть задана двоичная последовательность $\bar{b}$ длины N. Тогда *линейной эквивалентной сложностью* $L(\bar{b})$ (ЛЭС) этой последовательности называется минимальная длина такого ЛРР, который при некотором начальном заполнении и некотором выборе обратных связей может сгенерировать в точности ту же самую последовательность $\bar{b}$. Очевидно, что всегда выполняются неравенства: $0 < L(\bar{b}) \le N$. Наибольший интерес представляет случай, когда $L(\bar{b}) << N$. В частном случае, когда $\bar{b}$ представляет собой один период выходной последовательности ЛРР длины n, $L(\bar{b}) = n \approx \log_2(2^n - 1)$.

Для нахождения линейной эквивалентной сложности можно использовать алгоритм Берлекэмпа–Месси. Хотя большая эквивалентная сложность шифрующей гаммы и является необходимым условием обеспечения высокой стойкости потокового шифра, это условие оказывается далеко не достаточным для этого. Действительно, пусть, например, двоичная последовательность имеет произвольно большую длину N и следующий вид: 000000…..00001. Тогда ее эквивалентная линейная сложность, очевидно, равна N, поскольку любой ЛРР меньшей длины не сможет сгенерировать такую последовательность. С другой стороны, гамма в виде такой последовательности, конечно, не способна обеспечить стойкость потокового шифра.

Определение 3.2.3. Пусть $\bar{b}$ – двоичная последовательность длины N, т. е. $\bar{b} = b_0 b_1 \ldots b_{N-1}$ и L_k – линейная эквивалентная сложность ее подпоследовательности $b_0 b_1 \ldots b_k$, $k < N$. Тогда последовательность чисел $L_1, L_2, \ldots$ называется *линейным эквивалентным профилем (ЛЭП)* этой последовательности $\bar{b}$. Линейный эквивалентный профиль может также быть рассчитан с использованием БМ алгоритма и называется *совершенным*, если он близок к математическому ожиданию ЛЭП для чисто случайной последовательности. (В этом случае ЛЭП колеблется вокруг прямой линии $k/2$ [9].)

Расчет линейной эквивалентной сложности последовательности для узла перемножения после ЛРР (рис. 3.25). Для решения этой задачи необходимо перейти от рекуррентного задания выходной последовательности ЛРР (3.51) к ее явному виду, для чего потребуется существенным образом использовать математический аппарат теории конечных полей. Напомним, что рекуррентное задание ЛРР имело вид

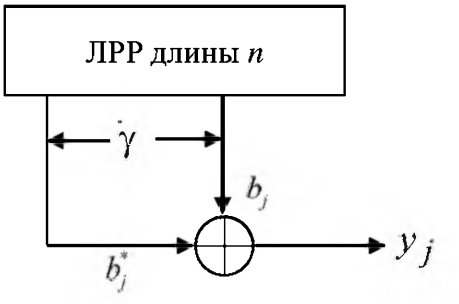


Рис. 3.25. Схема формирователя гаммы с перемножением двух выходных последовательностей ЛРР со сдвигом

$$b_j = \sum_{i=0}^{n-1} h_i b_{i+j-n},$$

где сумма вычислялась по модулю 2 или, иными словами, рассчитывалась в поле $GF(2)$. В комбинаторике [20] доказывается, что всякое рекуррентное уравнение допускает *явное решение*, если известно *характеристическое уравнение*, описывающее эту рекурренту. Применительно к нашему случаю это уравнение имеет вид $h(x)=0$, где $h(x)$ – полином обратных связей ЛРР. Так как этот полином по нашему условию примитивен, то его корни будут примитивными элементами конечного поля $GF(2^n)$, все они однократные, и их число равно n. Тогда явное выражение для рекуррентной последовательности принимает вид [20]

$$b_j = \sum_{k=1}^{n} A_k \alpha_k^j, \tag{3.56}$$

где коэффициенты $A_k \in GF\left(2^n\right)$ и определяются начальным заполнением ЛРР; α_k – k-й корень уравнения $h(\alpha_k)=0, \alpha_k \in GF\left(2^n\right)$, $j=0,1,2,\ldots$; n – длина ЛРР.

Поскольку из теории конечных полей известно, что все корни неприводимого полинома имеют представление α^{2^k}, $k=0,1,...,n-1$, где α – любой примитивный элемент поля $GF(2^n)$, то из (3.56) получаем

$$b_j = \sum_{k=1}^{n} A_k \left(\alpha^{2^{k-1}}\right)^j, \; h(\alpha)=0, \; A_k \in GF\left(2^n\right). \tag{3.57}$$

Пример. Рассмотрим ЛРР длины 3 на примитивном полиноме $h(x)=x^3+x+1$. Тогда $\alpha \in GF\left(2^3\right)$, причем другие корни $h(x)$ будут: α^2, $\alpha^4=\alpha^2+\alpha$ (разд. 3.1.9, табл. 3.1). Подставляя эти корни в (3.57), получаем

$$b_j = A_1\alpha^j + A_2\alpha^{2j} + A_3\left(\alpha^2+\alpha\right)^j. \tag{3.58}$$

Допустим, что начальное состояние ЛРР имеет вид: $a_0=1$, $a_1=0$, $a_2=0$. Тогда из (3.58) получаем систему уравнений для нахождения A_1, A_2, A_3:

$$1 = A_1 + A_2 + A_3;$$
$$0 = A_1\alpha + A_2\alpha^2 + \left(\alpha^2+\alpha\right)\cdot A_3;$$
$$0 = A_1\alpha^2 + A_2\left(\alpha^2+\alpha\right) + A_3\left(\alpha^2+\alpha\right)^2.$$

Легко показать, что решение этой системы будет следующим: $A_1 = A_2 = A_3 = 1$. Подставляя эти значения в (3.58), получаем

$$b_j = \alpha^j + \alpha^{2j} + \left(\alpha^2 + \alpha\right)^j, \ j = 0, 1, 2, \ldots$$

Хотя для данного начального заполнения все коэффициенты получились из конечного поля $GF(2)$, но не всегда будет так. Действительно, легко проверить, что для другого начального заполнения, например для $a_0 = 0$, $a_1 = 1$, $a_2 = 0$, решение системы даст коэффициенты уже из поля $GF(2^3)$: $A_1 = \alpha^2, A_2 = \alpha^2 + \alpha, A_3 = \alpha$.

Вернемся теперь к схеме перемножения последовательностей, взятых от двух отводов ЛРР: b_j и b_j^* (рис. 3.25). Используя их явные представления по формуле (3.57), получаем:

$$b_j = \sum_{k=1}^{n} A_k \left(\alpha^{2^{k-1}}\right)^j = \sum_{k=0}^{n-1} \tilde{A}_k \left(\alpha^{2^k}\right)^j; \qquad (3.59)$$

$$b_j^* = \sum_{k'=1}^{n^*} A_{k'}^* \left(\alpha^{2^{k'}}\right)^j = \sum_{k=1}^{n-1} \tilde{A}_{k'}^* \left(\alpha^{2^k}\right)^j, \ h(\alpha) = 0, \qquad (3.60)$$

$A_k, A_k^* \in GF\left(2^n\right)$. (Далее для простоты уберем знаки «~» над коэффициентами.) Подставим (3.59), (3.60) в выражение для произведения:

$$y_j = b_j \cdot b_j^* = \sum_{k=0}^{n} \sum_{k'=0}^{n} A_k A_{k'}^* \left(\alpha^{2^k + 2^{k'}}\right)^j. \qquad (3.61)$$

Поскольку α – примитивный элемент из поля $GF\left(2^n\right)$, то все α^s, $2 \le s \le 2^n - 1$, будут различными элементами поля.

Рассмотрим сначала случай, когда $(k, k') < n - 1$. Тогда $2^k + 2^{k'} < 2^{n-1} + 2^{n-1} = 2^n$ и, следовательно, все слагаемые $\alpha^{2^k + 2^{k'}}$ в (3.61) для этого случая будут различными. Пусть теперь $(k, k') = n - 1$, тогда $2^k + 2^{k'} = 2^n \Rightarrow \alpha^{2^n} = \alpha$. Ни для каких других пар k, k' мы не получим такого же результата, и поэтому общее число различных слагаемых в (3.61) будет равно $\frac{n \cdot (n-1)}{2} + n = \frac{n(n+1)}{2}$.

Есть опасение, однако, что некоторые коэффициенты при степенях в (3.61) будут равны 0. Это означает, что тогда будут выполняться следующие равенства: $A_k \cdot A_{k'}^* + A_{k'} \cdot A_k^* = 0$ для $k \ne k'$ или $A_{k'} \cdot A_k^* = 0$. Докажем, что такие равенства в действительности невозможны. Используя тот факт, что b_j^* – это задержанная на δ тактов последовательность b_j, получаем

$$b_j^* = \sum_{k=0}^{n-1} A_k \left(\alpha^{2^k}\right)^{j+\delta} = \sum_{k=0}^{n-1} A_k \left(\alpha^{2^k}\right)^{\delta} \cdot \left(\alpha^{2^k}\right)^{j}. \qquad (3.62)$$

Приравнивая (3.60) и (3.62), приходим к соотношению

$$A_k^* = A_k \left(\alpha^{2^k}\right)^{\delta}. \qquad (3.63)$$

Если $k = k'$, то из (3.63) получаем $A_k \cdot A_{k'}^* = A_k \cdot A_k^* = A_k^2 \left(\alpha^{2k}\right)^{\delta} \neq 0$, поскольку в противном случае здесь должно было бы выполняться условие $A_k = 0$. Однако тогда, как можно видеть из представления (3.56), ЛРР имел бы длину меньше, чем *n*, что противоречит начальному условию.

Пусть теперь $k \neq k'$. Тогда:

$$A_k \cdot A_{k'}^* = A_k \cdot A_{k'} \left(\alpha^{2^{'k}}\right)^{\delta};$$

$$A_{k'} \cdot A_k^* = A_{k'} \cdot A_k \left(\alpha^{2^k}\right)^{\delta};$$

$$A_k \cdot A_{k'}^* + A_{k'} \cdot A_k^* = A_k \cdot A_{k'} \left(\alpha^{2^{k'}}\right)^{\delta} + A_{k'} \cdot A_k \left(\alpha^{2k}\right)^{\delta}.$$

Из последнего равенства следует, что так как все действия выполняются в поле $GF\left(2^n\right)$, а его характеристика равна 2, то сумма любых двух элементов этого поля будет равна 0 тогда, и только тогда, когда эти элементы равны друг другу, т. е. когда выполняется соотношение

$$A_k \cdot A_{k'} \left(\alpha^{2^{k'}}\right)^{\delta} = A_{k'} \cdot A_k \left(\alpha^{2^k}\right)^{\delta},$$

которое, очевидно, эквивалентно равенству $\alpha^{2^{k'}} = \alpha^{2^k}$. Однако последнее равенство при $k \neq k'$ невозможно, поскольку α – примитивный элемент поля $GF\left(2^n\right)$.

Таким образом, мы доказали, что все коэффициенты при степенях в (3.61) являются ненулевыми, и тогда (3.61) принимает окончательно следующий вид:

$$y_j = \sum_{k=1}^{\frac{n}{2}\cdot(n+1)} \tilde{A}_k \left(\tilde{\alpha}_k\right)^j, \qquad (3.64)$$

где $\tilde{A}_k, \tilde{\alpha}_k \in GF\left(2^n\right)$. Соотношение (3.64) в действительности означает, что y_j – это выходная последовательность некоторого ЛРР длиной $n_{\text{эк}} = \dfrac{n \cdot (n+1)}{2}$, реализованного на полиноме обратных связей

$$h_{\text{эк}}(x) = \prod_{k=1}^{\frac{n\cdot(n+1)}{2}} (x + \tilde{\alpha}_k).$$

Отсюда и из определения ЛЭС следует, что для последовательности y_j, полученной как произведение последовательностей, взятых из двух отводов одного и того же ЛРР длиной n при любом начальном заполнении этого ЛРР, линейная эквивалентная сложность будет определена выражением

$$L(\bar{y}) = \frac{n\cdot(n+1)}{2}. \qquad (3.65)$$

Расчет линейной эквивалентной сложности последовательности, полученной перемножением более чем двух выходных последовательностей, взятых из отводов ЛРР. Ранее мы вывели формулу (3.65) для расчета ЛЭС для случая, когда последовательность получается как произведение последовательностей на выходах двух произвольно выбранных отводов ЛРР. Из этой формулы видно, что ЛЭС значительно возрастает по сравнению с длиной самого ЛРР. Так, например, если эта длина была равна 128, то ЛЭС оказывается равной 8256. Однако этого явно недостаточно для получения высокой стойкости потокового шифра. Перспективной представляется, на первый взгляд, идея сформировать гамму как произведение m последовательностей, полученных на выходах $m > 2$ отводов ЛРР, надеясь на значительное увеличение ЛЭС по сравнению со случаем $m = 2$. Поэтому рассмотрим схему, представленную на рис. 3.26, где δ_k означает расстояние (в числе ячеек памяти) между k-м и k+1-м отводами ЛРР длины n.

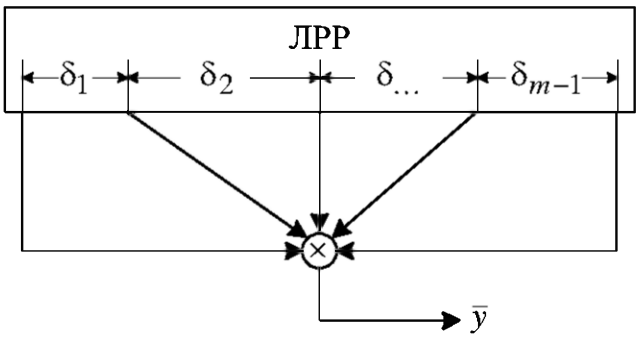


Рис. 3.26. Схема формирования гамм с перемножением $m > 2$ выходных последовательностей ЛРР

Используя технику, аналогичную той, которую мы применяли для случая $m = 2$, можно доказать [9], что ЛЭС $L(\bar{y})$ выходной последовательности $\bar{y}$ имеет следующую *верхнюю границу*:

$$L(\bar{y}) \le \sum_{k=1}^{m} C_n^k ,$$

где $C_n^k = n!/k!(n-k)!$ – число сочетаний из n элементов по k. Если все расстояния между отводами одинаковы, т. е. $\delta_1 = \delta_2 = \ldots = \delta_{m-1} = \delta$ и выполняется дополнительное условие $\gcd(2^n - 1, \delta) = 1$, то линейная эквивалентная сложность имеет также следующую *нижнюю границу* [9]:

$$L(\bar{y}) \ge C_n^m . \qquad (3.66)$$

(Заметим, что для разработчика потокового шифра нижняя граница для ЛЭС более важна, чем верхняя, поскольку именно она может гарантировать определенную сложность для предсказания гаммы и, следовательно, высокую стойкость потокового шифра.)

Однако использование в качестве нелинейного преобразования простого произведения последовательностей имеет существенный недостаток, поскольку в последовательности-произведении нарушается баланс нулей и единиц. Действительно, пусть $y = x_1 \cdot x_2$. Предположим, что заданы вероятности появления единиц и нулей во входных последовательностях:

$$P(x_1 = 1) = p_1; \qquad P(x_1 = 0) = 1 - p_1;$$
$$P(x_2 = 1) = p_2; \qquad P(x_2 = 0) = 1 - p_2.$$

Легко видеть, что: $P(y = 1) = p_1 \cdot p_2$; $P(y = 0) = 1 - p_1 \cdot p_2$. Тогда, даже для идеального баланса на входе, когда $p_1 = p_2 = 0{,}5$, получаем $P(y = 1) = 0{,}25$ и $P(y = 0) = 0{,}75$, т. е. баланс на выходе существенно ухудшается. Для m-кратного перемножения (рис. 3.26) получаем $P(y = 1) = p_1 p_2 \ldots p_m$, и поэтому даже при идеальном балансе на входе (на выходе) имеем $P(y = 1) = 2^{-m}$, что совершенно недопустимо и означает использование гаммы, состоящей из большого числа нулей, что приводит фактически к отсутствию всякого шифрования.

Расчет ЛЭС при перемножении выходов нескольких ЛРР. Для того чтобы увеличить ЛЭС, не нарушая баланса шифрующей гаммы, необходимо использовать другие типы нелинейных элементов, одним из которых является *генератор Джеффа*, приведенный на рис. 3.27, где $\otimes$ означает перемножитель, $\oplus$ – сумматор (в поле $GF(2)$), а $\circ$ – операцию инверсии двоичной последовательности. На вход такой схемы могут поступать выходные последовательности от нескольких раз-

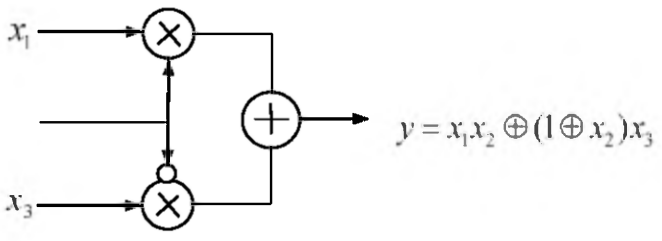


Рис. 3.27. Генератор Джеффа

личных ЛРР. Легко проверить, что такая схема фактически сохраняет баланс 0 и 1, если он был уже обеспечен на входе.

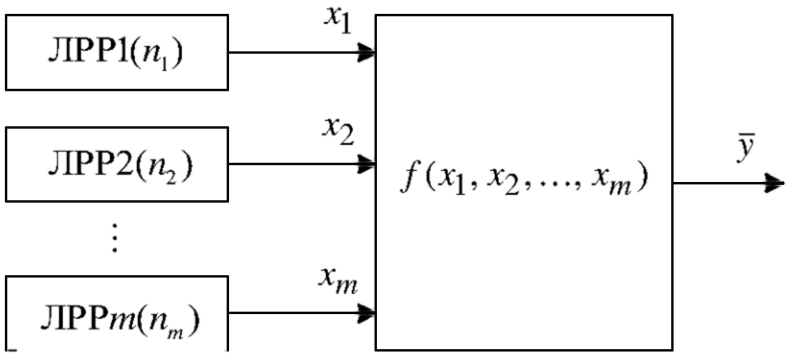


Рис. 3.28. Схема формирования гаммы с использованием нескольких ЛРР

Пусть в нашем распоряжении имеется m ЛРР. Тогда можно рассмотреть метод формирования гаммы, показанный на рис. 3.28, где $f(x_1, x_2, \ldots, x_m)$ – произвольная булева функция, а $n_1, n_2, ..., n_m$ – длины соответствующих ЛРР.

Утверждение 3.2.2 [9]. Если длины ЛРР $n_1, n_2, ..., n_m$ все различны и больше 2, а булева функция $f(x_1 ... x_m)$ представлена в алгебраически нормальной форме (АНФ) (разд. 3.1.8), то ЛЭС может быть вычислена после подстановки в АНФ длин соответствующих ЛРР, т. е.:

$$L(\bar{y}) = f(n_1, n_2, ..., n_m). \qquad (3.67)$$

(Заметим, что вычисления в (3.67) должны производиться, как в обычной арифметике, а не как в поле $GF(2)$.)

Пример. Рассмотрим случай, когда имеется три ЛРР с длинами n_1, n_2, n_3, а в качестве булевой функции используется генератор Джеффа. Представим булеву функцию, описывающую этот генератор (рис. 3.27), в АНФ: $y = x_1 x_2 \oplus x_2 x_3 \oplus x_3$. Используя (3.67), получаем

$$L(y) = n_1 n_2 + n_2 n_3 + n_3. \qquad (3.68)$$

В [9] доказано, что если длины ЛРР n_1, n_2, n_3, входящих в генератор Джеффа, являются взаимно простыми числами, то период его выходной последовательности будет равен $(2^{n_1} - 1)(2^{n_2} - 1)(2^{n_3} - 1)$. Дальнейшего увеличения ЛЭС гаммы (без ухудшения баланса) можно добиться, например, *каскадированием* подобных генераторов.

3.2.3. Построение датчика шифрующей гаммы на основе использования ЛРР с управляемым тактированием

В [9,21] предлагается много различных типов датчиков гаммы, построенных по этому принципу. Рассмотрим наиболее перспективный для практического применения генератор с так называемым *поочередным тактированием*. В этой схеме (рис. 3.29) ЛРР-1 тактируется регулярно импульсами ТИ, а ЛРР-2 и ЛРР-3 тактируются нерегулярно, в зависимости от наличия или отсутствия импульсов ТИ-1, ТИ-2 соответственно, вырабатываемых псевдослучайно при помощи выходной последовательности ЛРР-1. Обозначения на этой схеме такие же, как на рис.3.28, а ключи k_1, k_2, k_3 вводятся как начальные состояния ЛРР-1, ЛРР-2, ЛРР-3, имеющих длины n_1, n_2, n_3 соответственно. Если выход ЛРР-1 равен 1, то ЛРР-2 продвигается, а ЛРР-3 не продвигается, и на его выходе повторяется предыдущий символ. Если выход ЛРР-1 равен 0, то ЛРР-3 продвигается, а ЛРР-2 не продвигается, и на его выходе повторяется предыдущий символ.

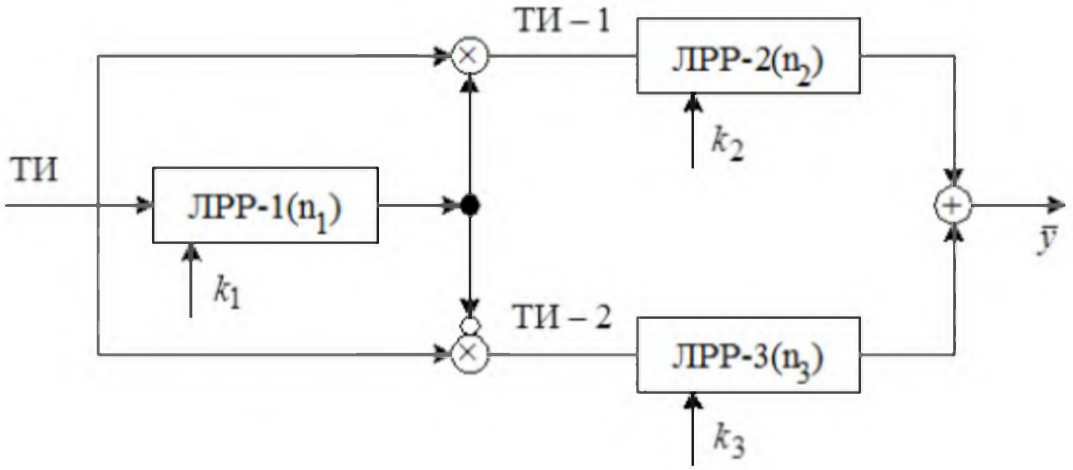


Рис. 3.29. Датчик гаммы, использующий ЛРР с управляемым тактированием

Доказывается [9], что период T выходной последовательности $\bar{y}$ данной схемы при условии, что вместо ЛРР-1 используется так называемый генератор *последовательности ДеБрейна*, которая получается из ЛРР-1, если в конце каждой генерируемой подпоследовательности, состоящей из $n_1 - 1$ смежных нулей, добавляется один 0, будет равен $T = 2^{n_1}(2^{n_2} - 1)(2^{n_3} - 1)$, если, кроме того, $\gcd(n_2, n_3) = 1$. Одновременно ЛЭС $L(y)$ выходной последовательности $\bar{y}$ будет удовлетворять следующему неравенству:

$$(n_2 + n_3) \cdot 2^{n_1 - 1} \le L(\bar{y}) \le (n_2 + n_3) \cdot 2^{n_1}. \qquad (3.69)$$

При практической реализации данной схемы целесообразно выбирать: $n_1 \approx n_2 \approx n_3 \approx 128$ и $\gcd(n_1, n_2) = 1$, $\gcd(n_2, n_3) = 1$. Тогда генератор гаммы,

построенный по рис.3.29, оказывается устойчивым к любым известным пока атакам на потоковые шифры [9].

3.2.4. Основные способы криптоанализа потоковых шифров

Известно множество атак на потоковые шифры, основными из которых являются следующие:

- тотальный перебор ключей;
-использование алгоритма Берлекэмпа–Месси (БМ);
-корреляционные атаки;
-быстрые корреляционные атаки;
-побочные атаки.

Рассмотрим каждую из этих атак более подробно.

Тотальный перебор ключей. Ключом в потоковых шифрах чаще всего является начальное заполнение ЛРР. Если длина ЛРР равна n, то общее число возможных ключей, очевидно, равно $2^n - 1$. Для исключения подобной атаки это число должно быть непереборно велико.Если ключом в шифре служат отводы обратных связей ЛРР (или, что то же самое, коэффициенты полинома обратных связей), то общее число ключей будет равно числу примитивных полиномов степени n, где n – длина ЛРР. Как отмечено в разд. 3.2.1 (формула (3.54)), это число равно $\varphi(2^n - 1)/n$, где $\varphi(\cdot)$ – функция Эйлера. (Если n – число Мерсенна, то по формуле (3.55) число таких полиномов, а следовательно, и число ключей, будет равно $(2^n - 2)/n.$) Тогда для исключения атаки перебором ключей данные величины должны быть непереборно большими.

В потоковых шифрах можно использовать в качестве ключа одновременно начальное заполнение и отводы ЛРР. Таким образом,количество возможных ключей возрастает очевидным образом.

Атака на основе использования алгоритма БМ. Сущность атаки заключается в том, что, даже не зная структуры шифра, но зная, что он основан на использовании одного или нескольких ЛРР и что линейная эквивалентная сложность шифрующей гаммы является обозримой величиной, находится отрезок шифрующей гаммы $\gamma = M \oplus E$ достаточно большой длины N (для чего должно быть известно N бит сообщенияMи соответствующих им бит криптограммы E). Далее к этой гамме применяется алгоритм БМ, что позволяет найти длину эквивалентного ЛРР, а также его отводы и начальное заполнение и, следовательно, вычислить произвольное продолжение этой гаммы. Знание продолжения гаммы позволяет, в свою очередь, дешифровать сообщение M как $M = E \oplus \gamma$ за пределами ее известных пока N бит. Сложность выполнения БМ составляет примерно $O(N^2)$ операций.

Для предотвращения этой атаки необходимо использовать потоковые шифры с НУУ, которые обеспечивают такую линейную эквивалентную сложность L гаммы, что нахождение $N = 2L$ элементов этой гаммы или выполнение $O(L^2)$ операций оказывается невозможным. Примером такого шифра является генератор гаммы с управляемым тактированием, показанный на рис.3.29, при выборе его параметров $n_1 \approx n_2 \approx n_3 \approx 128$.

Корреляционные атаки. Предположим, что датчик шифрующей гаммы построен по схеме, показанной на рис.3.28. Зная вид булевой функции $f(x_1, ..., x_m)$, обеспечивающей нелинейное преобразование выходов ЛРР, рассчитаем *теоретическую корреляцию* между двоичным выходом x_{ik} каждого из ЛРР $(i = 1, 2, 3)$ и двоичным элементом гаммы y_k.(Напомним, что корреляция двух двоичных элементов*x* и*y* определяется как следующая величина: $R(x, y) = \Pr(x = y) - \Pr(x \neq y)$, где $\Pr(A)$ означает вероятность события*A*.) Если корреляция окажется значимо отличающейся от нуля, то это создает базу для так называемой корреляционной атаки.Рассмотрим, для примера, генератор Джеффа, показанный на рис.3.30, и рассчитаем одну из корреляций:

$$P(x_{1k} = y_k) = P(x_{2k} = 1) + \frac{1}{2}P(x_{2k} = 0).$$

Если $P(x_2 = 1) = P(x_2 = 0) = \frac{1}{2}$, то $P(x_{1k} = y_k) = \frac{3}{4}$, $P(x_{1k} \neq y_k) = \frac{1}{4}$; тогда

$$R(x_k, y_k) = \frac{3}{4} - \frac{1}{4} = \frac{1}{2}.$$

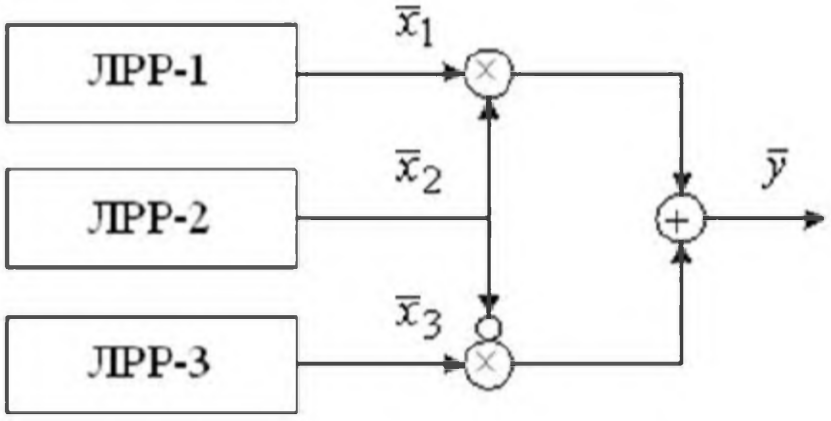


Рис. 3.30. Статическая атака на генератор Джеффа

Пусть известна выходная гамма $\bar{y}$ для датчика, построенного по схеме рис.3.30. Тогда моделируется эта схема при выборе случайнозаполненных ключами ЛРР-2, ЛРР-3при переборе всех возможных заполнений ЛРР-1. Тот ключ, который даст максимальную корреляцию с выходной гаммой $\bar{y}$,

принимается за истинный. Заметим, что в этом случае корреляция вычисляется как *статистическая оценка* теоретической корреляции по следующей общеизвестной формуле:

$$R(\overline{x}_i, \overline{y}) = 1/N \sum_{k=1}^{N} (-1)^{x_{ki}} (-1)^{y_k},$$

где x_{ki}, y_k – это k-е элементы последовательностей $\overline{x}_i$ и $\overline{y}$ соответственно, а N – число анализируемых элементов. Далее таким же образом находится истинный ключ для ЛРР-3 и, наконец, ключ для ЛРР-2, который даст, очевидно, единичную корреляцию при правильном выборе первого и второго ключей. (Метод, который проиллюстрирован на примере генератора Джеффа, очевидным образом распространяется и на датчик гаммы, показанный на рис.3.28.) Таким образом, получаем, что если при тотальном переборе необходимо перебрать $T = 2^{n_1 \cdot n_2 \ldots n_m}$ ключей, то для корреляционной атаки количество опробований будет равно $T' = 2^{n_1} + 2^{n_2} + \ldots + 2^{n_m}$. Очевидно, что $T' << T$, что и доказывает эффективность корреляционной атаки.

Для построения потокового шифра, защищенного от корреляционной атаки, достаточно выбрать длины входящих в него ЛРР так, чтобы количество опробований T' было нереализуемым. (Например, при условии, что $\min n_i \geq 50$.) Если выполнение данного условия невозможно (например, исходя из соображений сложности реализации шифратора или скорости его работы), то необходимо специальным образом выбрать булеву функцию $f(x_1, \ldots, x_m)$. Так, если эта булева функция выбрана корреляционно нечувствительной степени l (см. соответствующее определение в разд. 3.1.8), то это означает, что ненулевая корреляция будет существовать только между гаммой и объединением не менее чем l выходов ЛРР. Тогда количество опробований ключей должно быть не меньше, чем $T' = 2^{l \min n_i} >> T'$.

Быстрые корреляционные атаки. Это такие разновидности корреляционной атаки, которые, в отличие от грубого перебора, используемого в такой атаке, применяют более эффективные методы нахождения наиболее вероятных ключей. Некоторые методы основаны на знании отводов ЛРР, образующих датчик гаммы, и состоят в нахождении корреляции с теми элементами гаммы y_j, которые с высокой вероятностью совпадают с выходами некоторых ЛРР [22].

Другая оригинальная идея быстрой корреляционной атаки [23] состоит в том, что датчик гаммы аппроксимируется объединением ЛРР и *двоичного симметричного канала* с переходной вероятностью ошибки $p(e=1) < 0{,}5$, причем ЛРР, в свою очередь, представляется как кодер *симплексного* двоичного линейного кода с длиной кодовой комбинации $2^n - 1$ и числом информационных символов n. (Такая эквивалентность ЛРР симплексному коду хорошо известна из теории кодирования [8].) Эта аппрок-

симация для потокового шифра показана на рис. 3.31. При таком представлении потокового шифратора задача криптоанализа сводится к исправлению ошибок e, полученных в *виртуальном канале*, и если такие ошибки будут исправлены, то оказывается возможным вычислить n информационных символов кода, которые и совпадают с ключом k, если он выбран как начальное заполнение ЛРР.

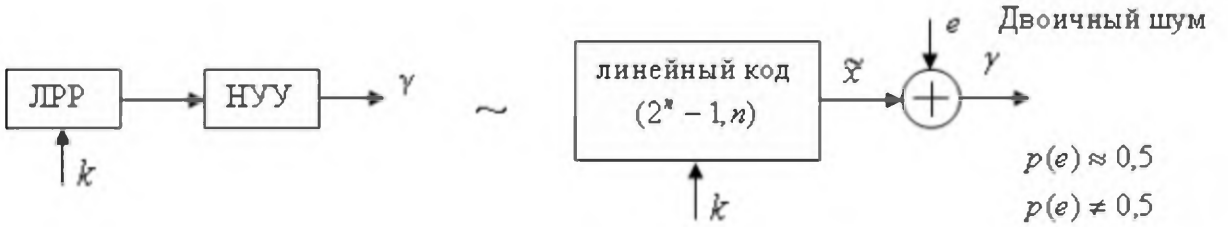


Рис. 3.31. Атака на основе эквивалентности потокового шифратора кодеру симплексного кода и канала с шумом

На эффективность данной атаки влияют два противоречивых факта: с одной стороны, мощный корректирующий код $(2^n-1,n)$, а с другой – большая вероятность ошибки виртуального канала, близкая к 0,5. Основной проблемой при практической реализации подобной атаки является нахождение эффективного метода исправления ошибок при данных условиях. Заметим, что здесь при декодировании невозможно использовать полные длины кодовых комбинаций, равные 2^n-1, поскольку для реальных шифраторов эти величины оказываются необозримыми. Однако укорочение длин кодовых блоков оказывается вполне возможным, поскольку из теории кодирования следует, что требуемая длина кодового блока должна быть порядка $n/(1-h(p(e))$, где $h(x)=-(x\log x+(1-x)\log(1-x))$ – энтропийная функция.

Побочные атаки на потоковый шифр. Основная побочная атака основана на возможности повторения шифрующей гаммы при различных сеансах связи, выполняемых на одном и том же ключе. Если такое событие происходит, то, как уже было отмечено ранее, появляется простая возможность дешифрования сообщений без знания ключа или использования сложных методов криптоанализа. Ввиду особой опасности этой атаки, повторим еще раз ее описание. Пусть на двух сеансах связи использовался один и тот же ключ, но шифровались разные сообщения M_1 и M_2. Криптограммы, соответствующие этим сеансам связи, будут иметь следующий вид:

$$E_1 = M_1 \oplus \gamma(k);$$
$$E_2 = M_2 \oplus \gamma(k),$$

и тогда возможна атака вида

$$E_1 \oplus E_2 = M_1 \oplus \gamma(k) \oplus M_2 \oplus \gamma(k) = M_1 \oplus M_2.$$

Поскольку, как правило, сообщения являются избыточными, а иногда и просто детерминированными, то «распутать» их сумму не представляет большого труда. (Заметим, что сумма двух сообщений по модулю какого-либо числа представляет собой пример «исторического» шифра, когда в качестве ключа выбиралась известная книга, а также номера страниц и строк в ней, а шифрование (и дешифрование) выполнялось при помощи сложения (или вычитания) по модулю целого числа, равного объему алфавита языка, на котором было представлено шифруемое сообщение с выбранным фрагментом ключа.)

Тривиальным методом исключения подобной атаки является смена ключа при каждом новом сеансе связи. Однако это требует большого расхода ключевого материала. Поэтому рассмотрим методы, позволяющие сохранять один и тот же ключ на разных сеансах связи, но изменять гамму. Если потоковый шифр использует ключ, который определяется только полиномом обратных связей ЛРР, то можно на каждом новом сеансе связи изменять начальное заполнение ЛРР. Это начальное заполнение либо выбирается детерминированным, но зависящим от номера сеанса связи и заранее согласованным передающей и принимающей сторонами, либо генерируется случайно при каждом новом сеансе связи на передающей стороне и передается в открытом виде по каналу связи на приемную сторону. Последнее обстоятельство, очевидно, не уменьшает стойкости шифра, поскольку ключом здесь является полином обратных связей ЛРР.

Возможна и более изощренная атака, которая также эксплуатирует слабость потокового шифра, состоящую в использовании суммирования по модулю 2. Для ее выполнения достаточно навязать законному пользователю хотя бы один дополнительный бит, вставленный в его предыдущее сообщение. Пусть соответственно сообщение, гамма и криптограмма первоначально имели вид: $M_1, M_2, \ldots$; $\gamma_1, \gamma_2, \ldots$; $E_1, E_2, \ldots$ Тогда после вставки известного нарушителю сообщения M' сразу после M_1 последовательности сообщения гаммы и криптограммы будут: $M_1, M', M_2, \ldots$; $\gamma_1, \gamma_2, \gamma_3, \ldots$; $E_1, E'_2, E'_3, \ldots$, – и все сообщения сразу после вставки вскрываются нарушителем при помощи цепочки следующих тривиальных вычислений:

$$\gamma_2 = E'_2 \oplus M' \rightarrow M_2 = E_2 \oplus \gamma_2 \rightarrow \gamma_3 = E'_3 \oplus M_2 \rightarrow M_3 = E_3 \oplus \gamma_3, \ldots$$

Защита от такой атаки состоит в отказе от шифрования постороннего материала и изменении гаммы для новых сеансов связи.

Если ключом k шифра является начальное заполнение ЛРР, то необходимо произвести модификацию шифрования, при которой новое начальное заполнение ЛРР будет равно $IV = IV_0 \oplus k$, где k – секретный ключ, а IV_0 – двоичная последовательность длины n, которая генерируется случайно и передается на приемную сторону в каждом новом сеансе связи.

Такой метод также не ослабляет стойкости шифра, поскольку знание IV_0 не дает никакой информации о IV, если ключ k по-прежнему случаен.

Теперь мы можем сформулировать основные требования, предъявляемые к разработке *стойкого потокового шифратора:*

1) количество ключей шифратора должно быть непереборно велико;

2) вырабатываемая шифратором гамма должна иметь необозримо большой период;

3) гамма должна иметь большую величину ЛЭС;

4) гамма должна иметь хороший баланс 0 и 1;

5) схема шифрования должна обеспечивать устойчивость к статистическим атакам, для чего булевы функции, описывающие нелинейные преобразования выходов ЛРР, должны иметь высокий порядок корреляционной нечувствительности;

6) потоковый шифратор должен абсолютно исключать повторение гаммы, генерируемой им на одном и том же ключе.

Существует множество разработок потоковых шифров, удовлетворяющих этим требованиям. Большинство практически используемых шифров основано на использовании ЛРР с псевдослучайным тактированием. Другим направлением разработки потоковых шифров являются так называемые *ранцевые* шифры [21].

3.2.5.Пример практически используемых в стандарте GSM потоковых шифровA5/1…A5/3

Из рис.3.32 видно, что генератор гаммы потокового шифра A5/1 состоит из трех ЛРР длины 19, 22 и 23 с отводами обратных связей, соответствующих примитивным полиномам. Это обеспечивает периоды выходных последовательностей данных ЛРР, равные соответственно $2^{19}-1, 2^{22}-1, 2^{23}-1$.

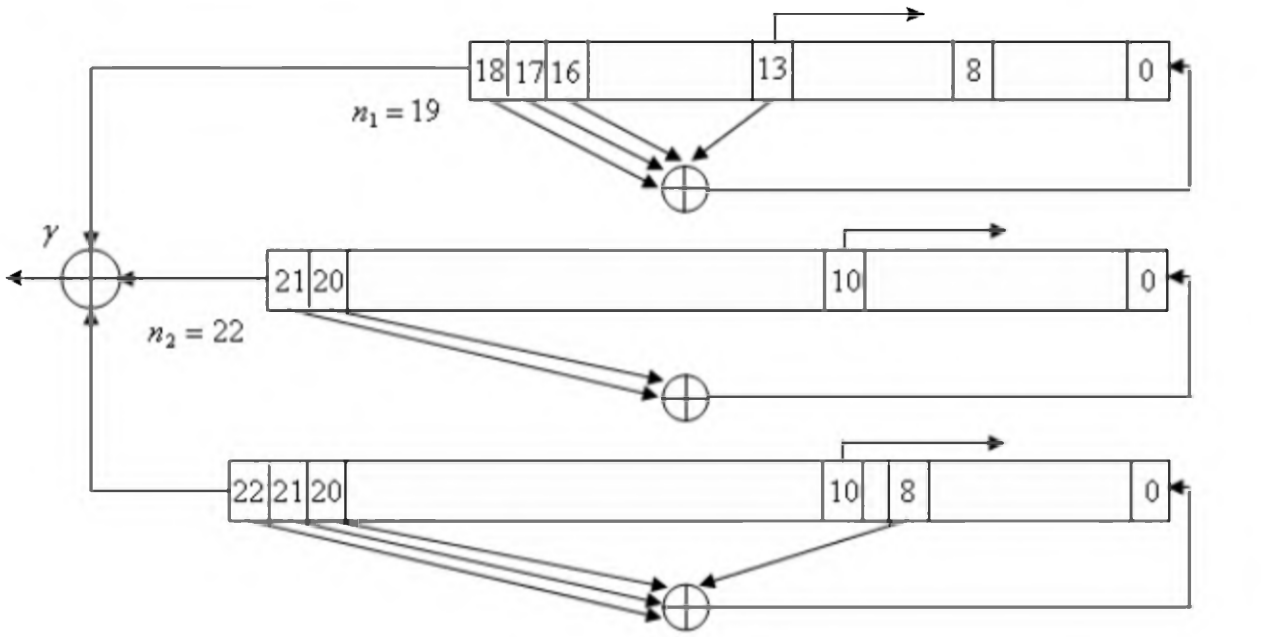


Рис. 3.32. Алгоритм шифрования A5/1 стандарта GSM

Все три регистра используют псевдослучайное тактирование, которое работает по следующему правилу: биты с отвода 8 первого ЛРР, а также с отводов 10 второго и третьего ЛРР подаются на так называемый *мажоритарный элемент*. Последний выдает на выходе значение 0 или 1, в зависимости от того, появляется на его входах больше нулей или единиц. Далее выход этого мажоритарного элемента сравнивается со значениями выходов на трех отводах ЛРР с номерами 8, 10, 10 (которые подавались ранее на входы этого мажоритарного элемента), и каждый ЛРР продвигается на один такт тогда и только тогда, когда сравниваемые биты оказываются одинаковыми. Шифрующая гамма формируется как сумма по модулю 2 выходов всех трех ЛРР. Ключом являются начальные заполнения всехЛРР, которые вводятся на начальном этапе без псевдослучайного тактирования (см. детали в [24]). Длина ключа оказывается,таким образом, равной 19+22+23=64 бита.

Стойкость шифра A5/1. При разработке этого шифра предполагалось, что он будет иметь высокую стойкость, так как количество его ключей достаточно велико, однако дальнейшие исследования, проводившиеся независимыми криптографами [24] показали, что у этого шифра есть слабые стороны. Одна из них состоит в том, что ЛРР, входящие в состав шифратора, имеют малые длины, и поэтому они подвержены некоторым модификациям статистических атак, а также атакам на основе обменных соотношений между требуемым объемом памяти и временем анализа.

В конечном итоге исследования, которые проводились начиная с 2000 г. (т. е. почти сразу же после введения этого стандарта), показали, что данный шифр может быть «взломан» с использованием обычного ПК в реальном времени. Результаты этих исследований для различных типов атак (подробное описание атак можно найти в [24]) помещены в табл. 3.14.

Таблица 3.14

Криптоанализ шифра A5/1

Типы атак	Времяпредварительной обработки	Времяприема гаммы	Количество 73 Gb дисков	Время вскрытия шифра
1	2^{42} тактов	2 мин	4	1 с
2	2^{48} тактов	2 мин	2	1 с
3	2^{48} тактов	2 с	4	1 мин

Как видно из табл. 3.14, шифрA5/1 не обладает высокой стойкостью, даже если он подвергается атакам со стороны непрофессиональных криптографов, имеющих в своем распоряжении лишь персональные компьютеры.
Применение и модификации шифра A5. Шифр A5/1 используется для шифрования сообщений в системе GSM-R на участке «мобильная станция – базовая станция». На мобильную станцию по каналам служебной связи пересылается сеансовый ключ шифрования из состава триплета, используемого центром коммутации в процедуре аутентификации.

В GSM-системах, поставляемых на восточный рынок, используется модификация A5/2 алгоритма A5/1[9]. В схеме стандарта A5/2 используются четыре ЛРР, причем три из них имеют такие же характеристики, как и в стандарте A5/1, а четвертый используется в качестве генератора нерегулярных тактовых импульсов для первых трех ЛРР.
Для усложнения шифрующей гаммы в схеме используется дополнительная нелинейная последовательность, образуемая суммой по mod 2 выходов трех мажоритарных элементов. На вход каждого мажоритарного элемента подается последовательность с некоторых промежуточных разрядов каждого ЛРР.

В поздних версиях GSM (GSM/GPRS) используется более совершенный алгоритм шифрования A5/3, называемый также GEA3. Алгоритм A5/3 представляет собой блоковый шифр Касуми в модификации OFB (разд. 3.1.12). Сам шифр Касуми построен по схеме Фейстеля с 8 раундами [9]. Сокращенная версия данного шифра с использованием только 6 раундов может быть практически взломана с использованием атаки на связанных ключах. Полная версия пока еще требует 2^{55} выбранных сообщений и 2^{74} шифрований [43].

4. АУТЕНТИФИКАЦИЯ СООБЩЕНИЙ

До сих пор рассматривалась лишь одна из функций криптосистем – *конфиденциальность,* т. е. обеспечение секретности содержания передаваемых сообщений, однако криптосистемы могут выполнять и другие функции. Важнейшей из них является *аутентификация сообщений*. Аутентификация обеспечивает проверку *подлинности* (иными словами – *целостности*) сообщений, позволяя с высокой надежностью обнаружить любые их случайные или преднамеренные изменения.

Частным случаем аутентификации сообщений является *аутентификация пользователей* (или авторов) сообщений. Если в сообщении не указано явно имя пользователя, то возникает задача его косвенного и надежного определения. Эта задача называется также задачей *идентификации пользователей*.

Особым и весьма распространенным типом аутентификации является создание *аутентифицированного ключа* между двумя или более пользователями сети связи. Эта задача решается при помощи выполнения этими пользователями определенных последовательностей действий (*протокола аутентификации*), которые обеспечивают не только выполнение собственно аутентификации, но и выполнение других условий, например таких, чтобы:

-распределенный ключ был известен только законным (*легитимным*) пользователям;

-пользователи знали, что их ключ известен их легитимным партнерам;

-пользователи знали, что их ключ был сгенерирован заново.

Методы решения этой задачи изучаются в части III настоящей книги. В данном разделе будут рассматриваться лишь собственно методы аутентификации сообщений, причем *без обеспечения функции конфиденциальности* сообщений одновременно с аутентификацией. Точная постановка этой задачи формулируется в разд. 4.1.

4.1. Общая структура (техника) аутентификации

Пусть имеется сообщение M, представленное в любой форме, например в виде двоичной последовательностей конечной длины. *Аутентификатором* E_s к этому сообщению называется некоторая последовательность, которая является функцией от этого сообщения и секретной последовательности k (ключа), т. е. $E_s = f(M,k)$. Таким образом, *аутентифициро-*

ванное сообщение представляет собой пару: сообщение, аутентификатор – (M, E_s). Для проверки подлинности сообщения необходимо иметь тот же самый ключ k и знать функцию $f(\cdot)$. Тогда по известной паре $(\tilde{M}, \tilde{E}_s)$ законный пользователь вычисляет аутентификатор $\tilde{\tilde{E}}_s = f(\tilde{M}, k)$ и сравнивает его с принятым аутентификатором $\tilde{E}_s$. Если $\tilde{\tilde{E}}_s = \tilde{E}_s$, то сообщение признается подлинным, в противном случае – ложным (т. е. не подлинным).

Различают два вида атак на системы аутентификации:

1) *атака имперсонизации*, когда злоумышленник пытается подменить сообщение M на другое сообщение M', не зная аутентификатора E_s и ключа k;

2) *атакаподстановки*, когда злоумышленник, зная пару (M, E_s), по заданному им ложному сообщению M' пытается построить такой ложный аутентификатор E'_s, что законный пользователь не сможет обнаружить эту подмену и примет пару (M', E'_s), как подлинную.

4.2. Классификация систем аутентификации и характеристики их эффективности

Различают два основных типа систем аутентификации:

-безусловно стойкие;

-вычислительно стойкие.

Эффективность безусловно стойких систем аутентификации определяется следующими параметрами: вероятностью успешной атаки имперсонизации p_i, вероятностью успешной атаки подстановки p_s, вероятностью необнаруженного навязывания $p_D = \max[p_i, p_s]$, длиной b аутентификатора E_s и объемом (длиной в битах N) требуемого для аутентификации ключа k.

Эффективность вычислительно стойких систем аутентификации определяется той же совокупностью параметров и, кроме того, минимально возможной сложностью вычислений (числом операций и объемом памяти), при которой данные вероятностные характеристики еще выполняются.

Таким образом, безусловно стойкие системы являются таковыми независимо от вычислительного ресурса злоумышленника, тогда как для вычислительно стойких систем их надежность зависит от этого вычислительного ресурса. Хотя на первый взгляд кажется, что всегда надо отдавать предпочтение безусловно стойким системам,это на самом деле не так. Как будет показано в следующих разделах, безусловно стойким системам присущ существенный недостаток (большой расход ключа), который резко

ограничивает область их применения. (Заметим, что тут имеется полная аналогия с совершенными и вычислительно стойкими шифрами, которые рассматривались ранее в разд. 2 данной части книги.)

4.3. Безусловно стойкие системы аутентификации

В [25] приводятся следующие нижние границы для вероятностей ошибок, характеризующих безусловно стойкие системы аутентификации:

$$p_i \geq 2^{-I(E_s;K)}; \tag{4.1}$$

$$p_s \geq 2^{-H(K/E_s)}; \tag{4.2}$$

$$p_D \geq 1/\sqrt{|k|}, \tag{4.3}$$

где $I(X;Y)$, $H(Y/X)$ означают количество информации и условную энтропию (по Шеннону), вычисленные для соответствующих случайных переменных, а $|k|$ означает общее количество ключей, используемых для аутентификации. (Очевидно, что если ключ представляет собой двоичную цепочку длиной N, то граница (4.3) принимает вид

$$p_D \geq 2^{-N/2}. \tag{4.4}$$

Если для некоторой системы аутентификации выполняется равенство в (4.4), то такая система называется *совершенной.* В [25] приводится необходимое условие для получения совершенной системы аутентификации

$$|M| \leq \sqrt{|k|} + 1. \tag{4.5}$$

Неравенство (4.3) на первый взгляд кажется неверным, поскольку случайность ключа как бы не используется полностью для обеспечения случайности подмены. Однако кажущееся противоречие объясняется тем обстоятельством, что знание злоумышленником аутентификатора, соответствующего известному сообщению, уменьшает неопределенность выбора ключа, при котором эта подмена не будет обнаружена. (Например, для системы аутентификации, показанной на рис.4.1, при получении злоумышленником сообщения M и аутентификатора E_s он мог бы выбрать не любой из 8 ключей, а лишь один из двух: k_7 или k_8.) Тогда злоумышленник, который хочет подменить сообщение M на M', может с равной вероятностью выбрать аутентификатор E'_s или E''_s.

Для построения систем аутентификации с гарантированными величинами p_i, p_s нам потребуется представить некоторый дополнительный математический аппарат.

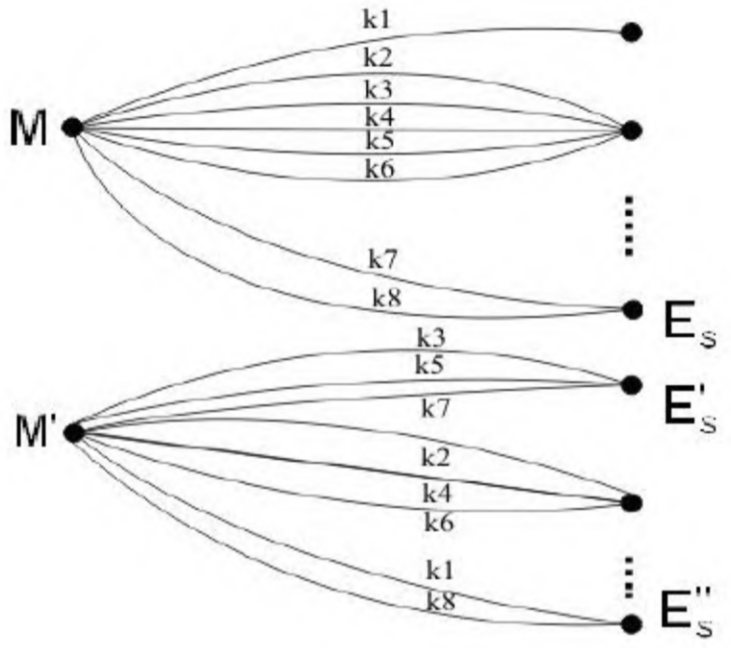


Рис. 4.1. Безусловно стойкая система аутентификации

Определение 4.1. *Хеш-функцией* $y = h(x)$ называется преобразование, отображающее множество всех двоичных последовательностей X длины n во множество двоичных последовательностей Y длины b, где $b < n$. Символически это преобразование можно записать так: $GF(2)^n \to GF(2)^b$.

Хеш-функции бывают *ключевыми* и *бесключевыми* (т. е. зависящими или не зависящими от ключа). Если хеш-функция является ключевой, то можно говорить о классе хеш-функций, где каждая функция из класса соответствует выбору определенного ключа.

Определение 4.2. Классом *строго универсальных* хеш-функций называется такое множество отображений $H : X \to Y$, что:

1) для любых $x \in X$, $y \in Y$: $\#\{h \in H : y = h(x)\} = \dfrac{|H|}{|Y|}$,

где $|H|$ – общее количество хеш-функций h, $|Y|$ – общее количество аутентификаторов Y, #{..} – количество хеш-функций, удовлетворяющих условию, представленному в фигурных скобках;

2) для любых $x_1, x_2 \in X$, $x_1 \neq x_2$ и $y_1, y_2 \in Y$

$$\#\{h \in H : h(x_1) = y_1, h(x_2) = y_2\} = \frac{|H|}{|Y|^2}.$$

Пример строго универсального класса хеш-функций. Пусть $x \in GF(2)^n$, $y = h(x) = (x \times h_0 + h_1)_b$, где «×», «+» означают соответственно умножение и сложение в конечном поле $GF(2^n)$, h_1, h_0 – двоичный ключ общей длиной $2n$, $(...)_b$ означает «усечение», т. е. выбор b левых (или

правых) бит последовательности, стоящей в скобках. Легко проверить, что такой класс хеш-функций является строго универсальным.

Рассмотрим метод построения безусловно стойкой системы аутентификации на основе использования хеш-функций, выбранных из строго универсального класса, приведенного выше. В такой системе длина сообщения должна быть равна n, длина ключа $2n$, а длина аутентификатора $b \le n$.

Утверждение 4.1. Если для аутентификации используются функции, выбираемые равновероятно из строго универсального класса, то вероятность успешного выполнения оптимальной атаки подстановки будет равна $p_s = \frac{1}{2^b}$, где b – длина аутентификатора.

Доказательство. Пусть имеется некоторое сообщение M, и законный пользователь выбрал некоторый ключ k_0, который определяет единственную хеш-функцию из строго универсального класса. По ключу k_0 этот пользователь вырабатывает аутентификатор $E_s = h(M, k_0)$. Обозначим через H' множество всех ключей из H, которые переводят заданное M в такой аутентификатор E_s (рис.4.2).

Пусть злоумышленник хочет сформировать ложное сообщение M', которое законный пользователь аутентифицировал бы как правильное. Тогда он должен создать ложный аутентификатор $E'_s = h(M', k_0)$. Однако он не знает ключ k_0, которым воспользовался бы для проверки законный пользователь. Вся информация, которую имеет злоумышленник при известных ему M и E_s (даже при его неограниченном вычислительном ресурсе), – это множество ключей H' (рис. 4.2). Поэтому *оптимальная стратегия* злоумышленника состоит в том, чтобы случайно (наугад) выбрать ключ k'_0 из множества H'. Тем не менее ему не обязательно в точности угадать истинный ключ k_0, а достаточно угадать один из ключей, принадлежащих H', которые дают отображение $M' \to E'_s$. Обозначим множество таких ключей через H'' (рис. 4.2). Согласно свойству 1 определения 4.2 строго универсального класса, количество ключей из H' равно в точности $|H'| = \frac{|H|}{|E_s|}$, а по свойству 2 того же определения количество ключей из H'' будет в точности равно $|H''| = \frac{|H|}{|E_s|^2}$. Тогда вероятность того, что злоумышленник, зная множество H', угадает один из элементов множества H'', т.е. что навязывание ложного сообщения M' окажется успешным, будет следующей:

$$p_s = \frac{|H''|}{|H'|} = \frac{1}{|E_s|} = \frac{1}{2^b}, \qquad (4.6)$$

что и требовалось доказать.

Сравним полученный результат с нижней границей для вероятности необнаруженной подстановки (4.3):

$$p_s \geq \frac{1}{\sqrt{|k|}} = \frac{1}{\sqrt{2^{2n}}} = \frac{1}{2^n}.$$

Так как в нашей схеме $p_s = \frac{1}{2^b}$, то эта граница достигается лишь при $b = n$, и для этого частного случая схема аутентификации на основе использования для аутентификации универсальных хеш-функций оказывается совершенной.

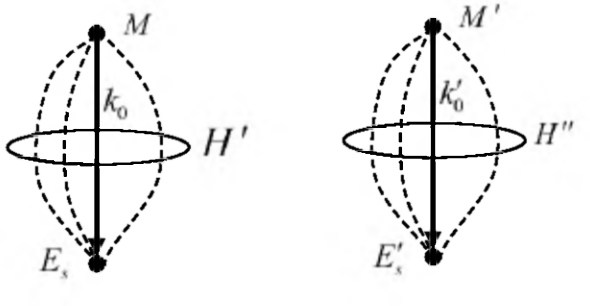


Рис. 4.2. Атака подстановки на систему аутентификации, использующую хеш-функции из универсального класса

Приведенный выше алгоритм аутентификацииимеет, однако, два существенных недостатка:

1) длина ключа достаточно велика (она всегда в 2 раза превышает длину сообщения!);

2) схема совершенна лишь при длине аутентификатора, равной длине сообщения.

Эти особенности неприемлемы при многих практических применениях. Кроме того, общим недостатком всех безусловно стойких схем аутентификации является то, что они могут быть использованы лишь *однократно*. Это означает, что если необходимо аутентифицировать на одном и том же ключе несколько сеансов при передаче различных сообщений, то это приведет к тому, что вероятность $p_s \to 1$ при увеличении числа сеансов. Этот факт следует из результата, приведенного в [39], который заключается в том, что если нарушитель наблюдает L пар сообщений аутентификаторов, то для вероятности оптимального навязывания P_2 будет справедлива нижняя граница:

$$P_2 \geq 1 / \frac{L+1}{\sqrt{|k|}}.$$

Для исключения последнего недостатка существует модификация, предназначенная для обеспечения безопасной работы многократных систем аутентификации, которую мы сейчас опишем. Пусть имеется система однократной аутентификации, основанная на использовании строго универсальных хеш-функций, которая определяется следующим уравнением

выработки аутентификатора: $E_s = h(k, M_1)$, где k – секретный ключ, M_1 – аутентифицируемое сообщение. Тогда аутентификатор для L-кратной системы предлагается формировать так: $E_s = (E_{S_1}, E_{S_2}, ..., E_{S_L})$, где

$$E_{s_i} = h\left(k, M_i\right) \oplus k_i, \tag{4.7}$$

$i = 1, 2, 3, ..., L$; k_i – дополнительный b-битовый ключ для i-го сеанса аутентификации.

В [26] доказывается, что если при использовании такой системы злоумышленник наблюдает L сообщений $M_1, M_2, ..., M_L$ и L соответствующих им аутентификаторов $E_{S_l}, ..., E_{S_l}$ и знает, что последние сформированы по правилу (4.7) (но, конечно, не знает секретных ключей $k, k_1, ..., k_L$), то вероятность необнаруженной подстановки любого ложного сообщения $M' \neq M_i, i = 1, 2, ..., L$ не может быть больше, чем $P_s = \frac{1}{2^b}$, где b (как и ранее) – число бит в аутентификаторе. Таким образом, в данной системе многократной аутентификации обеспечивается та же надежность, что и ранее была в однократной системе, но длина требуемого для этого ключа возрастает до $2n + Lb$ бит, что может оказаться, однако, значительно меньше, чем $2nL$ бит ключа k_i, который надо было бы использовать при смене его на каждом сеансе.

Для того чтобы избавиться от отмеченных ранее недостатков приведенной выше схемы однократной аутентификации (большая длина ключа и большая длина аутентификатора), рассмотрим другую схему аутентификации. Пусть сообщение M представлено в следующем виде: $M = \left(M_1, M_2, \ldots, M_t\right)$, где $M_i \in GF\left(2^n\right)$. Тогда сформируем аутентификатор следующим образом:

$$E_s = \sum_{i=1}^{t} M_i k_1^i + k_2,$$

где $k_1, k_2 \in GF\left(2^n\right)$ – ключи, а все действия выполняются в поле $GF(2^n)$. Для такой схемы мы получаем число всех возможных сообщений $|M| = 2^{nt}$, число всех возможных аутентификаторов $|E_s| = 2^n$, полное число ключей $|k| = 2^{2n}$. Тогда легко рассчитать *скорость аутентификации* (как отношение числа бит сообщения к сумме числа бит сообщения и аутентификатора), которое оказывается равным $R = \frac{t}{t+1}$. (Видно, что в этом случае в качестве аутентификатора просто выбирается один элемент конечного поля $GF(2^n)$.) Скорость расхода ключа в такой системе (т. е. отношение длины ключа к

длине сообщения) будет, очевидно, равна $R_k = \frac{2n}{tn} = \frac{2}{t}$. Можно доказать[39], что для такой схемы вероятность необнаруженной подстановки будет иметь вид:

$$P_s = \frac{t}{2^n}. \qquad (4.8)$$

Таким образом, для рассмотренной выше системы аутентификации получается, что при использовании длинных сообщений скорость аутентификации оказывается близкой к единице, скорость расхода ключа – близкой к нулю (что приближает ее по эффективности к системе без аутентификации), тогда как вероятность появления необнаруживаемого навязывания может быть сделана достаточно малой, и она лишь в t раз больше, чем в совершенных системах аутентификации.

Можно добиться и более свободного варьирования параметрами системы аутентификации, если перейти к выбору хеш-функций из так называемого *почти строгоуниверсального* класса [26] или использовать технику *интерактивной аутентификации* [26], но рассмотрение этих методов выходит за рамки нашей книги. Тем не менее, какие бы методы безусловно стойкой аутентификации ни использовались, все они имеют уже отмечавшийся ранее существенный недостаток. Если мы хотим отдельно аутентифицировать сообщения, передававшиеся на разных сеансах связи (или, скажем, содержащиеся в разных файлах), то это неизбежно приведет к *росту длины требуемого секретного ключа,* пропорциональному числу сеансов (файлов). Для того чтобы избавиться от этого недостатка и получить возможность многократно аутентифицировать различные сообщения при помощи одного и того же достаточно короткого ключа, необходимо переходить к использованию вычислительно стойких систем аутентификации.

4.4. Вычислительно стойкие системы аутентификации

Система аутентификации называется *вычислительно стойкой*, если возможность необнаружения подделки сообщения определяется вычислительной мощностью нарушителя.

4.4.1.Основные определения и классификация

Различают два класса вычислительно стойких систем аутентификации с использованием:

1) *симметричных* методов шифрования;

2) *несимметричных* методов шифрования.

В первой части книги «Основы криптографии» мы рассматриваем только системы аутентификации, принадлежащие к первому классу. (Системы аутентификации из второго класса будут изучаться в части II этой книги («Основы криптографии с открытым ключом»), где как раз и рассматриваются несимметричные методы шифрования.)

Вычислительно стойкие системы аутентификации, также как и безусловно стойкие, используют в основном технику хеш-функций, сконструированных, однако, по другому принципу. Эти хеш-функции делятся также на два основных класса:

- *ключевые хеш-функции* (т. е. такие, которые зависят от секретного ключа, разделяемого между законными пользователями);

-*бесключевые хеш-функции*, которые вычисляются по полностью открытому алгоритму.

В последнем случае, для того чтобы обеспечить преимущество законным пользователям перед незаконными, необходимо после выполнения бесключевого хеширования выполнить дополнительную процедуру с использованием ключей. В этом случае роль хеш-функций состоит в существенном уменьшении длины аутентифицируемого сообщения, что позволяет упростить процедуру собственно аутентификации, которая обычно выполняется с использованием несимметричной техники шифрования, называемой *цифровой подписью*. Поскольку рассмотрение вопросов несимметричного шифрования производится в части II книги, мы перенесем туда же и изучение бесключевых хеш-функций и ограничимся рассмотрением в данном разделе только ключевых хеш-функций.

Заметим, что в иностранной технической литературе принято называть ключевые хеш-функции, предназначенные для криптографического применения, *кодом аутентификации сообщений* (кратко – *MAC-кодом*), тогда как бесключевые хеш-функции называют *кодом, обнаруживающим модификации* (кратко – *MDC-кодом*).

4.4.2. Основные свойства и способы построения ключевых хеш-функций

Не умаляя общности, будем предполагать далее, что входом для MAC-кода являются двоичные последовательности x длины m, а выходом – двоичные последовательности y длины l, и обозначать это преобразование через $h(k,x)$, где k – секретный ключ аутентификации. Тогда аутентифицированное сообщение M будет представлять собой пару (x,y), где $x=M, y=h(k,x)$. Для проверки подлинности аутентифицированного сообщения $(\tilde{x},\tilde{y})$ законный пользователь вычисляет $\tilde{\tilde{y}}=h(k,\tilde{x})$ и сравни-

вает его с $\tilde{y}$. При совпадении этих последовательностей сообщение полагается подлинным, а при несовпадении – искаженным.

Определение 4.3. *Ключевой криптографической хеш-функцией* (MAC) называется функция $h(k,x)$, обладающая следующими свойствами:

1. Она легко вычисляется для известных x,k.

2. Длина выходной последовательности l может быть выбрана, вообще говоря, произвольной, но в типичных случаях $l \ll m$, где m – длина входной последовательности.

3. Если задано произвольное количество аутентифицированных пар: $\left(x_i, h\left(k, x_i\right)\right)$, $i = 1, 2, \ldots,$ – то вычислительно невозможно найти ни одной новой аутентифицируемой пары $(x, h(k,x))$ для любого нового входа $x \neq x_i$ при неизвестном ключе k (включая и случай, когда $h(k,x) = h(k,x_i)$ для некоторого i).

Свойство 3 означает, что MAC является *вычислительно стойким* методом аутентификации, поскольку злоумышленник, имея в своем распоряжении многократно аутентифицированные сообщения, не может без знания ключа создать новое сообщение и аутентификатор к нему так, что оно будет воспринято, как подлинное законным пользователем, обладающим секретным ключом k.

Заметим, что свойство 3, хотя и предполагает вычислительную невозможность нахождения ключа k по известным парам $x_i, h(k,x_i)$, не предполагает при выполнении последнего условия вычислительную стойкость системы аутентификации, поскольку нахождение секретного ключа не всегда необходимо для успешной подделки сообщения.

Построение MAC на основе CBC-моды блоковых шифров. Наиболее распространенным способом построения MAC является использование CBC-мод различных блоковых шифров. Пусть имеется некоторый блоковый шифр с длиной блока n и с алгоритмом шифрования $E = f_k(M)$. Предположим, что сообщение M имеет произвольную длину m (не обязательно кратную n). Тогда первоначально производится процедура *добавления* b дополнительных бит (обычно известной последовательности), что обеспечивает кратность $m+b$ длине блока шифра n. После этого формирование аутентификатора y длины $l = n$ (выхода хеш-функции) выполняется последующемуалгоритму:

$$y = y_t,\ y_i = f_k\left(M_i \oplus y_{i-1}\right),\ i = 1, 2, \ldots, t,\ y_0 = 0,\ t = (m+b)/n, \qquad (4.9)$$

где M_i – это i-й подблок сообщения длины n, т. е.

$$M = \left(M_1, M_2, \ldots; M_i, \ldots; M_{(m+b)/n}\right).$$

Очевидно, что длина MAC оказывается равной длине блока выбранного шифра n, сложность выполнения аутентификации определяется сложностью шифрования, а вычислительная стойкость аутентификации – стойкостью этого шифра.

Если длина ключа в выбранном блоковом шифре не допускает его полного перебора, а длина аутентификатора меньше длины ключа, то атака имперсонизации может состоять в попытке случайного угадывания правильного аутентификатора для ложного сообщения, при которой, очевидно, вероятность такого угадывания будет равна 2^{-l}, где l – длина аутентификатора. Атака подстановки на такую систему аутентификации затруднена из-за вычислительной сложности нахождения даже единственно возможного ключа для заданного сообщения и правильного аутентификатора. Здесь уместно еще раз подчеркнуть принципиальную разницу между вычислительно стойкой и безусловно стойкой системами аутентификации, поскольку для последней не представляет труда вычислить хотя бы один ключ, который при известном сообщении M дает известный аутентификатор E_s. Действительно, если хеширование в строго универсальном классе было задано соотношением $y=(x\times h_0+h_1)_b$, где $x=M, y=E_s$ (разд. 4.3), то, дополняя двоичную последовательность y длины b произвольной двоичной последовательностью $\tilde{b}$ длины $n-b$ так, что длина их конкатенации (объединения) $\tilde{\tilde{y}}=(b,\tilde{b})$ оказывается равной n, можно для произвольного ключа $\tilde{h}_1$ найти соответствующий ему ключ $\tilde{h}_0=(\tilde{y}-h_1)x^{1}$ для любого $x\neq 0$, причем сложность этих вычислений в конечном поле $GF(2^n)$ оказывается полиномиальной по n.

Следовательно, можно достаточно просто вычислить хотя бы один ключ $(\tilde{h}_0,\tilde{h}_1)$, который формирует заданный аутентификатор E_s для заданного сообщения M. Однако, как уже отмечалось ранее, проблема тогда состояла в том, какой ключ в действительности был использован среди огромного множества допустимых ключей.

Заметим, что существуют модификации MAC, где длина аутентификатораотличается от длины блокового шифра [9].

Построение MAC на основе MDC. Известны различные методы построения MAC на основе MDC [9]. Один из рассмотренных в [9] методов состоит в том, что сначала при помощи MDC находится промежуточный аутентификатор, не зависящий от ключа, а затем уже над ним выполняется некоторое преобразование (подобное шифрованию), зависящее от секретного ключа. Чаще всего в качестве такого преобразования используется несимметричное шифрование.

Другим вариантом построения MAC на основе MDC является использование ключа как части сообщения, поступающего на вход MDC. В этом случае такая хэш-функция называется HMAC [9], и она имеет следующий вид: $HMAC=h\left(k\|P_1\|h\left(k\|P_2\|M\right)\right)$, где «||» означает конкатена-

цию, P_1, P_2–цепочки данных, используемые для пополнения входных данных до полных блоков, k – секретный ключ, а $h(\cdot)$ –бесключевая хэш-функция.

Построение MAC с использованием специально разработанных алгоритмов. В этом случае сообщение обычно разбивается на блоки определенной длины, и далее эти блоки подвергаются определенным преобразованиям, включающим, например, сложение по модулю определенного числа, и управляемым секретным ключом [9]. Такой метод в англоязычной литературеназывается «предписанным» (dedicated).

4.4.3. Пример практически используемой вычислительно стойкой системы аутентификации (режим выработки имитовставки для стандартов ГОСТ Р34.12-2015, ГОСТ Р34.13-2015)

В этом режиме используется модификация блокового шифра CBC, рассмотренная в разд. 3.1.12. То есть сообщение M разбивается на подблоки длиной 64 или 128 бит (поскольку именно такие длины имеют шифры, используемые в ГОСТ Р34.12-2015). Далее последний подблок дополняется (если в этом есть необходимость) до длины 64 (128) бит, и затем последовательно выполняется преобразование (4.9), где в качестве функции $f_k(\cdot)$ используется шифрование по ГОСТ при выборе ключа длиной 256 бит. В отличие от классического режима CBC в данном режиме в процедуре обработки последнего блока на вход базовому алгоритму блочного шифрования подается покомпонентная сумма трех блоков: последнего блока сообщения, результата зашифрования на предыдущем шаге и вспомогательного ключа. Результатом выполнения полного цикла преобразования будет, очевидно, двоичный блок длиной 64 (128) бит, который может быть сокращен до s бит при помощи выбора s символов с большими номерами, взятых из полученного 64 (128)-битового блока. Последовательность из s бит и называется *имитовставкой*, которая присоединяется к полному сообщению. Для верификации принятого сообщения по известному ключу и принятому сообщению вычисляется имитовставка по правилу (4.9), которая сравнивается с принятой имитовставкой, и при их совпадении сообщение полагается подлинным, а в противном случае – отвергается.

Как отмечалось в разд. 4.4.2, стойкость такой аутентификации определяется стойкостью используемого шифра, и поскольку шифр ГОСТ считается стойким и, конечно, не допускает перебора всех его 2^{256} ключей в обозримое время, то и основанную на этом шифре аутентификацию можно полагать стойкой. Для дальнейшего ее улучшения можно увеличить длину имитовставки.

4.5. Система аутентификации на основе каналов с шумом

До сих пор основным условием обеспечения аутентификации сообщений, передаваемых между пользователями, было предварительное распределение ключей между ними. Однако, как уже было описано ранее в разд. 2.2.1, могут существовать и другие преимущества легитимных пользователей перед нелегитимными, например, когда существует некоторый центр распределения ключей (ЦРК), которому доверяют легитимные пользователи, и этот ЦРК может передавать данные по *открытым шумным каналам связи* к этим пользователям при условии, что нелегитимные пользователи имеют возможность перехватывать эти данные, но также по шумным каналам. Данный подход был подробно рассмотрен в [40].

Список литературы к части I

Основная

1. Бабаш, А. В. История криптографии / А. В. Бабаш, Г.П.Шанкин. – М. : Гелиос АРВ, 2002.

2. Kerckhoffs, A. La Cryptographie militaire / A. Kerckhoffs // Journales Scientices Militares. 9th Series. – 1983.February. – P. 161–191.

3. Гнеденко,Б. В.Курстеориивероятностей / Б. В. Гнеденко. – М.,1961.

4. Колесник, В. Д.Курстеорииинформации / В.Д.Колесник, Г.Ш.Полтырев. – М. : Физматгиз,1982.

5. Heys,H.M.TutorialonLinearandDifferentialCryptanalysis. (Unpublished)

6. Логачев, О. А. Булевы функции в теории кодирования и криптологии / О.А.Логачев [и др.] ; МЦНМО. – М., 2004.

7. Виноградов, И. М. Основы теории чисел / И.М.Виноградов. – М. : Физматгиз,1965.

8. Питерсон, У. Коды, исправляющие ошибки / У.Питерсон, Э.Уэлдон. – М. :Мир, 1976.

9. Menezes, A. J.HandbookofAppliedCryptography / A. J. Menezes[et.al.]. –New-York, London, Tokyo :CRCPress, 1997.

10. Hong,S. Provable Security Against Differential and Linear Cryptanalysis for SPN Structure / S. Hong [etal.]. FSE-2000 // LNCS. – 1978. – P. 273–278.

11. Изотов, Б. В. МетодыконструированияблочныхшифровнабазеSPиSL-сетей / Б. В. Изотов //Вопросызащитыинформации. – 2004. – № 3. – С. 24–37.

12. Andelman, D. On the Cryptanalysis of Rotor Machines and Substitution – Permutation Networks / D.Andelman, J.Reeds // IEEE Trans. on IT. – 1981. – Vol IT-28, №4.

13. Бабаш, А. В.Криптография / А.В.Бабаш, Г.П.Шанкин. – М. :Солон, 2002.

14. Courtois, N. Cryptanalysis of Block Ciphers with Overdefined Systems of Equations / N.Courtois, J.Pieprzyk. –http://www.nicolascourtois.net.

15. Biham, E. Differential Fault Analysis of Secret Key Cryptosystems / E.Biham, A.Shamir // LNCS. –1997. – Vol.1294. – P.513–525.

16. Mukherjii, A. Fault-based Analysis of Flexible ciphers / A.Mukherjii [etal.] // Comp. Science Journal of Moldova. – 2002. –Vol.10, №2(29). – P.223–236.

17. Oorschot, P. A Known Plaintext Attack on Two-key Triple Encryption / P.Oorschot, M.Wiener // LNCS. –1998. –Vol.1300. –P.318–325.

18. Коржик, В. И. Теоретические основы информационной безопасности телекоммуникационных систем : учебное пособие / В. И. Коржик, Д. В. Кушнир ; ГУТ. – СПб., 2000.

19. Зенин, О. С. Стандарт криптографической защиты AES / О.С.Зенин, М.А.Иванов. – М. : Кудиц-образ, 2002.

20. Холл,М.Комбинаторика / М. Холл. – М. :Мир, 1970.

21. Rueppel, R. A. Analysis and Design of Stream Ciphers / R. A. Rueppel. – Berlin :Springer – Verlag, 1986.

22. Penzhorn, W. Correlation Attacks on Stream Ciphers / W.Penzhorn // LNCS. – 1995. – Vol. 1039. – P. 159–172.

23. Mihaljevic, M. A Fast Iterative Algorithm for a Shift Register Initial State Reconstruction Given in the Noisy Outlet Sequence / M.Mihaljevic, J.Golic // LNCS. – 1995. – Vol. 453. – P. 165–175.

24. Biryukov, A. Real Time Cryptanalysis of A5/1 on PC / A. Biryukov [etal.]. –http://cryptoml.org/a5.ps.

25. Tilborg, Henk C.A. van. Fundamentals of Cryptology / Henk C.A. van Tilborg. –Kluwer, 2000.

26. Wegman, M. New Hash Functions and Their Use in Authentification and Set Equality / M.Wegman, L.Carter // Journal of Computer and System Sciences. – 1981. – 22. – P. 265–279.

27. Maurer, U. Secret – key Agreement Over Unauthenticated Public Channels – Part III: Privacy Amplification / U. Maurer, S.Wolf // IEEE Trans. on IT. – 2003. – Vol. 49. – № 4. – P. 839–851.

28. Коржик, В. И. Расчет помехоустойчивости систем передачи дискретных сообщений : справочник / В. И. Коржик [и др.]. – М. : Радио и связь, 1981.

Дополнительная

29. Maurer, U. Secret Key Agreement by Public Discussion Based on Common Information / U.Maurer // IEEE Trans. on IT. – 1993. – Vol. 39. – P. 733–742.

30. Korzhik, V. Optimization of Key Distribution Protocols Based on Noisy Channels within Active Adversaries / V. Korzhik [etal.] // LNCS.

31. Aono, T. Wireless Secret Key Generation Exploiting Reactance-Domain Scalar Response of Multipath Fading Channels / T. Aono [etal.] // IEEE Trans. Antennas Propag. – Vol. 53. – P. 3776–3781.

32. Korzhik, V. Secret Key Agreement over Multipath ChannelsExploiting Variable-Directional Antenna / V. Korzhik [etal.] // Int. Journal of Advanced Computer Science and Applications. – 2012. – № 1.

33. Bennett, C. Experimental Quantum Cryptography / C. Bennett [etal.] // Journal of Cryptology. –1992. – Vol. 5. – № 1.

34. Applied Physics Letters. – 2010. – Vol. 96.

35. Nyberg, K. Differentially uniform mappings for cryptography / K. Nyberg // LNCS. – 1994. – Vol. 765. – P. 55–64.

36. Kocher, P. Introduction to Differential Power Analysis and Related Attacks / P. Kocher [etal.]. – Cryptography Research, 1998.

37. Коржик, В. И. О возможности взлома аппаратной реализации шифра ГОСТ / В. И. Коржик, С. В. Тихонов // Проблемы информационной безопасности. Компьютерные системы. – 2012. – № 3. – С. 53–61.

38. Тихонов, С. В. Методы защиты аппаратной реализации шифра ГОСТ от атаки измерения потребляемой мощности в цепи питания / С. В. Тихонов, В. И. Коржик // Проблемы информационной безопасности. Компьютерныесистемы. –2013. – № 3.

39. Preneel, B. CryptographicHashFunctions / B. Preneel. – KluwerAcad.Publishers, 1993.

40.Коржик, В. И. Основы криптографии : учебное пособие / В. И. Коржик, В. П. Просихин. – СПб. :Линк, 2008.

41. ГОСТ Р 34.11–2012. Национальный стандарт Российской Федерации. Информационная технология. Криптографическая защита информации. Функция хэширования. – Издание официальное. – М. : Стандартинформ, 2012.

42. Hellman, M. A cryptoanalytic Time-Memory Trade off / M. Hellman // IEEE Trans. on IT. – 1980. – Vol. 26, № 4. – P. 401–405.

43. Kim, J. Related –Key Boomerang and Rectangle Attacks / J. Kim [et al].

44. Шнайер, Б. Прикладная криптография / Б. Шнайер. – М. : Триумф, 2002.

45. ГОСТ Р 34.12–2015. Национальный стандарт Российской Федерации. Информационная технология. Криптографическая защита информации. Блочные шифры. – Издание официальное. – М. : Стандартинформ, 2015.

46. ГОСТ Р 34.13–2015. Национальный стандарт Российской Федерации. Информационная технология. Криптографическая защита информации. Режимы работы блочных шифров. – Издание официальное. – М. : Стандартинформ, 2015.

ЧАСТЬ II

КРИПТОСИСТЕМЫ С ОТКРЫТЫМ КЛЮЧОМ

1. ИДЕЯ ПОСТРОЕНИЯ КРИПТОГРАФИИ С ОТКРЫТЫМ КЛЮЧОМ (ОК)

Основная идея состоит в создании таких криптосистем, когда алгоритм шифрования и дешифрования, а также ключ шифрования являются полностью открытыми, но не смотря на это, найти ключ дешифрования или сообщение не представляется возможным в обозримое время. Такое преобразование можно назвать *односторонним с подсказкой*, поскольку прямое преобразование (шифрование) выполняется просто, тогда как обратное преобразование (дешифрование) без дополнительной подсказки (ключа дешифрования) выполняется весьма сложно.

Рассмотрим для примера в качестве иллюстрации понятия «однонаправленности» *большой телефонный справочник*, в котором легко найти номер по названию организации или фамилии, но по номеру определить фамилию практически невозможно. Данный метод можно использовать в примитивной криптосистеме:

Шифруемое слово	**Фамилия**	**Номер телефона=шифр**
Д	Дмитриев	777-11-11
О	Ослов	222-33-31
Р	Ротов	730-54-89
О	Оков	456-58-32
Г	Глазьев	354-48-78
А	Андронов	324-50-38

Для легальных пользователей должен существовать специальный справочник, позволяющий по номеру телефона находить фамилию абонента.

На этом примере видна идея метода, состоящая в том, что легальным пользователям нужно давать некоторую подсказку в виде ключа дешифрования (в примере выше это специальный справочник).

Определение. *Асимметричной* криптосистемой (*криптосистемой с* ОК) называется такая криптосистема, в которой ключ дешифрования неравен ключу шифрования, причем невозможно найти ключ дешифрования по известному ключу шифрования.

Рассмотрим математическую модель шифрования/дешифрования, которая уже была приведена в первой части книги «Основы криптографии», но с использованием других (более подходящих для нашего случая) обозначений: $C = f_e(M)$, где e – ключ шифрования; $M = g_d(C)$, гдеd – ключ дешифрования; M – сообщение; C – криптограмма; f,g – функции шифрования и дешифрования, соответственно.

Для криптосистем с открытым ключом (КОК) также предполагается выполнение принципа *Керхгоффа* (т.е. нарушителю должно быть известно все, кроме ключа дешифрования *d*).

Сформулируем более точно основные требования, которые предъявляются к криптосистемам с ОК:

1) нахождение пары ключей *e* и *d* должно быть вычислительно простым;

2) при известном ключе шифрования *e* нахождение $C = f_e(M)$ должно быть вычислительно простым;

3) при известном ключе дешифрования *d* восстановление сообщения $M = g_d(C)$ должно быть вычислительно простым;

4) при известном ключе шифрования *e* нахождение *d* должно быть вычислительно сложным (необозримо большим);

5) при известном ключе шифрования *e*, но неизвестном ключе дешифрования *d,* нахождение сообщения *M* по известной криптограмме*C* должно быть вычислительно сложным.

Если условие 4 выполняется, то очевидно, что ключ шифрования можно сделать *открытым (общедоступным*). Отсюда происходит и название данного типа криптосистем.

Для того чтобы показать, что КОК имеет практический смысл, нужно ответить на два вопроса.

а) Можно ли выполнить пять вышеперечисленных требований?

б) Какие преимущества имеет КОК в сравнении с традиционными (симметричными) криптосистемами?

Начнем с ответа на второй вопрос.

1-е преимущество КОК. Основным преимуществом КОК является упрощение процедуры распределения ключей.

Пусть в криптосети (КС) имеется N пользователей и каждый хочет конфиденциально связаться с каждым. Тогда в такой сети нужно иметь $C_N^2 = \frac{N(N-1)}{2}$ ключей, причем ключ для каждой пары должен распределяться секретным образом (рис.1.1, а), и только после этого открыто передается зашифрованное сообщение.

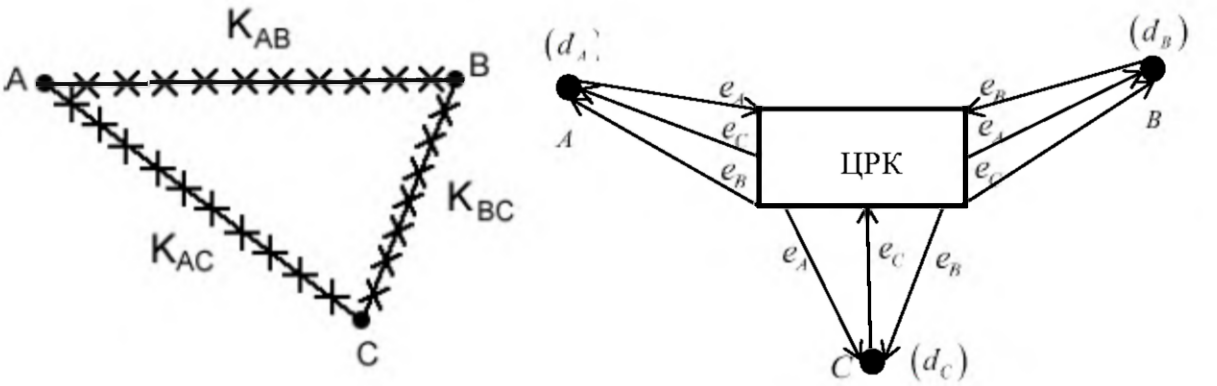


Рис. 1.1. Распределение ключей в различных КС:
а) симметричная КС; б) КОК

В КОК ключ шифрования e передается открыто (или хранится в каком-либо центре распределения ключей (рис.1.1, б), ахранить в секрете необходимо только свой ключ дешифрования d. Ключи других пользователей можно просто при необходимости запросить (рис.1.1, б).

Однако задача распределения ключей полностью не решается с использованием КОК, поскольку здесь возникает проблема *аутентификации* (обеспечения подлинности) открытых ключей. Если эта проблема не решена, то злоумышленник E может выдать себя за легального пользователя, посылая e_E к A и выдавая себя, скажем, за B.

(Решение данной проблемы рассматривается в третьей части книги «Основы криптографии»).

2-е преимущество КОК. Вторым преимуществом КОК перед обычными КС (криптосистемами) является возможность выполнения их пользователями других криптографических функций (цифровая подпись, некоторые протоколы и т.п.), в которых пара взаимодействующих пользователей не обязательно является дружественной.

3-е преимущество КОК. Третьим преимуществом КОК перед обычными КС является свойство «прозрачности» обеспечения стойкости этих криптосистем. Такие системы называются доказуемо секретными. Это означает, что стойкость криптосистемы будет настолько сложна, насколько сложно решение строго определенной математической задачи. В этом плане КОК позволяют доказать их стойкость проще, чем в традиционных симметричных КС.

Для того чтобы положительно ответить на первый вопрос, необходимо привести хотя бы один пример, удовлетворяющий всем пяти требованиям.

Предварительно рассмотрим один иллюстрирующий «механический» пример (рис. 1.2):

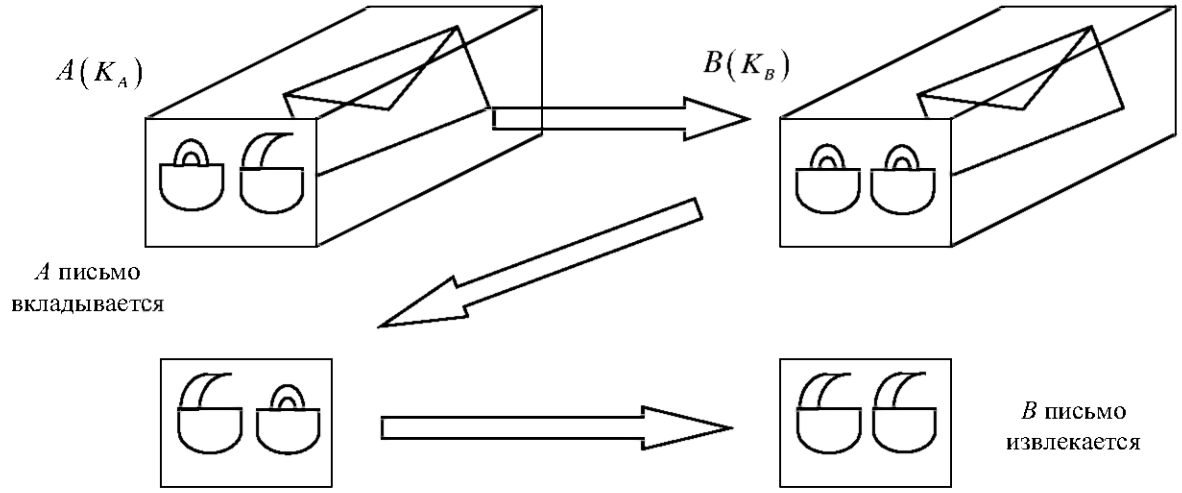


Рис. 1.2. Отсылка письма по почте в сейфе при наличии
у отправителя и получателя разных ключей к сейфу

Видно, что первоначально A вкладывает письмо в сейф и закрывает его на замок своим ключом K_A, оставляя второй замок открытым. Затем A посылает сейф к B. B запирает второй замок своим ключом K_B и возвращает сейф к A. A открывает свой замок ключом K_A и возвращает сейф к B. Наконец, B открывает второй замок своим ключом K_B и извлекает из сейфа письмо.

Данная идея может быть сформулирована в виде некоторого алгоритма, использующего обычную симметричную КС, но удовлетворяющую условию *перестановочности* операций шифрования с разными ключами, т. е., когда:

$$f_{k_A}\left(f_{k_B}(M)\right)=f_{k_B}\left(f_{k_A}(M)\right), \qquad (1.1)$$

где $f_x(y)$ – функция шифрования «y» при ключе «x».

Тогда шифрование без предварительного распределения ключей выполняется следующим образом:

A $\qquad\qquad$ B
(k_A) $\qquad\qquad$ (k_B)

1. $\xrightarrow{C_A=f_{k_A}(M)}$

2. $\xleftarrow{C_B=f_{k_B}\left(f_{k_A}(M)\right)}$

3. $\xrightarrow{C'_A=f^{-1}_{k_A}(C_B)}$ $f^{-1}_{k_B}\left(C'_A\right)=M$,

где $f_x^{-1}(y)$ – это функция дешифрования y на ключе x.

Однако здесь также остается открытой проблема аутентификации.

Во второй части книги предполагается представить криптосистемы, удовлетворяющие требованиям 1–5. Характерной особенностью КОК является то обстоятельство, что алгоритмы шифрования и дешифрования в них, как правило, выполняются как действия по модулю некоторого целого числа или как операции в конечных полях.

Этим КОК отличаются от традиционных криптосистем, для которых более типичными являются обычные булевыеоперации над двоичными переменными.

Для того чтобы понять работу КОК и доказать их стойкость необходимо подготовить некоторый математический базис. Этот базис состоит из основ теории чисел и теории квадратичных вычетов.

2. МАТЕМАТИЧЕСКИЙ БАЗИС КОК

2.1. Основы теории чисел

2.1.1. Представление чисел в различных позиционных системах

Пусть имеется целое число n. Рассмотрим его представление в b-ичной системе:

$$n \to \left(d_{k-1}d_{k-2}...,d_1d_0\right)_2,$$

где $0 \le d_i \le b-1, \quad i = 0, 1, 2, ..., k-1.$

Тогда

$$n = d_{k-1} \cdot b^{k-1} + d_{k-2} \cdot b^{k-2} + d_1 \cdot b + d_0.$$

Пример 2.1. Для целого числа 201 получаем представление: $201 \to (11001001)_2$. Количество разрядов k числа n по основанию b следующее:

$$k = \left[\log_b n\right] + 1 = \left[\frac{\ln n}{\ln b}\right] + 1,$$

где $[x]$ – означает нахождение целой части x. Если $x = 3{,}9$, то $[3{,}9] = 3$, если $x = 3{,}1$, то $[3{,}1] = 3$.

2.1.2. Битовые операции

Рассмотрим сначала *суммирование* двух чисел, заданных в двоичной форме:

	1111	перенос
+	1111000	= 120
	0011110	= 30
	10010110	= 150

Вывод. Для сложения двух k-битовых чисел требуется выполнить k битовых операций (включая сложение с переносом).

Процесс *умножения* в двоичном представлении:

×	1 1 1 0 1	= 29
	0 1 1 0 1	= 13
	1 1 1 0 1	
	0 0 0 0 0	
	1 1 1 0 1	
	1 1 1 0 1	
	0 0 0 0 0	
	1 0 1 1 1 1 0 0 1	= 377

Вывод. Количество битовых операций будет не более k^2, где $k = \left[\log_2 n\right] + 1$, а n – наибольшее из 2-х множителей. Тогда

$$k^2 = \left[\left[\log_2 n\right] + 1\right]^2.$$

Введем важное обозначение. Пусть имеются две функции от натуральных аргументов: $f(n)$ и $g(n)$. Тогда будем писать, что $f(n) = O(g(n))$ или кратко $f = O(g)$, если существуют такие константы C и n_0, что $f(n) \le Cg(n)$ при $n \ge n_0$.

Пример 2.2. Пусть $f(n) = 2n^2 + 3n - 3$ и $g(n) = n^2 \Rightarrow f = O(n^2)$, так как легко показать, что $2n^2 + 3n - 3 \le 3n^2$.

Используя это обозначение, можно записать, что сложность выполнения сложения двух k-битовых чисел будет $O(k)$, а умножения – $O(k^2)$. Иногда сложность выполнения операции называют *временем* ее выполнения, полагая, что одна элементарная операция требует какого-то фиксированного машинного времени.

Считается, что задача требует *полиноминального времени* для своего решения, если существует такое целое число s, что необходимое для решения задачи число операций равно $O(k^s)$, где k – число бит, представляющих исходные данные этой задачи.

2.1.3. Делимость. Алгоритм Евклида

Говорят, что *a делит b*, если существует такое целое число *d*, что $b = a \cdot d$. Можно также сказать, что *a* является *делителем b.*

Каждое число *a* имеет не менее двух делителей: *a* и 1. Если других делителей у него нет, то такое число называется *простым* (*p*), а если они есть, то число называется *составным*.

Основная теорема арифметики [1] гласит, что каждое натуральное (т. е. неотрицательное целое) число *n* может быть представлено как произведение степеней простых чисел, причем это представление единственно, т. е.:

$$n = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot \ldots \cdot p_i^{\alpha_i}, \qquad (2.1)$$

где $\alpha_1, \alpha_2, \ldots, \alpha_i$ – целые числа, а $p_1, p_2, \ldots, p_i$ – простые числа.

Пример 2.3. Для числа 4200 разложение имеет вид $4200 = 2^3 \cdot 3 \cdot 5^2 \cdot 7$. Легко увидеть, что при разложении вида (2.1) общее количество делителей *n* равно: $(\alpha_1 + 1) \cdot (\alpha_2 + 1) \cdot \ldots \cdot (\alpha_i + 1)$.

Определение 2.1.1. *Наибольшим общим делителем* двух чисел *a* и *b* называется такое наибольшее число *d*, которое делит оба этих числа. Наибольший общий делитель двух чисел обозначается обычно как $\gcd(a, b)$.

Пример 2.4. Можно проверить, что $\gcd(12, 15) = 3$. Легко найти наибольший общий делитель, если известно представление чисел в виде степеней простых чисел (2.1).

Пример 2.5. Так, для чисел 4200 и 10780 получим: $4200 = 2^3 \cdot 3 \cdot 5^2 \cdot 7$, $10780 = 2^3 \cdot 5 \cdot 7^2 \cdot 11$, а $\gcd(4200, 10780) = 2^2 \cdot 5 \cdot 7 = 140$.

Казалось бы, данный метод просто решает задачу нахождения gcd, однако разложение чисел на простые множители, т. е. так называемая *задача факторизации* является весьма сложной и неполиномиальной задачей. Означает ли это, что нахождение gcd также является неполиномиальной задачей? К счастью, (для КОК) это не так.

Эта задача решается в полиномиальное время с использованием известного еще с глубокой древности *алгоритма Евклида.* Пусть требуется определить $\gcd(a, b) = ?$, где $a > b$.

Выполним следующие шаги.

1. Разделим *a* на *b* с остатком: $a = b \cdot q_1 + r_1$, где $(r_1 < b)$.

2. Разделим *b* на r_1 с остатком: $b = r_1 \cdot q_2 + r_2$, где $(r_2 < r_1)$.

Продолжим выполнять подобные вычисления до тех пор, пока не получим на некотором этапе, что:

$$r_{n-2} = r_{n-1} \cdot q_{n-1} + r_n; \; r_{n-1} = r_n \cdot q_n.$$

Тогда оказывается, что $\gcd(a,b)=r_n$.

Пример 2.6. Найдем $\gcd(1547,560)$, выполняя шаги алгоритма Евклида:

$$1547=2\cdot 560+427; \quad 560=1\cdot 427+133;$$
$$427=3\cdot 133+28; \quad 133=4\cdot 28+21;$$
$$28=21+7; \quad 21=7\cdot 3;$$
$$\gcd(1547,560)=7.$$

Докажем, что алгоритм Евклида всегда приводит к нахождению gcd *за конечное число шагов.*

Легко заметить, что остатки строго уменьшаются с каждым шагом. Поэтому через конечное число шагов остаток должен стать равным 0, а следовательно, алгоритм всегда закончится после конечного числа шагов, причем последний остаток будет равен gcd. Действительно, если какое-то число делит как *a*, так и *b*, то оно должно делить r_1. Поскольку это число делит *b* и r_1, то оно должно делить r_2 и т. д., вплоть до r_n.

Проходя по этому алгоритму снизу вверх, легко заметить, что последний остаток должен делить все остатки. Тогда $r_n=\gcd(a,b)$, поскольку r_n, с одной стороны делит *a* и *b*, а с другой – сам делится на любой общий делитель *a* и *b*.

Найдем сложность вычисления (время вычисления) $\gcd(a,b)$.

Сначала докажем, что $r_{j+2}<\frac{1}{2}\cdot r_j$. Предположим, что $r_{j+1}\le\frac{1}{2}r_j$, тогда

$$r_{j+2}<r_{j+1}\le\frac{1}{2}r_j. \qquad (2.2)$$

Если же $r_{j+1}>\frac{1}{2}\cdot r_j$, то после следующего деления получаем $r_j=1\cdot r_{j+1}+r_{j+2}$. Отсюда следует $r_{j+2}=r_j-r_{j+1}<r_j-\frac{1}{2}\cdot r_j\le\frac{1}{2}\cdot r_j$, что и требовалось доказать.

Возвращаясь к оценке сложности алгоритма Евклида, мы видим, что за каждые два шага остаток уменьшается не менее, чем в два раза и не может быть меньше 1. Отсюда следует, что алгоритм нахождения gcd потребует не более $2\lceil\log_2 a\rceil$ делений, что дает оценку сложности $2(\log_2 a)_{\text{дел}}\approx O(\log(a))$.

При выполнении каждого деления, числа, участвующие в делении, не могут быть больше *a*, и поэтому число операций каждого деления равно

$O(\log^2 a)$. В итоге получаем, что сложность алгоритма Евклида равна $O(\log^3 a)$.

Замечание. В действительности, если также учесть факт уменьшения чисел на каждом шаге, то сложность можно оценить точнее, а именно, как $O(\log^2 a)$. В любом случае задача нахождения gcd является полиномиально сложной.

Утверждение 2.1.1. Пусть $d = \gcd(a, b)$, $a > b$. Тогда существуют такие целые (но не обязательно положительные) числа *u* и *v*, что:

$$d = u \cdot a + v \cdot b, \tag{2.3}$$

причем сложность нахождения *u* и *v* оценивается как $O(\log^3 a)$.

Доказательство основано на использовании алгоритма Евклида для нахождения наибольшего общего делителя чисел *a* и *b*. Затем по строкам этого алгоритма поднимаемся вверх и получаем необходимое представление (2.3).

Рассмотрим идею доказательства на числовом примере.

Представим $\gcd(1547, 560) = 7$ как линейную комбинацию чисел 1547 и 560. Для этого сначала повторим алгоритм Евклида:

$\gcd(1547, 560) = 7$;

$1547 = 2 \cdot 560 + 427$;

$560 = 1 \cdot 427 + 133$;

$427 = 3 \cdot 133 + 28$;

$133 = 4 \cdot 28 + 21$;

$28 = 21 + 7$;

$21 = 7 \cdot 3$;

$\gcd(1547, 560) = 7$.

Далее, следуя идее алгоритма для нахождения множителей *u* и *v*, получаем:

$$7 = 28 - 1 \cdot 21 = 28 - 1(133 - 4 \cdot 28) = 5 \cdot 28 - 1 \cdot 133 = 5 \cdot (427 - 3 \cdot 133) - 1 \cdot 133 =$$

$$= 5 \cdot 427 - 16 \cdot 133 = 5 \cdot 427 - 16 \cdot (560 - 427 \cdot 1) =$$

$$= 21 \cdot 427 - 16 \cdot 560 = 21 \cdot (1547 - 2 \cdot 560) - 16 \cdot 560 =$$

$$= 21(1547 - 2 \cdot 560) - 16 \cdot 560 = \underset{u}{21} \cdot 1547 - \underset{v}{58} \cdot 560.$$

Определение 2.1.2. Целые числа *a* и *b* называются *взаимно простыми*, если их наибольший общий делитель равен 1. Из доказанного ранее утверждения следует, что для двух взаимно простых чисел *a* и *b* можно представить их $\gcd = 1$ в следующем виде:

$$1 = u \cdot a + v \cdot b,$$

где $a > b$, причем сложность нахождения u и v равна $O\left(\log^3 a\right)$. (Данное свойство будет эффективно использоваться далее для нахождения, так называемых обратных элементов).

Определение 2.1.3. Пусть n – целое натуральное число, тогда *функцией Эйлера* $\varphi(n)$ называется количество целых не отрицательных чисел, меньших n и взаимно простых с n, т. е.:

$$\varphi(n) = \#\left\{0 \le b < n;\ \gcd(b, n) = 1\right\},$$

где $\#\{X\}$ означает количество элементов множества X.

Легко проверить, что $\varphi(1) = 1$, $\varphi(p) = p - 1$, если p – простое число.

Заметим, что вычисление функции Эйлера для произвольного составного числа не является полиноминальной задачей.

2.1.4. Операции по числовому модулю (сравнения, конгруэнтность)

Будем полагать, что $a = b \cdot \bmod m\left(a \overset{m}{\equiv} b\right)$, или говорить, что *a сравнимо* с b по модулю m, если $(a - b)$ делится на m без остатка.

Пример 2.7. Легко проверить, что $5 = 2 \bmod 3$ (также говорят, что *a конгруэнтно b*).

Над сравнениями можно производить обычные операции: сложение, вычитание, умножение, для которых выполняются соотношения:

$$\left.\begin{array}{l} a_1 \overset{m}{\equiv} b_1 \\ a_2 \overset{m}{\equiv} b_2 \end{array}\right\} \quad \begin{array}{l} a_1 + a_2 \overset{m}{\equiv} b_1 + b_2; \\ a_1 \cdot a_2 \overset{m}{\equiv} b_1 \cdot b_2. \end{array}$$

Докажем данное свойство для умножения:

$$\begin{array}{l} a_1 = b_1 + m \cdot t_1 \\ a_2 = b_2 + m \cdot t_2 \end{array}, \quad \text{где} \quad \begin{cases} t_1 - \text{целое} \\ t_2 - \text{целое} \end{cases}.$$

Тогда

$$a_1 \cdot a_2 = b_1 \cdot b_2 + b_1 \cdot m \cdot t_2 + b_2 \cdot m \cdot t_1 + m^2 \cdot t_1 \cdot t_2^2 =$$
$$= b_1 \cdot b_2 + m \cdot \underbrace{\left(b_1 \cdot t_2 + b_2 \cdot t_1 + m \cdot t_1 \cdot t_2\right)}_{N} = b_1 \cdot b_2 + m \cdot N,$$

где N – целое.

Поэтому в модульной арифметике можно рассматривать только наименьшие из чисел, входящих в сравнение, т. е. принадлежащие множеству $(0, 1, 2, ..., m-1)$, которое мы будем обозначать через Z_m. Тогда

$$a + b = res_m(a + b), \quad a \cdot b = res_m(a \cdot b),$$

где $res_m(x)$ – остаток от деления x на m.

Пример 2.8. По приведенным выше правилам

$$(10+13)\bmod 12 = 11, \quad (5\cdot 4)\bmod 3 = 2.$$

Легко проверить следующие *свойства* модульной арифметики:

$$(a+b)\bmod m = (b+a)\bmod m;$$

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

$$-b \bmod m = m - b.$$

Справедливы так же следующие *свойства*:

$$(a+(b+c))\bmod m = ((a+b)+c)\bmod m;$$

$$a\cdot(b+c)\bmod m = (a\cdot b + a\cdot c)\bmod m.$$

Заметим, что в отличии от обычной арифметики, используемой в вычислительной технике, вычисления в модульной арифметике должны быть выполнены абсолютно точно и если эти числа большие, то необходимо использовать специальные методы точных вычислений.

Рассмотрим вопрос обращения элементов в модульной арифметике.

Пусть $a \in Z_m$, тогда *обратным* к a называется $x \in Z_m$, которое удовлетворяет уравнению:

$$x\cdot a = 1\cdot \bmod m. \tag{2.4}$$

Напишем, что $x = a^{-1} \bmod m$.

Оказывается, что обратные элементы существуют не для всех чисел при заданном модуле.

Утверждение 2.1.2. Элемент $a \in Z_m$ имеет обратный элемент $a^{-1} \bmod m$ тогда и только тогда, когда $\gcd(a, m) = 1$.

Доказательство. Предположим противное, т. е. $d = \gcd(a, m) > 1$. Тогда покажем, что решения уравнения (2.4) не существует ни для одного целого $x \in Z_m$.

Действительно, если все же существует $x = a^{-1} \bmod m$, то это означает, что $ax - 1 = mt$. Так как d делит a и m, то d должно делить и «1», что невозможно при $d > 1$.

Предположим теперь, что $\gcd(a, m) = 1$, тогда согласно утверждению 2.1.1 существуют такие целые числа u и v, что: $1 = u\cdot a + v\cdot m$. Покажем, что $a^{-1} \bmod m = u$. Действительно, $u\cdot a - v\cdot m = 1$, где $v \geq 0$, но тогда

$$u\cdot a = v\cdot m + 1 \Rightarrow res_m(ua) = 1 \Rightarrow u = a^{-1} \bmod m.$$

Таким образом, в случае выполнения необходимого условия $\gcd(a, m) = 1$ обратный элемент существует, и он может быть найден при помощи алгоритма для нахождения чисел u и v (со сложностью $O(\log^3 a)$).

Пример 2.9. Можно проверить самостоятельно, что справедливо выражение $160^{-1} \bmod 841 = 205$. Очевидно, что если модуль $m = p$ (простое число), то для любого элемента $a \in Z_m$, существует обратный элемент, так как всегда $\gcd(a, p) = 1$. В этом случае говорят, что это множество образует конечное поле. (Подробнее об этом в подразд. 3.1.9, первой части данной книги.)

Легко заметить, что деление двух чисел по модулю заданного числа выполняется следующим образом: $a : b \bmod m = a \cdot b^{-1} \bmod m$.

Таким образом, если *m* простое число, то для всех чисел из множества Z_m существует полный набор действий, т. е. сложение, вычитание, умножение и деление.

2.1.5. Возведение в степень по модулю

Эта операция записывается следующим образом: $c = b^n \bmod m$, $b \in Z_m$, $n \geq 0$. Простейший способ выполнения возведения в степень состоит в выполнении последовательных умножений: $c = \underbrace{(b \cdot b \cdot \ldots \cdot b)}_{n} \bmod m$.

Однако если *n* достаточно велико, то данный способ не эффективен.

Быстрый способ возведения в степень. Представим *n* в двоичной форме $n = n_0 + 2 \cdot n_1 + 4 \cdot n_2 + \ldots + 2^{k-1} \cdot n_{k-1}$, где $n_i = 0,1$, k – количество двоичных разрядов в представлении *n*. Тогда

$$c = a^n \bmod m = a^{\sum_{i=0}^{k-1} n_i 2^i} \bmod m = \prod_{i=0}^{k-1} \left(a^{2^i} \right)^{n_i} \bmod m.$$

Откуда следует, что данный метод требует вычисления произведения, состоящего из *k* сомножителей, где каждый сомножитель представляет собой степени квадрата *a*.

В случае выполнения такого *быстрого возведения в степень,* оценка сложности (в числе битовых операций) может быть выражена как

$$O\left(\log_2 n \cdot \log^2 m \right)_k.$$

Таким образом, можно полагать, что даже для весьма больших показателей степени (т.е. для *n* имеющих большое количество разрядов) абсолютно точное возведение числа в степень по модулю возможно на обычном ПК.

2.1.6. Вычисление дискретного логарифма

Дискретный логарифм целого числа c по основанию b и по модулю целого числа m определяется следующим образом: это такое $x \in Z_m$, что $b^x = c \bmod m$. Тогда следует написать: $x = \log_b c \bmod m$. Такой логарифм называется также *логарифмом Зеча.*

Оказывается, что дискретный логарифм существует не всегда, например: $\log_3 15 \bmod 17 = 6$, а $\log_3 7 \bmod 13 =$ не существует. (Проверьте это!)

Для решения задачи нахождения дискретного логарифма можно применить метод перебора, т. е. перебирать целые числа, меньшие модуля, проверяя выполнение уравнения, приведенного выше, однако, при больших величинах модуля эта задача оказывается обозримой.

В подразд. 3.1.4 мы приведем оценку числа операций, необходимых для вычисления дискретного логарифма, из которой следует, что такая задача не является полиномиальной. Поэтому она относится к вычислительно трудным задачам и этот факт, как будет показано далее, является основой для построения ряда криптосистем с открытым ключом.

2.1.7. Малая теорема Ферма

Если p – простое число и p не делит a, то: $a^{p-1} = 1 \bmod p$.

Доказательство. Заметим, что $0, a, 2a, ..., (p-1)a$ различны по $\bmod p$. В противном случае, если предположить, что $i \cdot a = j \cdot a \bmod p$, при $i \neq j$, то $(i-j)a = 0 \bmod p$ и поэтому $p \mid (i-j)a$. Но поскольку p не делит a и $i, j < p$, сделано неверное предположение и тогда числа $0, a, 2a, ..., (p-1)a$ составляют всего лишь перестановку чисел $1, 2, ..., p-1$. Поэтому справедливо следующее равенство:

$$a \cdot 2a \cdot ... \cdot (p-1)a = a^{p-1}(p-1)! \bmod p = (p-1)! \bmod p.$$

Отсюда следует, что $(p-1)!(a^{p-1}-1) = 0 \bmod p$, и это приводит к условию $p \mid a^{p-1} - 1$ и поэтому $a^{p-1} - 1 = 0 \bmod p$. Последнее равенство равносильно утверждению малой теоремы Ферма: $a^{p-1} = 1 \bmod p$.

2.1.8. Теорема Эйлера (обобщение малой теоремы Ферма)

Если $\gcd(a, m) = 1$, *то* $a^{\varphi(m)} = 1 \bmod m$, *где* $\varphi(m)$ – *функция Эйлера*[1,2].

Теорема Ферма это частный случай теоремы Эйлера. Действительно, если $m = p$ – простое число, то поскольку $\varphi(p) = p - 1$, то по теореме Эйлера $a^{p-1} = 1 \bmod p$, что и дает утверждение теоремы Ферма.

2.1.9. Свойство мультипликативности функции Эйлера

Если $\gcd(m, n) = 1$, *то* $\varphi(m \cdot n) = \varphi(m) \cdot \varphi(n)$ [1,2].

Так как целое число n может быть представлено однозначно в виде степеней простых чисел, т. е.: $n = p_1^{n_1} \cdot p_2^{n_2} \cdot \ldots \cdot p_r^{n_r}$, где p_1, p_2 ,..., p_r – простые числа, $n_1, n_2, \ldots, n_r$ – целые числа,то из свойства мультипликативности и из того, что

$$\varphi(p^n) = p^n \left(1 - \frac{1}{p}\right)$$

(это нетрудно проверить) следует

$$\varphi(n) = p_1^{n_1}\left(1 - \frac{1}{p_1}\right) \cdot p_2^{n_2}\left(1 - \frac{1}{p_2}\right) \cdot \ldots \cdot p_r^{n_r}\left(1 - \frac{1}{p_r}\right) = n\prod_{p|n}\left(1 - \frac{1}{p}\right). \quad (2.5)$$

(Таким образом, функция Эйлера легко вычисляется для такого числа «*n*», которое имеет известную факторизацию.)

Утверждение 2.1.3. (Полезное для ускорения вычисления степени по модулю).

Если $\gcd(a, m) = 1$, $n' = n \bmod \varphi(m)$, *то* $a^{n'} \bmod m = a^n \bmod m$ [3].

Утверждение 2.1.4. (Полезное для анализа стойкости некоторых криптосистем с открытым ключом). *Пусть* $n = p \cdot q, p \neq q$, *где* $p \cdot q$ – *простые числа, тогда числа* p *и* q *можно найти, если известно* n *и* $\varphi(n) = (p-1) \cdot (q-1)$

.

Доказательство. Рассмотрим p, q как пару неизвестных целых чисел, для которых задано их произведение $p \cdot q = n$ и известна сумма, поскольку $p + q = n + 1 - \varphi(n) = p \cdot q + 1 - (p-1) \cdot (q-1) = 2b$, где b некоторое целое число.

Два числа, сумма которых равна $2b$, а произведение равно n, являются, очевидно, корнями уравнения $x^2 - 2bx + n = 0$ (теорема Виета). Тогда корни квадратного уравнения и есть необходимые нам числа p и q:

$$p = b + \sqrt{b^2 + n^2};$$

$$q = b - \sqrt{b^2 + n^2}.$$

2.1.10. Китайская теорема об остатках

Пусть $\gcd(m_i, m_j)=1$ *для* $i \neq j$. *Тогда система уравнений*

$$\left.\begin{aligned} x &= a_1 \bmod m_1 \\ x &= a_2 \bmod m_2 \\ &\vdots \\ x &= a_r \bmod m_r \end{aligned}\right\} \qquad (2.6)$$

имеет решение, и при этом, если два числа x' *и* x'' *– это решения данной системы, то они удовлетворяют уравнению:*

$$x' = x'' \bmod M, \qquad (2.7)$$

где $M = m_1 \cdot m_2 \cdot \ldots \cdot m_r$.

Доказательство. Докажем однозначность решения по $\bmod M = m_1 \cdot m_2 \cdot \ldots \cdot m_r$. Предположим, что есть два решения системы (2.6): x' *и* x''.Обозначим $y = x' - x''$, тогдаy – удовлетворяет системе:

$$\left.\begin{aligned} y &= 0 \bmod m_1 \\ y &= 0 \bmod m_2 \\ &\vdots \\ y &= 0 \bmod m_r \end{aligned}\right\} \Rightarrow y = 0 \bmod M,$$

так как $m_1, m_2, \ldots, m_r$ – взаимно простые. Отсюда следует, что $x' = x'' \bmod M$.

Покажем теперь как сконструировать хотя бы одно решениеx.

Обозначим $M_i = \dfrac{M}{m_i}$. Очевидно, что $\gcd(m_i, M_i)=1$, поэтому существует обратный элемент N_i к M_i по $\bmod m_i$, т. е.$M_i^{-1} = N_i$, $M_i \cdot N_i = 1 \bmod m_i$, который может быть найден по алгоритму Евклида для нахождения обратных элементов.

Положим теперь, что

$$x = \sum_{i=1}^{r} a_i M_i \cdot N_i .$$

Данное значение будет решением системы (2.6). Действительно, так как m_i делит M_j, $i \neq j$, видно, что все слагаемые будут равны нулю по $\bmod m_i$, за исключением i-го слагаемого. Тогда получаем: $x = a_i M_i N_i \bmod m_i$

и поэтому $x = a_i \bmod m_i$, при $i = 1, 2, \ldots, r$, т.е. x – решение системы (2.6).

2.1.11. Свойства делимости для некоторых представлений чисел

1. Пустьn, b – произвольные целые числа, тогда

$$\left(b^n-1\right):\left(b-1\right)=b^{n-1}+b^{n-2}+...+b+1.$$

2. Пусть b, m, n – целые числа. Тогда:

$$b^{mn}-1=\left(b^m-1\right)\left(b^{m(n-1)}+b^{m(n-2)}+...+b^m+1\right).$$

Эти утверждения легко доказываются, что может быть предложено в качестве упражнения для закрепления материала данного раздела.

2.2. Квадратичные вычеты и тестирование простых чисел

2.2.1.Квадратичные вычеты

Корни из единицы. Пусть задано конечное поле $GF\left(q\right)$, $q=p^n$. Возникает вопрос – сколько корней m-й степени из 1 существует в этом поле? Т. е. нужно решить уравнение: $x^m=1$, где m – любое целое число. Ответ дает следующее утверждение [2,3].

Утверждение 2.2.1. Число корней m-й степени из 1 в поле $GF\left(q\right)$, $q=p^n$ равно $\gcd\left(m,q-1\right)$. Отсюда следует, что если $\gcd\left(m,q-1\right)=1$, то 1 является единственным корнем m-й степени из 1 в этом поле.

Квадратичные вычеты. Рассмотрим простые поля $GF\left(p\right)$, где p – простое число, а $GF\left(p\right)$ состоит из элементов: $\left(0,1,2,3,...,p-1\right)$. Предположим, что $p>2$.

Ставится вопрос – какие из элементов этого поля являются квадратами этих или других элементов данного поля?

Определение 2.2.1. Если $a\in GF\left(p\right)$ является квадратом некоторого элемента $b\in GF\left(p\right)$, т. е. $a=b^2$, $b\in GF\left(p\right)$, то такой элемент поля называется *квадратичным вычетом*. Остальные элементы поля, не представимые в таком виде, называются *квадратичными невычетами*.

Пример 2.10.Если $p=11$, вычетами в таком поле являются 1, 4, 9, 5, 3, так как $1^2=1$, $2^2=4$, $3^2=9$, $4^2=5$, $5^2=3$. Элементы 2, 6, 7, 8, 10 (как легко проверить) будут невычетами.

Если записать элементы поля $GF(p)$ как степени примитивного элемента $\alpha, \alpha^1, \alpha^2, \alpha^3, ..., \alpha^{p-1} = 1$, то в этом случае квадратичные вычеты имеют вид: α^j, где j – четное число.

Чтобы определить, является ли элемент $a \in GF(p)$ квадратичным вычетом, используются *символы Лежандра*.

Определение 2.2.2. Символом Лежандра числа a и простого числа p называется:

$$\left(\frac{a}{p}\right) = \begin{cases} 0, \text{ если } p \mid a; \\ +1, \text{ если } a \text{ квадратичный вычет в } GF(p); \\ -1, \text{ если } a \text{ невычет в } GF(p). \end{cases}$$

Утверждение 2.2.2. Символ Лежандра может быть найден по формуле

$$\left(\frac{a}{p}\right) = a^{(p-1)/2} \bmod p \text{ [2,3].}$$

Однако данный метод не позволяет найти квадратный корень из a по $\bmod p$, даже если известно, что a – вычет.

Известен алгоритм решения задачи $\sqrt{a} \bmod p$, если найдено некоторое другое число $b \in GF(p)$, которое дает $\left(\frac{b}{p}\right) = -1$, т. е. b – невычет. Хотя сейчас не известен полиномиально сложный алгоритм, решающий задачу нахождения невычета, однако, с вероятностью 50 %, при случайном выборе элемента $b \in GF(p)$ мы будем попадать на невычет. Поэтому несколько попыток случайного выбора b с высокой вероятностью даст невычет.

Имея в своем распоряжении метод генерирования невычетов b, можно использовать следующую конструкцию для нахождения $\sqrt{a} \bmod p$ [2,3].

1) необходимо генерировать случайные числа $b \in Z_p$, $Z_p = \{0, 1, 2, 3, ..., p-1\}$ до тех пор, пока $b^2 - 4a$ не окажется квадратичным невычетом по $\bmod p$, т. е. $\left(\frac{b^2 - 4a}{p}\right) = -1$;

2) найти $r = x^{(p+1)/2} \bmod (x^2 - bx + a)$ при вычислении над коэффициентами в поле $GF(p)$;

3) выдать ответ: $r, -r$ – как решения задачи $\sqrt{a} \bmod p$. Сложность нахождения $\sqrt{a} \bmod p$ составляет $O((\lg p)^3)$ битовых операций. Если

$p = 3 \bmod 4$, то нахождение $\sqrt{a} \bmod p$ может быть значительно упрощено. Для этого следует:

1) вычислить $r = a^{(p+1)/4} \bmod p$, используя быстрый алгоритм вычисления степеней;

2) выдать в качестве ответа $(r, -r)$.

Действительно $a^{\frac{(p-1)}{2}} = 1 \bmod p$ как символ Лежандра для a, поскольку существует $\sqrt{a} \bmod p$.

Умножая обе части предыдущего равенства на a, получаем:

$$a \cdot a^{\frac{(p-1)}{2}} = a \bmod p \Rightarrow \left(a^{\frac{(p+1)}{4}} \right)^2 = a \bmod p \Rightarrow a^{\frac{p+1}{4}} = \sqrt{a} \bmod p .$$

Для случая, когда n составное число, нахождение $\sqrt{a} \bmod n$ является весьма трудной задачей и до сих пор не известно ни одного полиномиально сложного алгоритма ее решения. Доказано, что по сложности эта задача эквивалентна задаче факторизации чисел. Если же $n = p \cdot q$, причем p и q известны, то задача извлечения $\sqrt{a} \bmod n$ решается довольно просто по алгоритму, аналогичному рассмотренному выше. Данный факт широко используется в криптосистемах с открытым ключом.

2.2.2. Генерирование простых чисел

При генерировании простых чисел можно говорить о генерировании простых и неслучайных чисел, либо о генерирование простых чисел, являющихся частью секретного ключа и тогда их необходимо каждый раз генерировать заново и случайным образом. Типичным для решения этих задач является подход, состоящий из двух этапов:

1) генерирование случайного или псевдослучайного (если число не является секретным ключом) числа, которое по размерности удовлетворяет предъявленным требованиям;

2) проверка, является ли выбранное нечетное число простым. Если является, то оно принимается. Если же это число не является простым, тогда нужно повторить пункты 1 и 2 до успешного результата.

Возникает вопрос, сколько потребуется сделать попыток, в среднем, для генерирования простого числа заданной размерности?

Ответом на данный вопрос является следующая теорема:

Теорема.[5]. Пусть $\Pi(n)$ – число простых чисел, которые $\le n$, тогда $\lim\limits_{n \to \infty} \dfrac{\Pi(n)}{n / \ln n} = 1$.

Из этой теоремы можно получить аппроксимацию доли нечетных l-разрядных простых чисел, как: $\frac{2}{l\ln 10}$. (Для доказательства этого факта достаточно лишь заметить, что количество в точности l-разрядных нечетных чисел равно $\left(10^l - 10^{l-1}\right)/2$). Тогда среднее число попыток $\overline{s}$ для генерирования одного lразрядного простого числа будет, очевидно, равно $\frac{l \cdot \ln 10}{2}$.

Пример 2.11. Пусть $l = 100$, тогда

$$\overline{s} = \frac{l \cdot \ln(10)}{2} = \frac{100 \cdot \ln(10)}{2} = 115.$$

2.2.3. Важнейшие тесты по проверке простоты чисел

Все тесты делятся на *детерминированные* и *вероятностные*. Детерминированные тесты дают определенный ответ на вопрос о том, является ли данное число простым или составным. Случайные (вероятностные) тесты дают такой же ответ, но с некоторой вероятностью(обычно близкой к 1) того, что ответ будет правильным.

До недавнего времени (до 2002 г.) не было известно ни одного детерминированного алгоритма с полиномиальной сложностью. В 2002 г. три индийских математика нашли такой метод [6]. Его сложность оказывается равной $O\left((\log n)^{12}\right)$, хотя для специальных чисел, вида $2p+1$ сложность оказывается значительно меньше, а именно $-O\left((\log n)^6\right)$. Тем не менее, ввиду значительной сложности этого алгоритма, предпочтение отдается вероятностным алгоритмам, удовлетворяющим следующему условию: если n – простое, то оно всегда *проходит тест* (т. е. то, что оно простое определяется однозначно), если же оно составное, то может случиться, что оно пройдет тест на простоту, однако вероятность такого события может быть сделана сколь угодно малой. Рассмотрим далее два важнейших примера подобных алгоритмов тестирования чисел на простоту.

Тест Ферма. Вспомним малую теорему Ферма, которая гласит, что если p – простое число и p не делит a, то $a^{p-1} = 1 \bmod p$.Выполним тогда следующие шаги:

1) сгенерируем тестируемое число n и выберем параметр «секретности» t;

2) сгенерируем случайное число a_1: $2 \le a_1 \le n-1$, $n \nmid a_1$;

3) вычислим $r = a_1^{n-1} \bmod n$;

4) если $r \neq 1$, тогда n – составное число;

5) если $r = 1$, тогда переходим к шагу 2 и повторяем все то же самое с числом a_2 и т. д., вплоть до повторения t шагов. Если мы получаем

$$a_1^{n-1} = 1 \bmod n$$
$$a_2^{n-1} = 1 \bmod n$$
$$\vdots$$
$$a_t^{n-1} = 1 \bmod n,$$

то считаем nпростым числом.

Однако данный тест может привести к ошибке, когда на одном или на всех шагах это условие выполняется, но число n, тем не менее, является составным.

Пример 2.12.Пусть $n = 341 = 11 \cdot 31 \Rightarrow$(это легко проверить) $2^{340} \bmod 341 = 1$.

Случаи, когда для любых чисел $a_1, a_2, ..., a_t$ составное число n – проходит тест, являются особыми. Такие числа n – называются *числами Кармайкла*, при условии, что $\gcd(a, n) = 1$. Наименьшее число Кармайкла это число $n = 561 = 3 \cdot 11 \cdot 17$. Числа Кармайкла встречаются, однако, довольно редко. Можно доказать [3], что для того чтобы составное число n было бы числом Кармайкла, необходимо и достаточно, чтобы, во-первых, оно не делилось бы на квадрат некоторого простого числа, и, во-вторых, чтобы $p-1$ делило бы $n-1$ для каждого простого делителя p числа n.

Утверждение 2.2.3. При использовании теста Ферма, если число n не является числом Кармайкла, вероятность ошибки тестирования будет равна $1/2^t$, где t – число шагов. Таким образом, выбирая параметр «секретности» t достаточно большим, можно обеспечить высокую надежность тестирования простых чисел.

Тест Миллера-Рабина. Пусть заданы тестируемое нечетное число n и параметр «секретности» t.

Данный тест базируется на утверждении, доказываемом в теории чисел [2, 3].

Утверждение 2.2.4. Пусть $n = p$,нечетное простое число и пусть для него справедливо представление: $n-1 = 2^s \cdot r$, где s, r – числа, причем r – нечетное. Пусть a – такое, что $\gcd(a, n) = 1$, тогда или $a^r = 1 \bmod n$, или $a^{2^j \cdot r} = 1 \bmod n$, где j: $0 \le j \le s-1$.

С учетом данного утверждения, тест Миллера-Рабина выполняется следующими шагами:

1) представить $n-1$ в виде $2^s \cdot r$, где r – нечетное число;

2) сгенерировать случайное число a, такое что $2 \le a \le n-2$;

3) вычислить $y = a^r \bmod n$.

а) если $y = \pm 1$, тогда n прошло тест и возможно является простым (следует повторить этот тест для другого случайно выбранного числа a);

б) если $y \ne \pm 1$, то вычисляются $y^2 \bmod n$, $y^4 \bmod n$, $\ldots y^{2j}$ для $j < s$ до тех пор, пока не получится -1 для некоторого j. Если это происходит, нужно повторить тест для следующего a;

г) если ни при каких j не выполняется шаг 3б, то число n – составное и оно отбрасывается, как не прошедшее тест. Доказывается [2,3], что вероятность ошибки при использовании теста Миллера-Рабина $\approx \frac{1}{4^t}$, где t – число шагов, что значительно лучше, чем для теста Ферма.

Видно, что все операции этого теста имеют полиномиальную сложность.

3. ПОСТРОЕНИЕ КРИПТОСИСТЕМ С ОТКРЫТЫМ КЛЮЧОМ

3.1. Криптосистема РША (Райвеста-Шамира-Адлемана)

3.1.1. Метод формирования пар открытых/закрытых ключей для КС РША

Каждый пользователь КС РША, допустим A, выполняет следующие операции для формирования пары ключей:

а) генерирует пару простых чисел p и q $p \ne q$;

б) вычисляет $n = p \cdot q$ и функцию Эйлера $\varphi(n) = (p-1)\cdot(q-1)$;

в) генерирует e, где $1 \le e \le \varphi$, такое что $\gcd(e, \varphi) = 1$;

г) находит число $d = e^{-1} \bmod \varphi$, т. е. решение уравнения $e \cdot d = 1 \bmod \varphi$;

д) выбирает числа e, n, как свой открытый ключ, а d, как свой секретный ключ.

Как было показано ранее, описанные выше операции могут быть выполнены достаточно быстро даже для чисел n, имеющих 100 и более десятичных разрядов.

Действительно операции выполняются следующим образом:

а) случайным генерированием чисел и тестированием на простоту Миллера-Рабина;

б) обычным умножением, что требует $O\left((\log p)^2\right)$ битовых операций;

в) с использованием алгоритма Евклида;

г) с помощью расширенного алгоритма Евклида для нахождения обратного элемента.

Вся процедура генерирования пар ключей выполняется один раз на длительное время действия этих ключей. (В некоторых стандартах на шифр РША (например, [31]) при вычислении d используется не функция Эйлера $\varphi(n)$, а функция Кармайкла $\lambda(n) = lcm(p-1, q-1)$, где $lcm(x, y)$ – общее наименьшее кратное чисел x и y [32]).

3.1.2. Шифрование в КС РША

Предположим, что пользователь B хочет передать пользователю A сообщение M в зашифрованном виде. Для этого он выполняет следующие действия:

а) B получает подлинный открытый ключ $A\left(e_A, n_A\right)$;

б) преобразует сообщение M в последовательность целых чисел $M_1, M_2, ..., M_i, ...$ не превосходящих $\left(n_A - 1\right)$;

в) для каждого числа M_i, соответствующему части сообщения, формирует криптограмму c_i по правилу $c_i = M_i^{e_A} \bmod n_A$ и отправляет результат пользователю A.

Видно, что процедура шифрования может быть выполнена достаточно быстро при использовании алгоритма быстрого возведения в степень по модулю.

3.1.3. Дешифрование в КС РША

Пользователь A для дешифрования предназначенной ему криптограммы c_i выполняет следующие операции с использованием своего секретного ключа d_A:

а) дешифрует $M_i = c_i^{d_A} \bmod n_A$, $i = 1, 2, ...$;

б) преобразует число M_i в содержательный вид (форму представления информации).

Видно, что операция дешифрования выполняется достаточно быстро. Докажем, что представленная выше процедура дешифрования действительно позволяет получить истинное сообщение $m = m_i$, которое и было ранее зашифровано.

Доказательство. Поскольку по условию $e \cdot d = 1 \bmod \varphi$, то существует такое число k, что $e \cdot d = 1 + k \cdot \varphi$. Предположим сначала, что $\gcd(M, p) = 1$, тогда по теореме Ферма, имеем:

$$M^{p-1} = 1 \bmod p. \tag{3.1}$$

Возведем обе части (3.1) в степень $k(d-1)$, а затем умножим обе части на M:

$$M^{1+k(p-1)(q-1)} = M \bmod p. \tag{3.2}$$

Рассмотрим показатель степени в левой части равенства (3.2):

$$1 + k \cdot (p-1) \cdot (q-1) = 1 + k \cdot \varphi = e \cdot d$$

и тогда, следовательно,

$$M^{1+k(p-1)(q-1)} = M^{e \cdot d}.$$

Из равенства (3.2) получаем, что $M^{e \cdot d} = M \bmod p$, но, с другой стороны, $M^e = c$ и тогда: $c^d = M \bmod p$, что и требовалось доказать.

Если теперь отказаться от заданного ранее условия, что $\gcd(M, p) = 1$, то обязательно должно выполняться условие, что $\gcd(M, p) = p$, и соотношение (3.2) будет тривиально выполняться, так как обе его части равны 0 по модулю p.

В итоге получаем, чтово всех случаях $c^d = M \bmod p$. Следуя аналогичной технике, можно доказать, что $c^d = M \bmod q$, а поскольку p и q различные простые числа, и, следовательно, они взаимно простые, то по китайской теореме об остатках получаем, что

$$c^d = M \bmod (p \cdot q) \Rightarrow c^d = M \bmod n.$$

Последнее равенство доказывает, что используемая в РША процедура дешифрования, действительно, восстанавливает исходное сообщение.

3.1.4. Стойкость КС РША

Итак, для КС РША мы имеем следующие соотношения, связывающие ключи, сообщение и криптограмму:

$$\begin{aligned} &c = M^e \bmod n; \\ &M = c^d \bmod n; \\ &n = p \cdot q; \\ &e \cdot d = 1 \bmod \varphi; \\ &\varphi = (p-1) \cdot (q-1). \end{aligned} \tag{3.3}$$

Из уравнений (3.3) видно, что система РША может быть вскрыта, если удастся найти p и q, то есть факторизовать n.

Исходя из этого факта, p и q должны выбираться такой большой разрядности, чтобы факторизация числа n потребовала бы необозримо большого времени, даже с использованием всех доступных и современных средств вычислительной техники. В настоящее время задача факторизации чисел не имеет полиномиального решения. Разработаны лишь некоторые алгоритмы, упрощающие факторизацию, но их выполнение для факторизуемых чисел большой разрядности все равно требует необозримо большого времени. Действительно, сложность решения задачи факторизации для одного известного алгоритма можно оценить как $O\left(e^{\sqrt{\ln(n)\cdot\ln\ln(n)}}\right)$ [3].Так, например, если $\ln n = 200$, то число операций будет $\approx 1{,}37\cdot 10^{14}$.

При увеличении числа разрядов n до 1024 и более время факторизации становится совершенно необозримым. В 2009 г. Thorsten Kleinjung [23] факторизовал $n = p\cdot q$ для числа n, состоящего из 768 двоичных разрядов.

Если теперь предположить, что алгоритм факторизации неизвестен, но удалось как-то в полиномиальное время найти секретный ключ d, то можно доказать, что число n можно факторизовать, т. е. найти такие p и q, что $n = p\cdot q$ за полиномиальное время [3]. Действительно, если d известно, то можно найти ed. Открытый и закрытый ключи в КС РША связаны уравнением $ed = 1 \bmod \varphi$, откуда следует, что $ed - 1 = k\cdot\varphi$, где k – некоторое целое число. Представим $ed - 1 = 2^S \cdot t$, где t – нечетное число. Тогда по теореме Эйлера $a^{ed-1} = 1 \bmod n$, если выбрано такое a, что $\gcd(a, n) = 1$.

Рассмотрим число $a^{2^S\cdot t/2} = \sqrt{1} \bmod n = \pm 1 \bmod n$ и подберем такие a, что $a^{2^S\cdot t/2} \neq 1$. Тогда, возможно $n \mid a^{2^S\cdot t/2} - 1$. С другой стороны, $\varphi(n) = \dfrac{ed-1}{k} = \dfrac{2^{S-1}\cdot t\cdot 2}{k} = (p-1)(q-1)$, откуда видно, что или $(p-1) \mid 2^{S-1}\cdot t$, или $(q-1) \mid 2^{S-1}\cdot t$ и тогда $a^{2^S\cdot t} = 1 \bmod p$ или $a^{2^S\cdot t} = 1 \bmod q$. Наконец, из последних равенств получаем, что или $a^{2^{S-1}} - 1 = 0 \bmod p$, или $a^{2^{S-1}} - 1 = 0 \bmod q$ и тогда нахождение $\gcd\left(a^{2^{S-1}\cdot t} - 1, n\right)$ даст p или q, что и решит задачу факторизации.

Если каким-то другим способом удалось в полиномиальное время найти параметр φ, ясно, что КС РША может быть вскрыта. Однако, как было показано в разделе 2.1.9, знание φ позволяет просто факторизовать n, и, следовательно, задача нахождения φ вычислительно эквивалентна задаче факторизации. Тем не менее, для систем РША нельзя строго утверждать, что стойкость этой КС эквивалентна задаче факторизации [3]. Подобное свойство имеет место для иных КС, например, для КС Рабина, которую мы будем рассматриваться далее.

Другой естественной атакой на КС РША является дискретное логарифмирование – эта атака (при известном сообщении) выполняется следу-

ющим образом: $d = \log_c M \bmod n$. Однако задача дискретного логарифмирования по модулю многоразрядных чисел, также относится к трудным в математике и оказывается, что она имеет почти такую же сложность, как и задача факторизации [3].

3.1.5. Побочные атаки на КС РША

Помимо двух основных атак на КС РША: *факторизации* и *логарифмирования*, существует ряд побочных атак, основанных на использовании в этой КС таких параметров или режимов, которые значительно упрощают ее криптоанализ.

Рассмотрим далее некоторые из этих атак, связанных с неправильным выбором параметров КС РША при ее разработке или эксплуатации.

Первая побочная атака

Выбор малой величины открытого ключа – e

(Смысл такого выбора *e* для КС РША состоит в том, что это позволяет ускорить процедуру шифрования). Рассмотрим в качестве примера величину $e = 3$. Предположим, что с таким малым *e* производится шифрование в КС РША, и одно и то же открытое сообщение посылается хотя бы трем разным пользователям, имеющим разные *n*. Покажем, что в этом случае можно восстановить сообщение *M* без знания ключа. Тогда последовательность криптограмм будет выглядеть следующим образом:

$$\begin{aligned} C_1 &= M^3 \bmod n_1; \\ C_2 &= M^3 \bmod n_2; \\ C_3 &= M^3 \bmod n_3. \end{aligned} \qquad (3.4)$$

Весьма вероятно, что будет выполнено условие $\gcd(n_i, n_j) = 1$, при $i \neq j$

.

Вспомним китайскую теорему об остатках, которая гласит, что система уравнений (3.4) имеет общее решение:

$$x = M^3 \bmod (n_1 \cdot n_2 \cdot n_3),$$

которое может быть найдено в полиномиальное время.

Также весьма вероятно, что сообщение $M < n_1 \cdot n_2 \cdot n_3$, тогда дешифрование выполняется тривиальным образом как извлечение обычного кубичного корня из числа *x*, т. е. $M = \sqrt[3]{x}$. (Последняя задача может быть просто решена с использованием итеративного алгоритма Ньютона [30].)

Для защиты от этой атаки необходимо либо не использовать малые *e*, либо не отправлять одни и те же сообщения разным пользователям. Если в этом есть объективная необходимость, то сообщение нужно немного изменить, добавляя, например, в конце его небольшие, но различные случайные числа. Такой метод называется методом «*подсаливания*» сообщения.

Вторая побочная атака

Атака при малом объеме возможных сообщений

Предположим, что количество сообщений ограничено значениями $M_1, M_2, ..., M_r$, где r обозримо. (Это могут быть, например, различные команды – вперед, назад, влево, вправо и т. п.). Тогда сообщение может быть легко расшифровано.

Действительно, пусть криптограмма C перехвачена. Тогда необходимо зашифровать все команды известным открытым ключом и определить ту криптограмму, которая совпадает с принятой C:

$$\left.\begin{array}{l} C_1 = M_1^e \bmod n \\ C_2 = M_2^e \bmod n \\ \vdots \\ C_r = M_r^e \bmod n \end{array}\right\} \stackrel{?}{=} C.$$

Способ борьбы с такой атакой это «*подсаливание*» сообщений (т. е. присоединение к ним небольших цепочек бит, полученных с использованием чисто случайного датчика.)

Третья побочная атака

Малая экспонента дешифрования (d)

При выборе малого d существенно упрощается алгоритм дешифрования («малая», в данном случае, означает, что $d \approx \sqrt{n}$). Полное описание этой атаки довольно сложно (атака Винера – Wiener's attack [3, 5]). Основная идея состоит в том, что сначала выводится равенство

$$\frac{e}{n} = \frac{k}{dg}(1-\delta),$$

где $\delta = \dfrac{p+q-1-\dfrac{g}{k}}{n}$;

d – секретный ключ;

e – открытый ключ шифрования;

n – модуль РША;

p, q – простые числа ($n = p \cdot q$);

k, g –целые числа.

Поскольку левая часть этого равенства известна атакующему, то он может использовать метод усечения конечной непрерывной дроби, представляющей рациональную часть $\dfrac{e}{n}$, если δ достаточно мало для того, чтобы найти k и dg, а затем вычислить секретный ключ d [5].

Четвертая побочная атака

Атака с использованием мультипликативного свойства шифра РША

Из описания КС РША следует, что для любых сообщений M_1, M_2

$$\left(M_1 \cdot M_2\right)^e = M_1^e \cdot M_2^e = C_1 \cdot C_2 \bmod n,$$

где $C_1 = M_1^e \bmod n$, $C_2 = M_2^e \bmod n$.

Это свойство может использовать злоумышленник. Предположим, что злоумышленник E хочет дешифровать сообщение C предназначенное для пользователя A и предположим, что A согласен дешифровать любую другую криптограмму для E кроме C. Тогда E может дешифровать C, действуя следующим образом:

1. E выбирает $x \in Z_n$ и вычисляет $\tilde{C} = C \cdot x^{e_A} \bmod n$. Затем E просит A дешифровать $\tilde{C}$.

2. A выполняет эту просьбу: $\tilde{M} = \tilde{C}^{d_A} \bmod n$ и передает результат злоумышленнику E.

3. Поскольку $\tilde{M} = \tilde{C}^{d_A} = C^{d_A} \underbrace{\left(x^{e_A}\right)^{d_A}}_{x} \bmod n = Mx \bmod n$, то E зная $\tilde{M}$ определяет $M = \tilde{M} \cdot x^{-1} \bmod n$, т. е. дешифрует криптограмму C.

Чтобы защититься от такой атаки следует не дешифровать чужие сообщения. Если все же от этого нельзя отказаться, то после дешифрования надо проверить, что получилось – случайный или осмысленный текст. Если получился случайный текст, то не передавать дешифрованное сообщение E.

Пятая побочная атака

Атака на систему РША, использующую общие модули для нескольких пользователей

Это типичная ситуация при генерировании пар ключей для пользователей некоторым общим для них «Центром распределения ключей» (рис. 3.1).

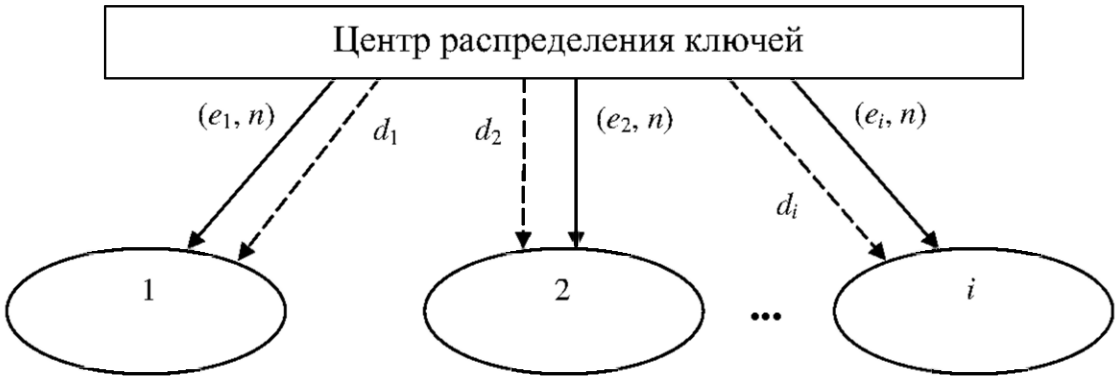


Рис. 3.1. Атака на КС РША, использующую общие модули

В этом случае любой пользователь i, имеющий ключи (e_i, d_i), способен определить любую другую пару ключей. Действительно, знание своего секретного ключа d_i позволяет ему факторизовать n, то есть определить q и p (разд. 3.1.4). Зная же e_j (как открытый ключ другого, j-го пользователя), а также, используя то, что $e_j \cdot d_j = 1 \bmod (p-1)(q-1)$, i-й пользователь может вычислить секретный ключ d_j любого другого пользователя.

Шестая побочная атака

Циклическая атака

Предположим, что известна лишь одна криптограмма C, полученная в криптосистеме РША. Тогда злоумышленник может легко найти ее преобразования:

$$C_1 = C^{e^1} \bmod n;$$

$$C_2 = C^{e^2} \bmod n;$$

$$C_3 = C^{e^3} \bmod n;$$

$$\vdots$$

$$C_r = C^{e^r} \bmod n.$$

Эти вычисления злоумышленник продолжает до тех пор, пока результат не совпадет с исходной криптограммой C. (Это событие должно произойти рано или поздно на каком-то шаге k, так как шифрование – это по существу перестановка чисел $\{0, 1, 2, ..., n-1\}$.) Тогда он может найти сообщение как $M = C^{e^{k-1}} \bmod n$, поскольку $M^e = \left(C^{e^{k-1}}\right)^e = C^{e^k} = C$. Однако в [3] доказывается, что такой метод дает и факторизацию числа n, и, поэтому, при больших n этот подход не лучше прямого метода факторизации модуля КС РША.

Седьмая побочная атака

Отсутствие шифрования

Этот случай возможен, если в результате шифрования мы получаем открытое сообщение, т. е. $M^e \bmod n = M$. Такое условие должно выполниться хотя бы для одного из сообщений, например, для сообщений $M = 0, 1, n-1$. На самом деле существует в точности $\left[1 + \gcd(e-1, q-1)\right] \times$ $\times \left[1 + \gcd(e-1, q-1)\right]$ таких сообщений, которые вообще не шифруются [3].

Их число всегда не менее 9. Однако при случайном выборе q и p доля таких сообщений будет ничтожно мала, и они почти никогда не встретятся

на практике. Кроме того, можно каждый раз проверять не выполнилось ли условие $C = M$ и если да, то «подсаливать» сообщение.

3.1.6. Физические атаки на КС РША

Атака, основанная на нахождении времени вычислений определенных операций

Такие атаки предполагают, что шифрование/дешифрование выполняются в аппаратной форме. Пусть существует некоторый невскрываемый чип дешифрования (рис.3.2).

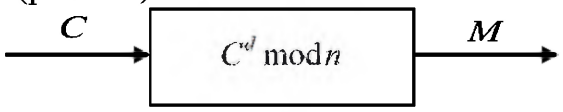


Рис.3.2. Дешифрование при помощи невскрываемого чипа

Напомним, что быстрое возведение в степень состоит в последовательном возведении криптограммы в квадрат и умножениях на криптограмму по модулю n.

Пример 3.1.1. Рассмотрим например дешифрование при помощи возведения в степень 171:

$$C^{171} = \left(\left(\left(\left(\left(\left((C)^2\right)^2 C\right)^2\right)^2 C\right)^2\right)^2 C\right)^2 C,$$

что соответствует выполнению следующих операций возведения в квадрат и умножения:

кв →кв→умнож→кв→кв→умнож→кв→кв→умнож→кв→умнож.

Типично, что операция умножения требует больше времени, чем возведение в квадрат. Если злоумышленник может определить время выполнения последовательности операций при дешифровании в легальном чипе, например, контролируя потребление энергии или излучения в окружающее пространство, то он может восстановить и последовательность действий, а следовательно и секретный ключ d. (Защитой от такой атаки может быть зашумление окружающего пространства, или источников питания специальными шумовыми генераторами, а также нормирование времени выполнения всех операций).

Атака внешним воздействием [5]

(Выполняется при помощи радиационного или электромагнитного излучения).

В некоторых случаях, для сокращения времени, производят вычисления по составляющим модуля с последующим использованием китайской теоремы об остатках, т. е.:

$$M_1 = C^d \bmod p, \quad M_2 = C^d \bmod q,$$

а затем находится

$$M = \left(M_1 \cdot a + M_2 \cdot b\right) \bmod n,$$

при выполнении условия, что:

$$a = 1 \bmod p \text{ и } a = 0 \bmod q;$$

$$b = 0 \bmod p \text{ и } b = 1 \bmod q.$$

Предположим, что в момент вычисления M_1, внутри чипа все происходит правильно, а при вычислении M_2 возникает ошибка (скажем, за счет созданного злоумышленником электромагнитного импульса). Тогда

$$M_1 = C^d \bmod p, \; M_2 \neq C^d \bmod q,$$

и результат вычисления сообщения будет содержать ошибки, т.е.:

$$\tilde{M} = \left(a \cdot M_1 + b \cdot \tilde{M}_2\right) \bmod n.$$

Далее злоумышленник, зная M и $\tilde{M}$ находит:

$$\left(M - \tilde{M}\right) \bmod p = a \cdot M_1 - a \cdot M_1 = 0 \bmod p,$$

$$\left(M - \tilde{M}\right) \bmod q = b \cdot M_2 - b \cdot \tilde{M}_2 = b \cdot \left(M_2 - \tilde{M}_2\right) \bmod q \neq 0 \bmod q.$$

Отсюда следует, что нахождение $\gcd\left(M - \tilde{M}, n\right) = p$ дает нетривиальную факторизацию n, то есть его сомножитель p, поскольку $M - \tilde{M}$ делится на p, но не делится на q, т.е. $\gcd\left(M - \tilde{M}, n\right) \neq n$.

3.1.7. Выбор параметров для КС РША

Как было отмечено ранее, правильный выбор параметров и режимов работы может обеспечить стойкость КС РША для любых существующих вычислительных средств криптоанализа. При этом необходимо обратить внимание прежде всего на выбор величины модуля n и его множителей:

1) уместно выбрать битовую длину $l(n)$ модуля n более 768 (а для обеспечения высокой стойкости $-l(n) \geq 1024$);

2) для того чтобы избежать атак с использованием некоторых специальных алгоритмов факторизации, числа p и q должны быть примерно одинаковой величины, то есть порядка $\sqrt{n}$ (512 бит для высокой стойко-

сти). Однако их разность $(p-q)$ не должна быть малой, поскольку тогда эту разность можно будет подобрать перебором и затем факторизовать n. Иногда рекомендуется использовать так называемые *строго простые числа*р и q, которые удовлетворяют следующим условиям:

а) $p-1$ и $q-1$ имеют большой простой делитель r;

б) $p+1$ и $q+1$ имеют большой простой делитель;

в) $r-1$ имеет большой простой делитель.

Однако последние исследования показывают [3], что эти условия не является необходимыми для обеспечения стойкости, даже относительно всех атак, при выборе $l(n)>1024$;

3) выбор малых экспонент шифрования e ($e=3$ число используется на практике, так как это требует одного возведения в квадрат и одного умножения), вообще говоря, допустим, но для защиты от соответствующей атаки необходимо тогда «подсаливать» сообщения.

Используют также экспоненту шифрования $e=2^{16}+1=65537$, что требует 16 возведений в квадрат и одного умножения. Данный метод, даже без «подсаливания» сообщений, имеет преимущество перед выбором экспоненты $e=3$, поскольку, используемая для малых экспонент, атака будет эффективной, если только *одно и тоже сообщение* шифруется и посылается одновременно… 65537 пользователям, что, конечно, маловероятно;

4) при построении КС для группы пользователей, использующих шифры РША, в случае когда именно центры распределения ключей снабжают их ключевыми данными, необходимо контролировать распределения общих модулей.

3.2. Криптосистема Рабина

3.2.1. Генерирование ключей

Эта процедура выполняется при помощи следующих шагов:

1) необходимо генерировать два простых числаp и q примерно одинаковой разрядности;

2) вычислить $n=p\cdot q$;

3) выбрать в качестве открытого ключаn, а в качестве закрытого – его множители p,q.

3.2.2. Шифрование

Если пользователь *B* хочет зашифровать сообщение *M* для пользователя *A*, то он выполняет следующие шаги:

1) получает открытый ключ*n*пользователя *A*;

2) представляет сообщение *M* в виде последовательности блоков такой длины, что сообщение из каждого блока может быть задано целым числом $M_i \in (0, 1, 2, ..., n-1)$;

3) вычисляет криптограмму $C_i = M_i^2 \bmod n$, $i = 1, 2, ...$

4) отправляет C_i, $i = 1, 2, ...$ пользователю *A*.

3.2.3. Дешифрование

1) Зная *p* и *q*, пользователь *A* реализует полиномиально сложный алгоритм для нахождения $\sqrt[2]{C_i}$ (подразд. 2.2.1), т. е. пользователь *A* находит четыре корня: M_1, M_2, M_3, M_4;

2) используя избыточность, содержащуюся в сообщении, пользователь*A* определяет какое из этих четырех сообщений является истинным. Если в сообщении отсутствует естественная избыточность, то ее необходимо ввести до шифрования (например, повторив несколько последних бит сообщения).

Замечание. Ранее, в подразд.2.2.1 был рассмотрен полиномиально сложный алгоритм для нахождения $\sqrt{a} \bmod p$, где *p* – простое число. Данный метод легко распространяется на случай, когда модуль – это произведение двух простых чисел. Такой модифицированный алгоритм выполняется следующими шагами:

1) вычисляется $\sqrt{a} \bmod p$, т. е. находятся два корня $+r, -r$ по $\bmod p$;

2) аналогично вычисляются корни $+s$, $-s$, по $\bmod q$;

3) используя рассмотренный в разделе 2 алгоритм, находят числа *c* и *d*, такие, что $c \cdot p + d \cdot q = 1$;

4) рассчитывается $x = (r \cdot d \cdot q + s \cdot c \cdot p) \bmod n$,

$y = (r \cdot d \cdot q - s \cdot c \cdot p) \bmod n$;

5) в качестве решения $\sqrt{a} \bmod n$ выбираются числа $\pm x (\bmod n)$, $\pm y (\bmod n)$.

Видно, что схема Рабина обеспечивает простое генерирование ключей, а также весьма простой алгоритм шифрования, но имеет сложный ал-

горитм дешифрования. Поэтому данный метод можно рекомендовать в тех случаях, где шифрование должно быть простым, а при дешифровании этого не требуется.

3.2.4. Стойкость КС Рабина

Ясно, что если злоумышленник сможет факторизовать n, то он становится «легальным» пользователем, однако задача факторизации является сложной и не имеет пока полиномиальных решений. Поэтому выбором, скажем, $l(n)>768$, или для обеспечения большой стойкости $l(n)=1024$, обеспечивается невозможность факторизации. Кроме того, для системы Рабина можно доказать и обратное утверждение.

Теорема [3]. Пусть $n=p\cdot q$, где p и q – простые числа, и пусть существует алгоритм R, который для каждого целого числа C, заведомо являющегося квадратом некоторого числа x по $\bmod n$ (т. е. $x^2=C \bmod n$), находит решение этого уравнения x при помощи $F(n)$ битовых операций. Тогда существует вероятностный алгоритм, который факторизует n со средним числом битовых операций $2\left(F(n)+O\left(\lg^2 n\right)\right)$.

Доказательство. Выберем случайное целое число m, $0<m<n$ и найдем $C=m^2 \bmod n$. Решим уравнение $x^2=C \bmod n$, используя алгоритм R решения при помощи $F(n)$ операций. Обозначим через k найденные решения. Имеется 4 возможности (примем вероятность каждой из них= $\frac{1}{4}$):

а) $k=+m \bmod p$ и $k=+m \bmod q$;

б) $k=+m \bmod p$ и $k=-m \bmod q$;

в) $k=-m \bmod p$ и $k=+m \bmod q$;

г) $k=-m \bmod p$ и $k=-m \bmod q$.

Для решения задачи факторизации подходят только случаи б) и в). Действительно, тогда факторизацию можно выполнить, вычисляя для этих случаев:

б) $\gcd(k-m,n)=p$;

в) $\gcd(k-m,n)=q$.

Итак, при каждом случайном выборе m, мы приходим к возможности факторизации n с вероятностью $\frac{1}{2}$. Так как известно, что сложность нахождения gcd требует $O\left(\lg^2 n\right)$ операций, то, учитывая сложность $F(n)$, выполнения алгоритма R, получаем $2\left(O\left(\lg^2 n\right)+F(n)\right)$ операций необходимых для факторизации.

Таким образом, стойкость КС Рабина строго эквивалентна задаче факторизации и поэтому ее можно назвать *доказуемо секретной криптосистемой.* Заметим, что КС Рабина, так же, как и КС РША, имеет побочные атаки (при малом количестве сообщений, при использовании общих модулей и т.д.)[3].

3.3. Метод распределения ключей Диффи-Хелмана

Недостатком всех КС с открытым ключом является относительная сложность шифрования и дешифрования по сравнению с симметричными КС, что приводит к большим затратам времени при выполнении таких процедур. Это особенно существенно при шифровании больших объемов информации (например, содержащейся на жестких дисках компьютеров или в больших базах данных). Преимуществом же КС с открытым ключом является простой метод распределения ключей. Для объединения положительных свойств симметричных и несимметричных КС используют так называемые *гибридные КС*, в которых распределение ключей осуществляется с помощью несимметричного алгоритма, а собственно шифрование с помощью симметричного алгоритма. Так как обновление ключей требуется сравнительно редко, то гибридная система обеспечивает практически такую же скорость шифрования/дешифрования, как и симметричная КС.

Для распределения ключей можно использовать любую КС ОК, например, РША или Рабина. Тогда гибридная КС будет иметь вид, показанный на рис. 3.3.

На таком принципе построена, например, система PGP, используемая для шифрования электронной почты [7].

Однако для решения задачи только по распределению ключей к симметричным шифрам, можно применить другой ассиметричный алгоритм, называемый распределением ключей *Диффи-Хелмана*.

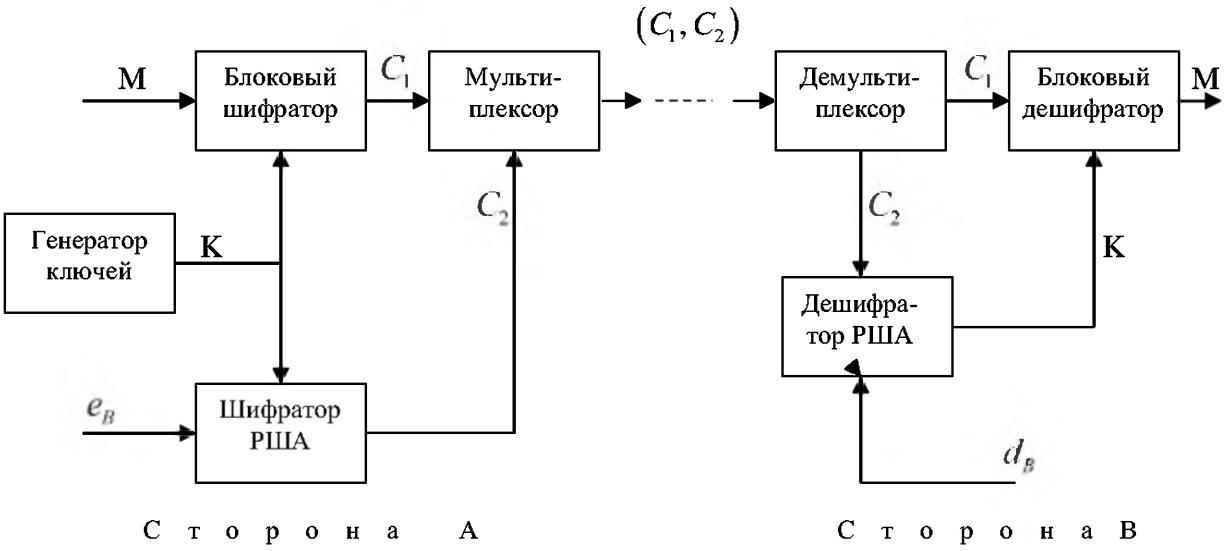


Рис. 3.3. Гибридная криптосистема

Перед выполнением этого алгоритма, пользователи A и B согласуют открытые параметры: примитивный элемент $\alpha \in Z_p$ и p, где p – простое число. Далее выполняется протокол, показанный на рис. 3.4.

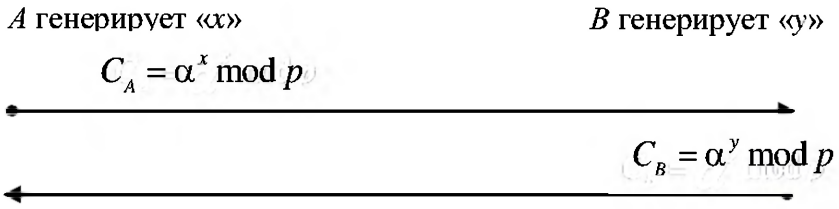


Рис. 3.4. Распределение ключей по методу Диффи-Хелмана

После получения необходимой информации, A иB вычисляют общий ключ K следующим образом:

$$\begin{cases} K_A = C_B^x \bmod p = \left(\alpha^y\right)^x \bmod p = \alpha^{yx} \bmod p, \\ K_B = C_A^y \bmod p = \left(\alpha^x\right)^y \bmod p = \alpha^{xy} \bmod p, \ K = K_A = K_B. \end{cases}$$

3.3.1. Стойкость КС Диффи-Хелмана

Если злоумышленник умеет в обозримое время вычислять дискретный логарифм, то он может найти и секретный ключ A или B, поскольку $x = \log_\alpha C_A \bmod p$. Однако поскольку задача дискретного логарифмирования является трудной, то данный способ при больших величинах p нереализуем.

Вместе с тем, стойкость КС Диффи-Хелмана не строго эквивалентна задаче факторизации, а соответствует так называемой *проблеме Диффи-Хелмана,* которая формулируется следующим образом: *зная* $p, \alpha, \alpha^x \bmod p, \alpha^y \bmod p$, *нужно найти* $\alpha^{xy} \bmod p$.(Впрочем, последняя задача по сложности не намного уступает задаче дискретного логарифмирования).

Важно отметить, что метод распределения ключей Диффи-Хелмана может быть полностью скомпрометирован *активными злоумышленниками*, выдающими себя за пользователей*A* или *B*. Если этот факт подмены (или имитации) величин C_A, C_B не будет обнаружен, то вырабатывается совместный ключ не между *A* и *B*, а между злоумышленником и одним из пользователей. Таким образом, для КС Диффи-Хелмана, так же как и для всех КС ОК, необходимо обеспечивать подлинность открытых данных, т.е. обеспечить решение задач аутентификации.

3.4. КС Эль-Гамаля

Это некоторая модификация КС РША, стойкость которой не связана непосредственно с проблемой факторизации. Она широко применяется в стандартах цифровой подписи и допускает естественное обобщение на случай КС, построенных на основе использования эллиптических кривых, что будет рассмотрено далее.

3.4.1. Генерирование ключей

Пользователь *A* проделывает следующие операции для генерирования ключей:

1) генерирует простое число *p* и примитивный элемент $\alpha \in GF(p)$;

2) выбирает случайное число *a*, такое, что $1 \le a \le p-2$ и вычисляет число $\alpha^a \bmod p$;

3) в качестве открытого ключа выбирается набор: $(p, \alpha, \alpha^a \bmod p)$, а в качестве закрытого ключа число *a*.

3.4.2. Шифрование

Пользователь *B* выполняет следующие шаги для шифрования сообщения *M*, предназначенного пользователю *A*:

1) получает открытый ключ *A*;

2) представляет сообщение *M* в виде цепочки чисел M_i, каждое из которых не превосходит $p-1$;

3) выбирает случайное число *k* такое, что $1 \le k \le p-2$;

4) вычисляет $\gamma = \alpha^k \bmod p$, $\delta = M_i \cdot \left(\alpha^a\right)^k \bmod p$;

5) посылает криптограмму $C = (\gamma, \delta)$ пользователю *A*.

3.4.3. Дешифрование

Пользователь *A* выполняет следующие шаги для дешифрования сообщения, полученного от пользователя *B*:

1) используя свой закрытый ключ, вычисляет $\gamma^{-a} \bmod p$;

2) восстанавливает сообщение $M_i = \gamma^{-a} \cdot \delta \bmod p$.

Действительно, $\gamma^{-a}\delta = \alpha^{-ak} M_i \alpha^{ak} = M_i \bmod p = M_i$.

Особенностью схемы Эль-Гамаля является то, что она относится к так называемым схемам *рандомизационного шифрования*, поскольку при шифровании в ней используется дополнительная случайность в виде числа *k*. (Считается, что рандомизационное шифрование более стойко по отношению к некоторым методам криптоанализа, например,к таким как статистические атаки [3]).

Преимущество КС Эль-Гамаля состоит также и в том, что тогда все пользователи в сети могут выбирать одинаковые α и *p*, что невозможно для КС РША. Кроме того, как будет показано далее, эта схема может быть естественным образом распространена на случай эллиптических кривых.

Существенным недостатком схемы является то, что длина криптограммы в ней в два раза больше длины сообщения.

3.4.4. Стойкость КС Эль-Гамаля

Проблема восстановления сообщения M по заданным $p, \alpha, \alpha^a, \delta$ при неизвестном a, эквивалентна решению задачи Диффи-Хелмана. Действительно, поскольку $\gamma = \alpha^k$ и α^a известны криптоаналитику, то решение проблемы Диффи-Хеллмана позволило бы ему найти α^{ka}, а, следовательно, и α^{-ka}, что, в свою очередь, позволило бы дешифровать сообщение M при известномδ (шаг 2 легитимного дешифрования).

Ясно также, что если будет решена проблема нахождения дискретного логарифма, то криптосистема Эль-Гамаля будет вскрыта. (Однако обратное утверждение пока никем не доказано.) При выборе p с разрядностью 768 бит (для повышенной стойкости –до 1024 бита), стойкость КС Эль-Гамаля будет такой же, как и у КС РША при выборе в последней тех же параметров для модуля.

Важно отметить, что для шифрования различных сообщений M_i, M_j необходимо использовать различные значения чисел k, поскольку в противном случае $\frac{\delta_i}{\delta_j} = \frac{M_i}{M_j}$ и тогда сообщение M_j может быть легко найдено, если известно сообщение M_i.

3.5. Построение КС на основе использования эллиптических кривых

3.5.1.Элементы теории эллиптических кривых над конечными полями

Определение 3.5.1. Пусть $GF(q)$, $q = p^n$ – некоторое конечное поле. Тогда *эллиптической кривой* (ЭК) E над полем $GF(q)$ называется множество пар элементов (x, y), $x, y \in GF(q)$, которые удовлетворяют уравнению:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_5, \qquad (3.5)$$

где $a_1, a_2, a_3, a_4, a_5 \in GF(q)$.

Если $n = 1$, $p > 3$, т.е. $GF(p)$ – простое поле, то уравнение (3.5) приводится к виду:

$$y^2 = x^3 + ax + b. \tag{3.6}$$

Если $p = 2$, то уравнение (3.5) будет иметь следующий вид:

$$y^2 + xy = x^3 + ax^2 + b. \tag{3.7}$$

Далее, будем рассматривать только ЭК над простыми полями $GF(p)$, $p > 3$.

Пример 3.5.1. Пусть нужно проверить, что точка $x = 1$, $y = 4$ лежит на эллиптической кривой $y^2 = x^3 + 2x + 3$ над полем $GF(5)$.Тогда находим:

$$y^2 \bmod 5 = 1;$$
$$x^3 \bmod 5 = 1;$$
$$2x = 2,\ 1 + 2 + 3 \bmod 5 = 1 = 1 \bmod 5.$$

Видим, что уравнение (3.6) удовлетворяется, и, следовательно, точка лежит на кривой.

Легко заметить, что ЭК имеет не более чем $2q + 1$ точек (одна точка находится в бесконечности и для каждого *x* имеется не более двух значений *y*, удовлетворяющих уравнению (3.5). Однако на самом деле их значительно меньше.

Количество точек на ЭК над полем $GF(q)$ можно оценить с помощью *теоремы Хассе*.[5]

Теорема. Пусть *N* – число точек ЭК над полем $GF(q)$. Тогда

$$\left|N - (q - 1)\right| \le 2\sqrt{q}. \tag{3.8}$$

Важным свойством ЭК является то, что между точками этих ЭК можно задать операции *сложения* и нахождения *противоположной точки*.

Множество точек на ЭК образуют так называемую *группу* относительно операций специфического сложения, заданной на ЭК. Далее нам потребуется определение понятия группы.

Определение 3.5.2. Конечной группой *G* называется конечное множество элементов $g \in G$, на котором задано сложение между элементами, противоположные элементы $-g$ и нейтральный элемент 0, удовлетворяющим следующим аксиомам.Если $g, g' \in G$, то:

1) $g + g' \in G$;

2) $g + (g' + g'') = (g + g') + g''$;

3) $g + 0 = g$;

4) $g + (-g) = 0$.

Если $g + g' = g' + g$, то группа называется *Абелевой*.

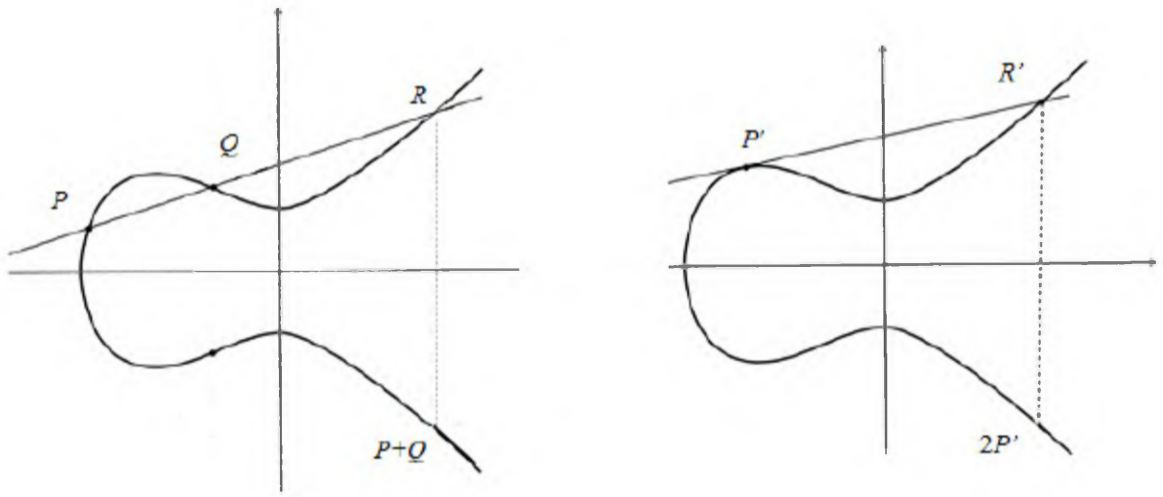
Легко заметить, что в любом конечном поле, рассмотренном ранее в подразд. 3.1.9 первой части книги, операции относительного сложения и умножения образуют конечные Абелевы группы.

Определение операции нахождения противоположного элемента и суммы элементов на ЭК

Пусть ЭК задана уравнением $y^2 = x^3 + ax + b$. Для того чтобы лучше понять, как выполняется сложение точек и нахождение противоположной точки, рассмотрим сначала ЭК, заданные над полем вещественных чисел, которые можно изобразить геометрически, как это показано на рис.3.5.

Для любой точки P ЭК с координатами (x, y) обратная точка $-P$ будет иметь (по определению) координаты $(x, -y)$.

Для двух точек P и Q на ЭК, которые имеют соответственно координаты (x_1, y_1) и (x_2, y_2), причем $x_1 \neq x_2$, сумма этих точек определяется геометрически, как точка, обратная к той, которая получается на пересечении прямой линии, проходящей через точки P,Q и другой ветви ЭК еще в одной точке R (рис.3.5, а).



а) б)

Рис. 3.5. ЭК $y^2 = x^3 - x$ на плоскости:
а) сложение различных точек; б) сложение одинаковых точек

Если $x_1 = x_2$, $y_1 = -y_2$ – это означает, что $P = -Q$.Тогда, по определению, $P + Q = O$, гдеO – точка в бесконечности. Если же $P = Q = P'$, то $P' + P' = -R'$, где R' точка, которая получается как пересечение касательной кЭК в точке P' и другой ветви ЭК (рис.3.5, б).Причем, если касательная в точке P' является «одиночной», то $R' \neq P'$, а если это «двойная» касательная, т. е. когда P' это точка перегиба кривой, то $R' = P'$.

Все эти геометрические построения могут быть заданы в аналитической форме, которая позволяет найти координаты точки противоположной данной, или равной сумме двух точек, заданных на ЭК.

Рассмотрим случай, когда ЭК задается уравнением (3.6). Пусть также $P \neq Q$, где $P(x_1, y_1)$ и $Q(x_2, y_2)$ две разные точки на эллиптической кривой. Тогда для координат точки равной сумме точек P и Q получаем уравнения [2]:

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2;$$
$$y_3 = -y_1 + \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3). \qquad (3.9)$$

Аналогично можно вывести аналитическое уравнение для касательной (используя понятие формальной производной) и затем получить выражения для суммы точек, когда $P = Q$ [2]. Тогда для случая той же эллиптической кривойполучаем для координат точки-суммы выражение:

$$x_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1; \qquad (3.10)$$

$$y_3 = -y_1 + \left(\frac{3x_1^2 + a}{2y_1}\right)(x_1 - x_3). \qquad (3.11)$$

Очевидно, что все операции, заданные в уравнениях для нахождения координат точек суммы, могут выполняться и над конечным полем $GF(p^n)$, что и требуется для построения КС ОК, тогда как эллиптические кривые над полем вещественных чисел и их геометрическая интерпретация были полезны лишь для наглядности.

«Возведение в k-ю степень» точки P на эллиптической кривой понимаетсякак k-кратное сложение этой точки с самой собой на этой кривой:

$$"P^k" = P + P + ...P.$$

Для быстрого вычисления степени точек на эллиптической кривой при больших значениях показателей можно использовать технику быстрого возведения в степень, рассмотренную ранее, которая состояла в двоичном разложении числа k, последующем сложении и удвоении.

Пример 3.5.2. Пусть $k = 171 = 10101011$, $P \in E$ над $GF(q)$. Тогда:

$$2P = P + P;$$
$$4P = 2P + 2P;$$
$$8P = 4P + 4P; \qquad (3.12)$$
$$128P = 64P + 64P;$$

$$171 \cdot P = 2\Big(2\Big(2\Big(2\Big(2\Big(2(2P)+P\Big)\Big)+P\Big)\Big)+P\Big)+P\,.$$

Ранее было дано определение Абелевой группы относительно операций сложения. Из рассмотренных свойств точек, принадлежащих ЭК над полем $GF(q)$, следует, что множество таких точек образует Абелеву группу относительно сложения, причем порядок этой группы (т. е. число ее элементов) не совпадает с порядком поля q. Для многих криптосистем с открытым ключом операции шифрования и дешифрования фактически выполняются на Абелевых группах, и этот факт открывает возможности для построения КС ОК на основе использования эллиптической кривой.

3.5.2. Задание КС над эллиптическими кривыми

Теория ЭК может быть полезной для решения различных задач, связанных с построением КС ОК, а именно для задач:

- тестирования простых чисел;
-факторизации больших чисел;
-вычисления дискретных логарифмов;
-построения новых криптосистем с ОК;
-построения новых систем с цифровыми подписями.

Далее мы рассмотрим только две последние задачи. Описание первых трех можно найти в [2,5,8].

К эллиптическим кривым E, над которыми строятся КС ОК, обычно предъявляются следующие требования [8]:

1) кривые рассматриваются либо над простыми полями $GF(p)$, либо над полями $GF(q)$ характеристики p= 2, когда $q = 2^n$;

2) исходя из соображений секретности, а именно, для того чтобы проблема дискретного логарифмирования над ЭК (см. далее) не решалась бы проще, чем вычисление логарифмов Зеча, необходимо чтобы эллиптическая кривая не была бы *сингулярной*, *суперсингулярной* и *аномальной*. Если эллиптическая кривая выбирается над простым полем $GF(p)$ и удовлетворяет уравнению (3.6), то данные условия выполняются, когда $4a^3 + 27b^2 \neq 0 \bmod p$ и количество точек на ЭК не равно p [3, 27];

3) кривая должна иметь точку высокого порядка (2^{160}) относительно операции сложения на этой кривой. (Отметим, что порядком точки называется ее порядок, как элемента Абелевой группы);

4) в качестве коэффициентов a, b можно выбирать (0,1), но рекомендуется генерировать их случайным образом.

3.5.3. Обобщение КС Эль-Гамаля на случай эллиптических кривых

Для того чтобы создать КС Эль-Гамаля на ЭК, необходимо выбрать достаточно большое поле и задать уравнение этой кривой. Число точек на этой кривой должно быть непереборно велико. Алгоритм случайной генерации кривой с требуемыми свойствами подробно описан в [8].

Генерирование ключей

1) Генерируется точка $\alpha \in E$ большого порядка;

2) выбирается случайное целое число a, такое, что $1 \le a \le q-2$ и вычисляется $\alpha \cdot a$, что означает сложение точки $\alpha \in E$ с самой собой a раз по правилам сложения, задаваемым уравнениям аналогичным (3.9)–(3.11);

3) данные E, $\alpha \in E$, $a \cdot \alpha$ представляют собой открытый ключ, тогда как a – закрытый ключ.

Шифрование

Пользователь B шифрует сообщение M для пользователя A следующим образом:

1) получает открытый ключA, т. е.$\left(E, \alpha, a \cdot \alpha\right)$;

2) представляет сообщение M как точку (или как точки) на ЭК E, для чего можно разместить данные в координате x, а координату y выбрать так, чтобы точка $\left(x, y\right)$ лежала на заданной кривой E;

3) генерирует большое, целое случайное число k;

4) вычисляет $\gamma = k \cdot \alpha$, $\delta = M + k \cdot a \cdot \alpha$;

5) посылает пользователю A криптограмму $C = \left(\gamma, \delta\right)$.

Дешифрование

Используя свой секретный ключ a, пользователь A

1) вычисляет ($-\gamma \cdot a = b$);

2) восстанавливает сообщение $M = b + \delta$.

Стойкость КС Эль-Гамаля, реализованной на ЭК

Определение 3.5.3. Пусть E – эллиптическая кривая, а P – точка на этойкривой и n – целое неотрицательное число. Тогда *проблемой дискретного логарифмирования над заданной* ЭК E называется решение относительно n следующего уравнения

$$nP = Q, \tag{3.13}$$

где P и Q известны.

Ясно, что если криптоаналитик в состоянии решить задачу логарифмирования над ЭК, то КС, описанная выше, будет взломана. Однако реше-

ние этой задачи имеет сложность порядка $n^{\beta}, \beta > 0$, что неизмеримо сложнее, чем вычисление nP при известных n, P.

Более того, для построения стойкой КС Эль-Гамаля над ЭК достаточно выбрать битовую длину цепочки, описывающей n порядка 150–180 бит [8], тогда как для «классической» КС Эль-Гамаля необходимо выбирать для представления модуля p более 768 бит.Именно то обстоятельство, что «логарифмирование» над ЭК значительно сложнее логарифмирования над конечными полями и создает перспективы использования таких кривых в КС ОК.

3.5.4. Построение системы распределения ключей Диффи-Хелмана над эллиптическими кривыми

Предварительно пользователи согласуют между собой эллиптическую кривую E над полем $GF(q)$ и точку P на этой кривой, так же как это делалось при построении КС Эль-Гамаля. Далее пользователь A случайно генерирует целое положительное число n_A, $1 \le n_A \le q-2$ и посылает по каналу связи кB «точку» эллиптической кривой $P_A = n_A P$. В свою очередь, пользователь B случайно генерирует целое положительное число n_B, $1 \le n_B \le q-2$ и посылает по каналу связи к A «точку» эллиптической кривой $P_B = n_B P$. Наконец, A вычисляет ключ $K_A = n_A P_B$, а B – ключ $K_B = n_B P_A$, которые, очевидно, будут совпадать и их можно использовать как общий ключ между A и B.

Стойкость такого метода распределения ключей будет определяться сложностью решения задачи дискретного логарифмирования над ЭК и при соответствующем выборе поля $GF(q)$, самой эллиптической кривой E и точки P на ней, она может быть достаточно высокой.

3.6. КС Мак-Элис

В отличие от предыдущих КС ОК, данная КС использует сложность решения задачи по *исправлению ошибок линейными кодами* для обеспечения ее стойкости. Поэтому потребуется освежить наши знания по корректирующим кодам, которые изучались ранее в курсах «Теории электрической связи» и «Передачи данных». (Заметим, что КС Мак-Элис относится к так называемым «пост-квантовым КС», т. е. к КС с ОК, которые сохраняют свою стойкость даже при реальном появлении «квантовых компьютеров».

Появление таких компьютеров в ближайшие 10 лет представляется весьма проблематичным, однако, даже если это событие произойдет, то, хотя задача факторизации может быть решена ими в полиномиальное время, и следовательно, такие КС как РША или Рабина могут быть взломаны, задача исправления ошибок линейными, случайно выбранными кодами, на которой основана стойкость КС Мак-Элис, не может быть решена в полиномиальное время, даже при использовании квантовых компьютеров. Поэтому подобная КС имеет перспективы применения и в «пост-квантовый период»).

3.6.1. Краткие сведения о линейных кодах [4]

Линейный двоичный (n,k) код V (ЛК) это множество, состоящее из 2^k двоичных последовательностей $\overline{y}$, длины n каждая, для которых справедливо условие: если $\left.\begin{matrix}\overline{y}_1 \in V \\ \overline{y}_2 \in V\end{matrix}\right\}$ то, $\overline{y}_1 \oplus \overline{y}_2 \in V$, где $\oplus$ означает покомпонентное суммирование по *mod*2.

Каждый ЛК может быть однозначно задан своей *порождающей* $k \times n$ *матрицей* G, состоящей из двоичных элементов 0 и 1 (рис. 3.6).

Если G – порождающая матрица кода V и $\overline{x}$ – двоичная информационная последовательность длины k, то кодирование (т. е. преобразование ее в кодовую последовательность $\overline{y}$) производится по правилу:

$$\overline{y} = \overline{x} \cdot G, \tag{3.14}$$

где операции умножения вектора на матрицу выполняются в поле $GF(2)$.

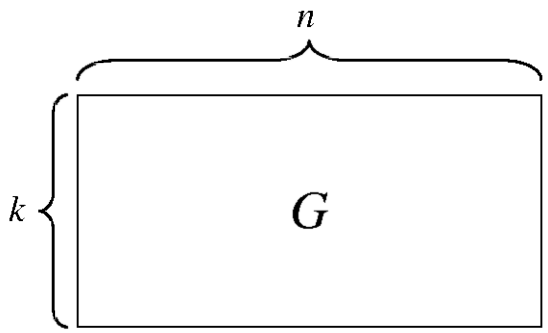


Рис. 3.6. Порождающая матрица линейного кода

Линейный код может быть представлен в систематической форме, когда каждое его слово состоит из *k-информационных* символов $\overline{x}_k$ и следующих за ними $n-k$ *проверочных* символов $\overline{c}_{n-k}$, которые формируются по информационным при помощи линейных операций над полем $GF(2)$. Эти линейные соотношения являются нетривиальной частью так называемой

проверочной матрицы кода, которая однозначно определяется по его порождающей матрице [4].

Если в кодовом слове $\bar{y}$ возникает ошибка $\bar{e}$ (при его передаче по каналу связи или при хранении), т. е. $\tilde{\bar{y}} = \bar{y} \oplus \bar{e}$, (где $\bar{e}$ имеет единицы в позициях с ошибками и нули в остальных позициях, а $\oplus$ означает поэлементное суммирование по *mod*2), то, используя свойства кода, некоторые ошибки можно исправить (т. е. восстановить информационную последовательность $\bar{x}$ по искаженному кодовому слову $\tilde{\bar{y}}$). Исправляющие свойства кода задаются его, так называемым *минимальным кодовымрасстоянием* $d_{\min}$, которое определяется как минимальное число позиций, в которых отличаются любые несовпадающие между собой кодовые комбинации.

Если код имеет минимальное кодовое расстояние $d_{\min}$, то он гарантированно (т. е. полностью) исправляет все ошибки кратности не более $t = \left[\frac{d_{\min}}{2}\right]$, где x – означает целую часть числа x, а кратность ошибки – число искаженных позиций, т. е. число единиц в образце ошибки $\bar{e}$.

В теории кодирования [4] разработаны методы построения ЛК, т. е. фактически порождающихих матриц, которые обеспечивают максимально возможные величины $d_{\min}$ при заданных параметрах (n, k).

Как видно из соотношения (3.14), процедура кодирования для ЛК оказывается достаточно простой. Более того, для некоторых классов ЛК (например, для так называемых *циклических* кодов), она может быть еще более упрощена [4].

Однако процедура декодирования (т. е. исправления ошибок) остается весьма сложной для произвольных линейных кодов. Действительно, если известно, что данный код имеет заданное $d_{\min}$ и произошла ошибка кратности $t \le [d_{\min} / 2]$, то для декодирования достаточно вычислить *расстояния Хэмминга* (т. е. число различных позиций) между принятым искаженным словом $\tilde{\bar{y}}$ и всеми возможными кодовыми словами $\bar{y} \in V$ и принять решение о передаче того кодового слова, для которого это расстояние минимально. Однако число кодовых слов в коде с параметрами (n, k) равно 2^k, и при больших значениях k это число оказывается непереборно большим, т. е. такая процедура является практически нереализуемой. Более того, в теории вычислительной сложности [9] доказывается утверждение, что если бы для любых ЛК был бы найден полиномиально сложный алгоритм исправления ошибок по *минимуму расстояния Хэмминга*, то его можно было бы использовать и для решения множества других трудных задач, которые до сих пор не имеют полиномиально сложных решений. Поэтому задача

нахождения простого алгоритма декодирования для произвольных ЛК является весьма сложной.

Для того чтобы упростить процедуру исправления ошибок в задачах связи используют *подоптимальные алгоритмы* декодирования для подклассов ЛК (например, для так называемых *кодов Гоппы*, для которых сложность декодирования задается соотношением $O\left(n^{2\div 3}\right)$ [3]).

Особенность ЛК, выражающаяся в простоте кодирования и сложности декодирования, где последнее может быть преодолено при переходе к некоторым подклассам ЛК, и используется для построения КС Мак-Элис.

3.6.2. Описание КС Мак-Элис

Если пользователь A хочет сгенерировать свою пару открытый/закрытый ключ, то эта процедура реализуется следующими шагами:

1) генерируется случайная порождающая матрица G_A для специального подкласса двоичных кодов (скажем, для кодов Гоппы), которые гарантированно исправляют не менее t_A ошибок и имеют полиномиально сложный алгоритм декодирования;

2) генерируется случайная двоичная *несингулярная* (т. е. имеющая ненулевой определитель) матрица S_A, размером $k \times k$;

3) генерируется случайная *перестановочная* $n \times n$ матрица P_A. (Перестановочной называется такая матрица, произведение которой на любой вектор дает лишь перестановку его позиций. Такая матрица в каждой строке и в каждом столбце содержит по одной единице, а остальные ее элементы – нули.);

4) вычисляется матрица $\hat{G}_A = S_A \cdot G_A \cdot P_A$ (ее размерность будет $k \times n$);

5) публикуется открытый ключ $\hat{G}_A, t_A$ и сохраняется в тайне секретный ключ $\left(S_A, G_A, P_A\right)$.

3.6.3. Шифрование КС Мак-Элис

Если пользовательB хочет зашифровать сообщение M для пользователя A, то он должен выполнить следующие шаги:

1) получить от A открытый ключ $\left(\hat{G}_A, t_A\right)$;

2) преобразовать сообщение M в последовательность двоичных блоков M_i длины k.

Далее для каждого из полученных блоков необходимо проделать следующие операции:

1) сгенерировать случайный двоичный вектор Z_i, длины n и *веса* (т. е. числа единиц в нем) не более t_A;

2) вычислить двоичный вектор $C_i = M_i \hat{G}_A \oplus Z_i$;

3) послать вектор C_i к пользователю A как криптограмму для сообщения M_i.

3.6.4. Дешифрование КС Мак-Элис

Для того чтобы восстановить сообщение M_i по криптограмме C_i, пользовательA должен выполнить следующие шаги:

1) вычислить $\hat{C}_i = C_i \cdot P_A^{-1}$, где P_A^{-1} это матрица обратная к P_A;

2) используя известный алгоритм декодирования для кода с порождающей матрицей G_A, исправить не более t_A ошибок в $\hat{C}_i$, что даст некоторый двоичный вектор $\hat{M}_i$ длины k;

3) восстановить $M_i = \hat{M}_i \cdot S_A^{-1}$.

Для доказательства того, что описанная выше процедура действительно восстанавливает зашифрованное сообщение M_i, преобразуем сначала представление для $\hat{C}_i$:

$$\hat{C}_i = C_i \cdot P_A^{-1} = \left(M_i \hat{G}_A \oplus Z_i\right) \cdot P_A^{-1} = \left(M_i S_A G_A P_A \oplus Z_i\right) \cdot P_A^{-1} = \left(M_i S_A\right) G_A \oplus Z_i \cdot P_A^{-1}. \quad (3.15)$$

Из выражения (3.15) видно, что двоичный вектор $\hat{C}_i$ представляет собой закодированное ЛК (с порождающей матрицей G_A) сообщение $\left(M_i S_A\right)$ с добавкой двоичного шума $Z_i P_A^{-1}$ веса не более t_A, так как P_A^{-1} будет также перестановочной матрицей, умножение на которую сохраняет вес слова Z_i, который был определен ранее как не более, чем t_A.

Поскольку код с порождающей матрицей G_A может гарантированно исправить не менее t_A ошибок, это означает, что по $\hat{C}_i$ пользовательA может абсолютно точно восстановить $\left(M_i S_A\right)$, причем сложность декодирования будет при этом полиномиальной. Наконец, исходное сообщение M_i восстанавливается после умножения последнего результата на S_A^{-1}, т. е.:

$$M_i S_A \cdot S_A^{-1} = M_i.$$

Заметим, что в отличие от метода РША, и подобно тому, как это было в шифре Эль-Гамаля, метод Мак-Элис является рандомизационным, поскольку случайный вектор помехи Z_i не входит в состав ключей этой КС.

На шаге 1 генерирования ключей можно использовать семейство кодов Гоппы. (Тогда известно [3], что для любого неприводимого над полем $GF\left(2^m\right)$ многочлена $g(x)$ степени t существует двоичный код Гоппы длины $n=2^m$ с числом информационных символов $k \geq n - m \cdot t$. Этот код исправляет не менее, чем t ошибок и имеет полиноминально сложный алгоритм декодирования. Многочлен $g(x)$ называется *многочленом Гоппы* и не совпадает с порождающим многочленом циклического кода, тем более что не всякий код Гоппы является циклическим кодом. По известному многочлену Гоппы можно построить его порождающую матрицу, хотя данная задача не является простой. Известен пакет программ Magma которая ее реализует.)

3.6.5. Стойкость КС Мак-Элис

Рассмотрим две основные атаки на КС Мак-Элис:

1) зная C_i, $\hat{G}_A$, t_A можно попытаться исправить ошибки в C_i, но поскольку, как легко убедиться, порождающая матрица $\hat{G}_A$ является совершенно случайной, для определяемого ей кода не известно неперебоpных методов исправления ошибок, а переборные практически нереализуемы при соответствующем выборе параметров исходного кода;

2) для восстановления M_i можно попытаться случайно выбрать k столбцов в матрице $\hat{G}_A$. Если теперь обозначим через G_k, C_{ik}, Z_{ik} ограничение $\hat{G}_A, C_i, Z_i$ этими столбцами, то будет выполняться уравнение: $\left(C_{ik} + Z_{ik}\right) = M_i \cdot \hat{G}_A$. Если случайно окажется, что $Z_{ik} = 0$ и $\hat{G}_A$ несингулярна, то M_i может быть получено как решение уравнения $C_{ik} = M_i \cdot \hat{G}_k$. Однако вероятность того, что $Z_{ik} = 0$ оказывается равной C_{n-t}^k / C_n^k и при соответствующем выборе параметров она будет весьма малой величиной.

Для обеспечения высокой стойкости КС Мак-Элис, рекомендуется следующие ее параметры [3]:

$$n = 1024 \text{ бит},\ t = 38,\ k > 644.$$

Представленный ранее материал позволяет сформулировать следующие основные свойства КС Мак-Элис:

1) достаточно предсказуемая стойкость;

2) простота шифрования и дешифрования;

3) увеличение длины криптограммы по сравнению с длиной сообщений в n/k раз;

4) большая битовая длина как открытого, так и закрытого ключей.

Последнее свойство особенно существенно ограничивает практическое применение КС Мак-Элис.

4. ЦИФРОВЫЕ ПОДПИСИ С ИСПОЛБЗОВАНИЕМ КС ОК

Аутентификация сообщений с использованием симметричных КС (см. первую часть данной книги) не позволяет решить проблему *возможного спора* между создателем сообщения и его получателем. Действительно, получатель, обладая тем же секретным ключом, что и автор сообщения, может по своему желанию сформировать новое сообщение, добавить к нему аутентификатор и утверждать, что он получил именно это сообщение. Для разрешения проблемы подобного спора, при использовании симметричных КС, необходимо присутствие третьего участника (*доверенного лица*), которому сообщение должно посылаться в копии. Поскольку такой сценарий в большинстве случаев является неудобным, необходимо использовать несимметричные КС, выполняя аутентификацию (в том числе и обеспечение подлинности сообщений) на основе так называемых *цифровых подписей* (ЦП). Еще один вариант ЦП, известный как ЦП Диффи–Лампорта, хотя и использует симметричные методы шифрования, но требует предварительного размещения некоторых цифровых данных в общедоступных для чтения, но защищенных от подделки файлах, что также является затруднительным. Кроме того, после каждого вычисления ЦП часть секретного ключа становится известной. Поэтому такая ЦП фактически оказывается одноразовой [27].

Рассмотрим аутентификацию сообщений без обеспечения их конфиденциальности, поскольку такая задача может быть тривиально решена при помощи аутентификации уже зашифрованных сообщений.

ЦП – это некоторые дополнительные данные, присоединяемые к основному сообщению, которые формируется, зависящими, как от сообщения, так и от секретного ключа автора сообщения. Для проверки подлинности сообщения (называемой иначе процедурой *верификации*) используется открытый ключ автора сообщения, который доступен любому пользователю.

4.1. Основные требования, предъявляемые к ЦП

К цифровой подписи естественно предъявить следующие требования:

1) уникальность (т. е. возможность использования ее только одним определенным пользователем);

2) невозможность отказа от выполненной ЦП;

3) допустимость верификации ЦП любым пользователем;

4) простота выполнения процедур создания и верификации ЦП;

5) возможность выполнения ЦП для больших объемов информации (файлов, дисков и т. д.).

Как видно из перечисленных выше требований, ЦП применительно к данным и документам, представленным в электронной форме, является почти полным эквивалентом обычной *бумажной подписи*. Однако между ними имеется одно важное и принципиальное различие:*бумажная подпись не зависит от содержания сообщения, тогда как ЦПзависит.* Именно эта особенность ЦП и позволяет обеспечить невозможность ее подделки при помощи простого копирования в электронной форме и переноса в другое сообщение!

Стойкость ЦП понимается как ее способность противостоять необнаруженной преднамеренной подделке или случайному искажению подписываемого сообщения, причем для ЦП, основанных на использовании КС ОК, под стойкостью понимается, как правило, *вычислительная стойкость*, т. е. невозможность необнаруженной подделки при ограниченном вычислительном ресурсе злоумышленника. (В литературе [10] рассматриваются также и безусловно стойкие ЦП.)

Существует множество алгоритмов формирования ЦП, основанных на различных КС ОК, к рассмотрению некоторых из них мы сейчас и переходим.

4.2. ЦП на основе различных КС ОК

4.2.1. ЦП на основе КС РША

Пусть имеется некоторое сообщение $\bar{M}$ и некоторым пользователем A сгенерирована пара открытый/закрытый ключ для системы РША, т. е. числа: e_A, n_A, d_A. Тогда сообщение $\bar{M}$ разбивается на блоки, каждый из которых может быть представлен целым числом, не превосходящим n_A. Для каждого из таких сообщений-цифр M, формируется ЦП S по следующему правилу: $S = M^{d_A} \bmod n_A$. Далее ЦП присоединяется к сообщению, образуя

так называемое *подписанное сообщение*, т. е. пару M, S. Для верификации ЦП, пользователь должен получить открытый ключ A, а также «подписанное» (но возможно фальсифицированное) сообщение $\left(\tilde{M}, \tilde{S}\right)$ и вычислить $\tilde{\tilde{M}} = \tilde{S}^{e_A} \bmod n_A$. Далее он сравнивает $\tilde{\tilde{M}}$ с $\tilde{M}$ и при их совпадении полагает, что сообщение $\tilde{M}$ действительно подписано A, в противном же случае отвергает его, как подделку или искажение. Правомочность такой верификации для неискаженных подписанных сообщений основывается непосредственно на алгоритме дешифрования КС РША (подразд. 3.1.3).

Стойкость ЦП данного типа очевидно эквивалентна стойкости КС РША, а последняя, в свою очередь, основывается на сложности решения проблемы факторизации или логарифмирования. Имеются также и побочные атаки на ЦП, похожие на побочные атаки на КС РША [3].

Интересно отметить, что фактически проверка ЦП здесь сводится к дешифрованию сообщения, и поэтому для сообщений, содержащих естественную или искусственно введенную избыточность, можно формировать и передавать, при необходимости, только ЦП S, обеспечивая тем самым выполнение функции его конфиденциальности.

4.2.2. ЦП на основе КС Эль-Гамаля

Поскольку такая ЦП является более сложной и практически важной, то рассмотрим ее подробнее, включая и алгоритм генерирования пар ключей.

Генерирование ключей:

1) генерируется большое простое число p и примитивный элемент α над конечным полем $GF(p)$;

2) генерируется число a: $1 \le a \le p-2$;

3) вычисляется $y = \alpha^a \bmod p$;

4) выбирается открытый ключ верификации ЦП: (p, α, y) и секретный ключ создания ЦП:a.

Формирование ЦП

Если пользователь A хочет подписать сообщение M представленное в виде числа, принадлежащего Z_p, то он выполняет следующие операции:

1) генерирует секретное числоk: $1 \le k \le p-2$, $\gcd(k, p-1) = 1$;

2) вычисляет $r = \alpha^k \bmod p$;

3) вычисляет $k^{-1} \bmod (p-1)$;

4) вычисляет $t = k^{-1}(M - ar) \bmod (p-1)$;

5) формирует ЦП S к сообщению M, как пару чисел $S=(r,t)$.

Проверка (верификация) ЦП

Для того чтобы проверить подпись S, созданную A под сообщением M, любой пользователь выполняет следующие шаги:

1) получает открытый ключ A: (p, α, y);
2) проверяет, что $1 \le r \le p-1$ и если это не выполняется, то отвергает ЦП;
3) рассчитывает $v_1 = y^r r^t \bmod p$;
4) рассчитывает $v_2 = \alpha^M \bmod p$;
5) принимает ЦП как правильную, при условии, что $v_1 = v_2$.

Покажем, что изложенный выше метод верификации ЦП действительно даст правильный ответ для корректно созданной ЦП.

Если ЦП была сгенерирована A, то $t = k^{-1}(M - ar) \bmod (p-1)$. Умножая обе части этого равенства на k, получаем $tk = (M - ar) \bmod (p-1)$. После переноса ar в левую часть, имеем: $(ar + kt) = M \bmod (p-1)$, что эквивалентно следующему равенству $M = (p-1) \cdot l + ar + kt$, где l – целое число.

Найдем теперь v_2, полученное на шаге 4 алгоритма верификации, используя теорему Ферма для дальнейших преобразований

$$v_2 = \alpha^M \bmod p = \alpha^{(p-1)\cdot l + ar + kt} \bmod p = \alpha^{ar+kt} \bmod p = \left(\alpha^a\right)^r \cdot \left(\alpha^k\right)^t = y^r \cdot r^t \bmod p = v_1,$$

что и доказывает правильность верификации.

Стойкость ЦП на основе КС Эль-Гамаля

Злоумышленник может попытаться подделать подпись к своему сообщению M следующим образом: сгенерировать случайное число k, вычислить $r = \alpha^k \bmod p$, а затем попытаться найти $t = k^{-1}(M - ar) \bmod (p-1)$. Однако для выполнения последней операции ему необходимо знать a, что возможно лишь после решения задачи *дискретного логарифмирования* (шаг 3 генерирования ключей). При соответствующем выборе параметров ЦП эта задача оказывается вычислительно нереализуемой.

При выполнении ЦП к различным сообщениям, необходимо генерировать различные случайные числа k, поскольку в противном случае секретный ключ ЦП легко вычисляется. Действительно, если это не так, то получаем следующие равенства:

$$t_1 = k^{-1}(M_1 - ar) \bmod (p-1), \quad t_2 = k^{-1}(M_2 - ar) \bmod (p-1),$$

из которых получаем, что

$$(t_1 - t_2)k = (M_1 - M_2) \bmod (p-1).$$

Если $t_1 - t_2 \neq 0 \bmod (p-1)$, что весьма вероятно,то

$$k = (t_1 - t_2)^{-1}(M_1 - M_2) \bmod (p-1).$$

После нахождения k, вычисление секретного ключа a оказывается весьма простым по известным числам t и r (шаг 4 генерирования ЦП). Существование такой атаки требует хранения в секрете случайных чисел k или их уничтожения сразу же после генерирования ЦП, поскольку если злоумышленник будет иметь к ним доступ, то он легко найдет секретный ключ и сможет выполнять подписи под любыми сообщениями от лица законного пользователя. Не представляет труда подделать ЦП под бессмысленным (случайным) сообщением. (Такая атака называется *экзистенциальной*.) Действительно, для этого достаточно случайно сгенерировать пару чисел u и v, удовлетворяющих условию $\gcd(u, p-1) = 1$, а затем вычислить элементы ЦП $r = \alpha^u y^v \bmod p = \alpha^u \cdot \alpha^{av} \bmod p = \alpha^{u+av}$ и $t = -rv^{-1} \bmod (p-1)$ для бессмысленного сообщения $M = t \cdot u \bmod (p-1)$ Проверка ЦП под таким сообщением даст правильный результат, поскольку

$$v_1 = y^r r^t \bmod p = \alpha^{ar} \cdot \alpha^{(u+av)t} \bmod p,$$

$$v_2 = a^M \bmod p = \alpha^{tu} \cdot \bmod p,$$

и равенство $v_1 = v_2$ выполняется так как выполняется равенство

$$\alpha^{ar} \cdot \alpha^{(u+av)t} = \alpha^{tu} \bmod p \sim \alpha^{ar} \cdot \alpha^{avt} =$$

$$= 1 \bmod p \sim \alpha^{ar} \cdot \alpha^{av\left(-\frac{r}{v}\right)} \bmod (p-1) \sim \alpha^{ar} \cdot \alpha^{-ar} \bmod p = 1,$$

где последнее равенство очевидно.

Таким образом, при выборе модуля p, который в двоичном представлении имеет длину более 768 бит, обеспечивается хорошая стойкость ЦП, а для обеспечения долговременной стойкости целесообразно увеличить ее до 1024 бит.

Заметим, что возможны модификации алгоритма ЦП Эль-Гамаля, которые обеспечивают помимо выполнения ЦП, также и выполнение функции конфиденциальности сообщений [13].

4.2.3. Обобщение ЦП на случай эллиптических кривых

Такое обобщение легко достигается при помощи замены операции возведения в степень по модулю $(\alpha^a \bmod p)$ на операцию многократного сложению точки P самой с собой $(a \cdot P)$. Казалось бы, легче всего использовать алгоритм формирования ЦП по Эль-Гамалю, однако он все же требует некоторых изменений, поскольку если на 2-м шаге формирование ЦП выполнялась бы процедура вычисления $r = P \cdot k$, то на 3-м ша-

ге проверки ЦП вычисление $v_1 = y^r \cdot r^t$ оказывается невозможным, так как не определена операция $y^r = (P \cdot a)^{P \cdot k}$ возведения точки в степень точки.

Поэтому опишем кратко метод выработки ключей, выполнения ЦП и ее проверки, задаваемый алгоритмом «The Elliptic Curve Digital Signature Algorithm (ECDSA)» [24].

Прежде всего отметим, что ЭК выбирается либо над полем $GF(p)$ с уравнением $y^2 = x^3 + ax + b$, либо над $GF(2^m)$ с уравнением $y^2 + xy = x^3 + ax^2 + b$. (Мы будем рассматривать только первый случай.) Рассчитывается количество точек N и выбирается простой делитель n числа $N\left(n > 2^{160}, n > 4\sqrt{p}\right)$, а также условие: $n \nmid p^k - 1$ для каждого k, $1 \le k \le 20$, $n \ne p$. Если эти условия не выполняются, то случайно генерируются другие коэффициенты ЭК a и b. Выбирается произвольная точка $G' \in E$ и находится $G = (N/n)G'$. Если $G = 0$, то выбор ЭК повторяется). Проверяется также выполнение условия $4a^3 + 27b^2 \ne 0 \bmod p$.

Генерация ключей для ECDSA.

1. Генерируется случайное число $d \in [1, n-1]$.
2. Вычисляется точка на ЭК $Q = d \cdot G$.
3. Полагается, что d – это секретный ключ ЦП, а Q – открытый ключ ЦП.

Формирование ЦП для ECDSA

1. Необходимо выбрать случайно целое число k, $1 \le k \le n-1$.
2. Вычислить $k \cdot G = (x_1, y_1)$.
3. Вычислить $r = x_1 \bmod n$. Если $r = 0$, то вернуться к шагу 1.
4. Вычислить $k^{-1} \bmod n$.
5. Далее нужно вычислить $s = k^{-1}(M + dr) \bmod n$, где M – подписываемое сообщение, представленное как целое число меньше n. Если $s = 0 \bmod n$, то вернуться к шагу 1.
6. ЦП к сообщению M необходимо рассматривать как числовую пару (r, s).

Проверка ЦП для ECDSA.

1. Нужно проверить, что r и s находятся в интервале $[1, n-1]$.
2. Вычислить $\omega = s^{-1} \bmod n$.
3. Вычислить $u_1 = M\omega \bmod n$ и $u_2 = r \cdot \omega \bmod n$.
4. Вычислить $X = u_1 G + u_2 Q$, $X = (x_1, y_1)$.
5. Если $X = 0$, то отклонить ЦП. Если нет, то вычислить $v = x_1 \bmod n$.
6. Принять ЦП тогда и только тогда, когда $v = r$.

Если ЦП (r, s) сообщения M действительно сформирована законным пользователем, имеющим пару ключей (Q, d), то проверка ЦП по п. 6 всегда окажется корректной. Действительно,

$$k = s^{-1}(M + dr \bmod n) = s^{-1}M + s^{-1}rd \,(\bmod n) = \omega \cdot M + \omega \cdot rd \,(\bmod n) = u_1 + u_2 d \,(\bmod n).$$

Тогда $u_1 G + u_2 Q = (u_1 + u_2 d)G = kG$ и поэтому $v = r$, что и требовалось для корректности ЦП.

Преимущество выполнения ЦП на эллиптических кривых по сравнению с модульными операциями, также как и в случае шифрования, состоит в том, что длина от-

крытого ключа может быть выбрана значительно меньше при обеспечении такой же стойкости ЦП при атаке на нее дискретным логарифмированием.

Все рассмотренные ранее варианты выполнения ЦП имеют три основных недостатка:

1) значительное увеличение объема (длины) подписанного сообщения, по сравнению с объемом (длиной) самого сообщения, поскольку длина ЦП оказывается больше или равной длине сообщения;

2) вычисление ЦП к каждому последовательному сообщению $M_i, i = 1, 2, \ldots L$ приводит к недопустимо большому расходу времени при выполнении ЦП для значительных величин L;

3) как было отмечено при описании ЦП на основе КС Эль-Гамаля, для сообщений, не содержащих избыточности (например, данных или криптограмм), возможна подделка ЦП.

Для устранения этих недостатков используются так называемые *бесключевые хеш-функции*. В этом случае подписи подлежит не само сообщение, а хеш-функция сообщения, т. е. обычно короткая цепочка длины порядка 128–512 бит. Ясно, что при этом первый и второй недостатки, приведенные выше, устраняются, поскольку длина ЦП оказывается равной длине хеш-функции. Что же касается третьего недостатка, то он также устраняется при использовании так называемых однонаправленных хеш-функций, определение и способы реализации которых будут описаны далее.

4.3. Бесключевые хеш-функции

Как отмечалось выше, подписываются обычно не сами сообщения, а результат их преобразования хеш-функциями (ХФ), который будем далее называть просто *хешем*. В первой части данной книги уже рассматривались *ключевые хеш-функции*. Однако при выполнении ЦП не имеет смысла их использовать, поскольку проверка ЦП не должна, как правило, требовать знания секретного ключа. Кроме того, ключевые ХФ строятся обычно на основе симметричных систем шифрования, тогда как ЦП основана на несимметричных КС. Таким образом, вид ХФ, т. е. алгоритм вычисления хеша $h = h(x)$, должен быть полностью известным. (Далее, не умаляя общности, будем полагать, что как аргументы ХФ x, так и значения хешей h являются двоичными цепочками длины n и m соответственно.)

Вместе с тем выполнение подписей под ХФ (а не непосредственно под самими сообщениями) приводит к появлению новой атаки на ЦП. Действительно, если удастся подменить истинное сообщение на другое, но та-

кое, которое имеет тот же самый хеш, то проверка ЦП под этим фальсифицированным сообщением покажет ее правильность, и, следовательно, такая подмена не будет обнаружена.

Поэтому ХФ, используемые для выполнения ЦП, (называемые *криптографическими хеш-функциями*) должны удовлетворять особым требованиям.

4.3.1. Основные требования, предъявляемые к криптографическим ХФ

1. *Однонаправленность.* При известном хеше h вычислительно неосуществимо (т. е. требует нереализуемо большого числа операций) нахождение хотя бы одного значения x, для которого $h(x)=h$, т. е. $h(x)$ оказывается *однонаправленной функцией (ОНФ)*.

2. *Слабая коллизионная стойкость.* Для заданных $x, h(x)=h$ вычислительно неосуществимо найти такое значение x', которое удовлетворяет уравнению $h(x')=h$.

3. *Сильная коллизионная стойкость.* Вычислительно неосуществимо найти такую пару аргументов x, x', для которых выполняется соотношение $h(x)=h(x')$.

Хорошая криптографическая ХФ должна обладать таким свойством, что при любом случайном и равномерно распределенном выборе аргумента x, вероятность преобразования его ХФ в фиксированный хеш $h=h(x)$ будет близка к величине 2^{-m}, где m – длина двоичной цепочки хеша. Конечно, с другой стороны, представляется парадоксальным, что, если сообщение имеет длину n бит, а длина хеша равна m, то количество всех возможных хешей равно 2^m, а всех возможных сообщений $2^n >> 2^m$, и тогда для заданного хеша существует 2^{n-m} сообщений, которые отображаются в тот же самый хеш. Однако вероятность попасть на одно из таких случайно выбранных сообщений должна оказаться весьма малой.

Для того чтобы «снять» кажущийся парадокс (т. е. противоречие между реальностью и интуицией) рассмотрим простую ХФ следующего вида: $y=[x\times h]_m$, где $x, h \in GF(2)^n$, «×» – умножение в конечном поле $GF(2^n)$, а $[z]_m$ – выбор первых «m» координат вектора z.

Тогда ясно, что количество векторов $x \in GF(2)^n$, которые будут хешироваться в любой вектор $\tilde{y} \in GF(2)^m$ будет одинаково и равно 2^{n-m}. Все эти вектора имеют вид $(y, y_0)\times h^{-1}$, где (y, y_0) – конкатенация вектора y длины m и любого вектора y_0 длины $n-m$, h^{-1}, обратный к h элемент по-

ля $GF\left(2^n\right)$, а умножение производится в конечном поле $GF\left(2^n\right)$. Поскольку при равномерном распределении на входе ХФ вероятность появления любого $x \in GF\left(2\right)^n$ равна 2^{-n}, то вероятность попасть на входной вектор $x' \neq x$, который дает после хеширования вектор $\tilde{y}$ будет равна $2^{-n}\left(2^{n-m}-1\right) \approx 2^{-m}$, что вполне строго доказывает желаемый результат, т. е. фактически «снимает» парадокс. Конечно, приведенную ХФ нельзя использовать для реального хеширования, поскольку ввиду линейности она не обладает даже слабой коллизионной стойкостью. При использовании нелинейных ХФ этот факт надо доказывать заново, и, возможно, оценка вероятности случайно обращенной ХФ окажется значительно хуже, чем приведенная выше.

При случайном выборе L различных аргументов ХФ $x_1, x_2, ..., x_L$ вероятность P того, что хотя бы для одного из них хеш совпадет с заранее заданным значением h, будет равна $1-\left(1-2^{-m}\right)^L \cong L2^{-m}$. Поэтому число попыток L, необходимых для обращения ХФ, будет с вероятностью P равно $P2^m$.

В частном случае вероятности успеха $P=1/2$ число таких попыток оказывается равным 2^{m-1}. Отсюда, казалось бы, следует, что даже при выборе длины хеша равной 64, такая атака оказывается нереальной, не говоря уже о том, что даже при успешном подборе аргумента, дающего заданное значение хеша, это еще не означает, что полученный аргумент будет соответствовать сообщению, которое хочет фальсифицировать злоумышленник.

Оказывается, однако, что нарушение свойства *строгой коллизионной устойчивости* является более простой задачей и это требует значительного увеличения длины хеша для обеспечения высокой стойкости ЦП.

Атака с целью нарушения этого свойства основана на так называемом *парадоксе дней рождения*, суть которого состоит в следующем.

Пусть имеется урна, содержащая N шаров, перенумерованных от 1 до N. Из этой урны случайно извлекается M шаров с замещением (т. е. с возвратом извлеченного шара обратно в урну). Номера извлеченных шаров регистрируются в списке. Тогда вероятность $P\left(N,M\right)$ того, что в этом списке хотя бы один номер шара встретится не менее двух раз может быть оценена асимптотически (т. е. при больших величинах N) по следующей формуле [12]:

$$P\left(N,M\right)=1-\exp\left(-M^2/2N\right). \qquad (4.1)$$

Эта задача называется парадоксом дней рождения потому, что если выбрать число N равным числу дней в не високосном году, т. е. 365, а число людей M в случайно собранной группе равным 23, то вероятность, что

по крайней мере два человека из этой группы родились в один день года, оказывается удивительно большой ….0,5155.

Атака на ХФ, основанная на парадоксе дней рождения, состоит в следующем.

Пусть имеется истинное (легальное) сообщение x и сообщение x', которое хочет создать злоумышленник таким образом, чтобы оно было подтверждено подписью легального лица. Тогда злоумышленник генерирует список Z, состоящий из $2^{m/2}$ небольших модификаций сообщения $\tilde{x}$, благоприятных для легального владельца ЦП. Далее он генерирует поочередно сообщения x', любое из которых подходит для фальсификации. Как только появляется такое $\tilde{x}'$, что $h(\tilde{x}')=h(\tilde{x})$ для любого $\tilde{x}\in Z$, процедура генерирования $\tilde{x}'$ заканчивается. Затем злоумышленник просит владельца ЦП подписать сообщение $\tilde{x}$ и подменяет его сообщением $\tilde{x}'$. Теперь он может быть уверен, что скопированная им подпись к сообщению $\tilde{x}$ будет верна и для сообщения $\tilde{x}'$. Среднее число попыток генерирования сообщения $\tilde{x}'$ будет $2^{m/2}$ с вероятностью порядка $1/2$ как это следует из формулы (4.1), если в ней положить $N=2^m$, $M=2^{m/2}$.

4.3.2. Способы построения стойких криптографических бесключевых ХФ

Как было отмечено в предыдущем пункте, ХФ должны удовлетворять трем основным условиям, что достигается выбором алгоритма формирования хешей и соответствующим выбором параметров этого алгоритма. Известно три основных способа построения ХФ на основе использования:

1) блоковых шифров;

2) специально разработанных алгоритмов хеширования;

3) КС ОК.

Возможны и комбинации данных методов.

Далее будем рассматривать только первый и второй подходы, поскольку третий представляет пока лишь теоретический интерес.

Использование первого подхода представляется вполне естественным, поскольку стойкие блоковые шифры не позволяют найти вход (сообщение) по выходу (криптограмме). Однако нам нужно построить бесключевую ХФ!

Если попытаться построить ее как итерацию блокового алгоритма шифрования, т. е. когда $h=h_t, h_i=f_{g(h_{i-1})}(M_i)$, $i=1,2,...,t$, $h_0=IV$, где M_i, $i=1,2,...,t$ – последовательные блоки сообщения, $f_x(y)$ – алгоритм блокового шифрования сообщения y на ключе x, IV – известное начальное зна-

чение, $g(\cdot)$ – функция, которая преобразует n-битовый блок в ключевой блок подходящей длины, то легко видеть, что такая ХФ не будет ОНФ.

Действительно, пусть сообщение состоит всего из двух блоков длины n равных длине блоков используемого шифра (т. е. $M=(M_1,M_2)$). Тогда $h=h_2$, где $h_2=f_{g(h_1)}(M_2)$, $h_1=f_{IV}(M_1)$ и нарушение слабой коллизионной стойкости достигается выбором нового сообщения $M'=(M_1,M'_2)$, где $M'_2=f^{-1}_{g(h_1)}(h)$, а $f^{-1}_{(K)}(E)$ – известная функция дешифрования, выбранного шифра при заданном ключе K и криптограмме E, которые полностью известны нарушителю.

Нельзя также пользоваться методом, который был ранее выбран для построения MAC-кода, но при общедоступном ключе K и начальном хеше IV (подразд. 4.4.2. первой части книги):

$$h=h_t,\ h_i=f_K(M_i\oplus h_{i-1}),\ i=1,2,...,t,\ h_0=IV\ .$$

В этом случае также нарушается слабая коллизионная стойкость. Действительно, если сообщение состояло всего из двух блоков M_1 и M_2, то злоумышленник может изменить M_1 на M'_1 и вычислить $h'_1=f^{-1}(K,M'_i)\oplus=IV$ что приведет к корректной ХФ для законного пользователя.

Поэтому при хешировании на основе блоковых шифров, как мы увидим далее, обычно используют некоторые модификации предыдущего метода.

При реализации ХФ на основе блоковых шифров нужно различать методы с длиной хеша равной длине шифруемого блока и длинах хешей кратных длине блока шифрования. Последнее важно, когда используются алгоритмы шифрования с относительно короткой длиной блока, например, как у шифра DES, где эта длина равна 64 бита. Как отмечалось ранее, атака, основанная на парадоксе дней рождения, может оказаться не стойкой в этом случае, но уже ее удвоение, делает такую атаку вычислительно нереализуемой. Рассмотрим далее несколько примеров алгоритмов хеширования для ХФ с одиночной и двойной длиной хешей.

Другие примеры подобного рода ХФ можно найти в [3].

Примеры ХФ с одиночной длиной блоков-выходов (хешей) [3]

Схема Матиаса–Мейера–Осеаса (Matyas–Meyer–Oseas).

$M=(M_1,M_2,...,M_t)$, M_i – это n-битовые блоки.

$h=h_t,\ h_0=IV;\ h_i=f_{g(h_{i-1})}(M_i)\oplus M_i,\ 1\le i\le t$.

Схема Дэвиса–Межера (Davies–Meger).

$M=(M_1,M_2,...,M_t)$, M_i – это k-битовые блоки, где k длина ключа для выбранного алгоритма шифрования;

$h = h_t$, $h_0 = IV$ (известный начальный блок длины n), $h_i = f_{M_i}(h_{i-1}) \oplus h_{i-1}$, $1 \le i \le t$.

Схема Миагучи–Пренила (Miyaguchi–Preneel).

$M = (M_1, M_2, ..., M_t)$, M_i – это n-битовые блоки,

$h = h_t$, $h_0 = IV$; $h_i = f_{g(h_{i-1})}(M_i) \oplus h_{i-1}$, $1 \le i \le t$.

Из приведенных выше примеров следует, что для нарушения слабой коллизионной устойчивости, злоумышленнику необходимо решить проблему нахождения ключей в используемых шифрах по заданным криптограммам и сообщениям, что для стойких шифров представляет собой вычислительно нереализуемую задачу. Поэтому можно полагать, что для таких шифров ХФ, реализованные по приведенным выше алгоритмам, принадлежат к классу ОНФ.

Пример ХФ с двойной длиной выхода (хеша)

Алгоритм на основе шифра DES [3].

Пусть $M = (M_1, M_2, ..., M_t)$, M_i – это 64-битовые блоки, а знак || обозначает конкатенацию, а C_i^L, C_i^R – левая и правая 32-битовые половины блока C_i; IV и $\tilde{IV}$ это 64-битовые несекретные постоянные, которые выбраны (в 16-ричном представлении) следующим образом:

$$IV = 0x5252525252525252;\ \tilde{IV} = 0x2525252525252525.$$

Далее определим g and $\tilde{g}$ как две функции, которые преобразуют 64-битовые блоки u в 56-битовые блоки, подходящие для ключей DES, следующим образом:

$$g(u) = u_1 10 u_4 u_5 u_6 u_7 u_9 u_{10} ... u_{63};\quad \tilde{g}(u) = u_1 01 u_4 u_5 u_6 u_7 u_9 u_{10} ... u_{63},$$

где $u = (u_1, u_2, \ldots, u_{64})$.

Тогда 128-битовые хеши h формируются следующим образом:

$h_0 = IV$, $K_i = g(h_{i-1})$, $C_i = E_{K_i}(M_i) \oplus M_i$, $h_i = C_i^L \| \tilde{C}_i^R$;

$\tilde{h}_0 = \tilde{IV}$, $\tilde{K}_i = \tilde{g}(\tilde{h}_{i-1})$, $\tilde{C}_i = E_{\tilde{K}_i}(M_i) \oplus M_i$, $\tilde{h}_i = \tilde{C}_i^L \| C_i^R$, $i = 1, 2, ..., t$;

$h = h(M) = h_t \| \tilde{h}_t$.

Специально разработанные ХФ

Это класс ХФ, которые не используют непосредственно различные модификации блоковых шифров, но специально разработаны для выполнения преобразования бесключевого хеширования. В англоязычной литературе они обычно называются *предписанными* (*Dedicated*) ХФ или *заданным изначально* (*From scratches* [3]).

В данном классе имеется много различных алгоритмов. Это прежде всего: BCA, RIPEMD, HAVAL, N-hash, FFI-hash, Snefru, MD4, MD5, SHA[3], а также два Российских стандарта. Поскольку полное описание, не только всех, но даже одного из последних алгоритмов является весьма громоздким, ограничимся перечислением основных принципов построения такого алгоритма как ГОСТ Р34.11-2012

Стандарт ГОСТ Р34.11-2012, проектное название «Стрибог», определяет процедуру вычисления хеш-функции для любых последовательностей двоичных символов, которые применяются в криптографических методах обработки и защиты информации, в том числе для процедур обеспечения целостности, аутентификации и электронной цифровой подписи. Длина формируемого хеш-кода 256 или 512 бит.

Вычисление хеш-функции осуществляется согласно итеративному алгоритму (рис. 4.1). Для этого входное сообщение m разбивается на блоки m_j, $j=1,2,...,N$ длиной 512 бит. Для неполного последнего блока используется дополнение вида 00…01 длиной 512 – $|m_N|$, где $|m_N|$ – длина укороченного блока, записываемое слева от блока.

Параллельно с этим вычисляются два блока:

- число обработанных бит,
- контрольная сумма всех блоков.

Над каждым блоком сообщения последовательно осуществляется преобразование с помощью функции сжатия. После этого функцией сжатия обрабатывается блок общей длины и блок контрольной суммы. Последний блок на выходе функции сжатия соответствует хеш-коду сообщения $h(m)$.

Начальное (внутреннее) состояние функции сжатия задается инициализационным вектором IV равным 0^{512} для хеш-кода длиной 512 бит и $(00000001)^{64}$ для хеш-кода длиной 256 бит.

Функция сжатия реализует итерационную процедуру (рис. 4.2) над очередным блоком сообщения m_j и предыдущим значением хеш-кода h_{j-1}:

$$h_0 = IV\ , h_j = g\left(m_j, h_{j-1}\right),\ j=1,2,...,N,$$

где

$$g\left(m_j, h_{j-1}\right) = K_{13} \oplus LPS\left(...\oplus\left(K_3 \oplus LPS\left(K_2 \oplus LPS\left(m_j, K_1\right)\right)\right)...\right)\oplus m_j \oplus h_{j-1},$$

а коэффициенты K_i вычисляется по формуле:

$$K_i = LPS\left(K_{i-1}, C_{i-1}\right)\quad i=2,...,13\ ,$$

где C_i – интерационные константы, представляющие 512-разрядные двоичные числа, задаваемые ГОСТом $K_1 = h_{j-1}$.

Преобразование LPS составляет основу функции сжатия и включает три последовательно проводимых преобразования S, P, L:

S – замена байт. 512 бит аргумента представляются как 64-байтный массив и каждый байт заменяется по заданной таблице.

P – переупорядочивание байт. Байты аргумента меняются местами по определенному стандартом порядку.

L–линейное преобразование. Аргумент рассматривается как восемь 64-битных векторов, каждый из которых заменяется результатом умножения вектора на определенную стандартом матрицу A_{64X64} над полем $GF(2)$.

Реализация стандарта на цифровом процессоре архитектуры x86.64 обеспечивает скорость работы 94 МБ/с и требует 87 тактов на байт.

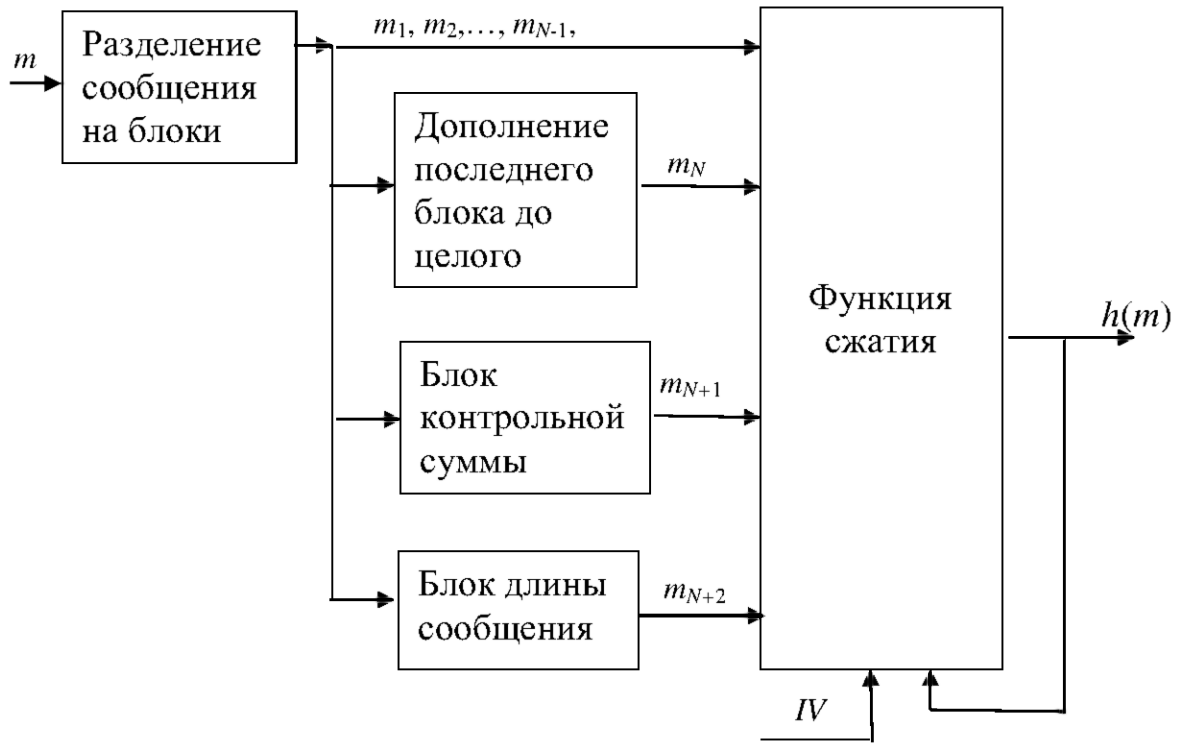


Рис. 4.1. Алгоритм вычисления хэш-функции

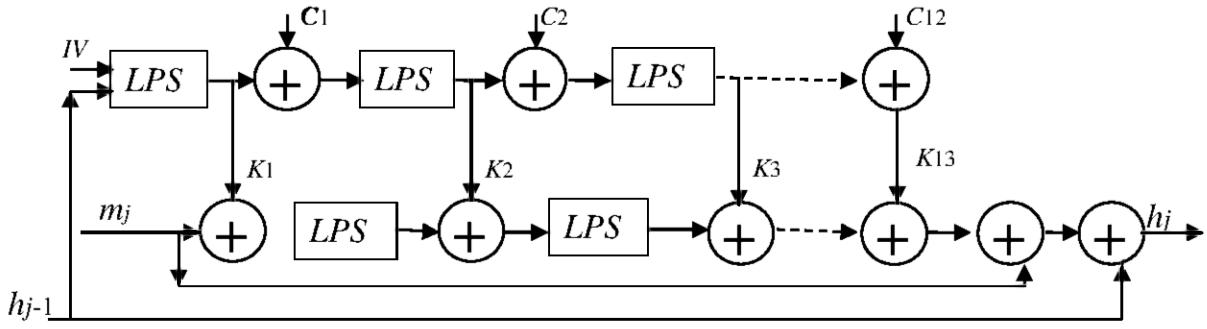


Рис. 4.2. Функция сжатия

4.4. Выводы о возможности построения стойких ЦП

Материал, изложенный в предыдущих разделах, позволяет сделать следующие важные выводы о возможности аутентификации сообщений на основе ЦП.

1. ЦП является эффективным средством обеспечения подлинности и целостности всех видов сообщений, которые могут быть представлены в цифровой форме.

2. Стойкость ЦП (т. е. ее устойчивость к необнаруженной подделке сообщений) может быть обеспечена при помощи выбора необходимого несимметричного алгоритма шифрования и ХФ, а также соответствующих параметров этих алгоритмов, причем в практически используемых системах под стойкостью понимается вычислительная стойкость, определяемая числом необходимых операций или (и) объемом памяти, необходимыми для подделки ЦП. Использование бесключевых криптографических ХФ позволяет также избежать так называемой экзистенциальной атаки, которая заключается в формировании фальшивой ЦП под бессмысленным сообщением (подразд. 4.2.2). Однако она не устраняет возможности формирования фальшивой ЦП, например, при использовании алгоритма Эль-Гамаля, если при верификации ЦП был неосмотрительно проигнорирован шаг 2, а именно проверка того условия, что одна из компонент ЦП r удовлетворяет условию: $1 \le r \le p-1$. Если это условие было нарушено, то злоумышленник может «подписать» любое сообщение M' (в том числе и смысловое), причем подпись под ним будет верифицирована как подлинная. Для этого ему требуется только иметь в своем распоряжении какое-либо другое сообщение $M \ne M'$, подписанное легальной ЦП (r, t). Ввиду важности и простой реализуемости такой атаки, рассмотрим ее более подробно.

Атакующий вычисляет ХФ $h(M')$ от M' и находит $u = h(M') \cdot (h(M))^{-1} \bmod (p-1)$ в предположении, что $(h(M))^{-1} \bmod (p-1)$ существует. Затем он вычисляет $t' = t \cdot u \bmod (p-1)$ и r' такое, что $r' = r \cdot u \cdot \bmod (p-1)$ и $r' = r(\bmod p)$, что всегда можно сделать, используя китайскую теорему об остатках. Тогда пара чисел t' и r' будет правильно верифицирована как ЦП под сообщением M', если не требовать выполнения условия $1 \le r' \le p-1$. (В действительности r' вычисляется по $\bmod p \cdot (p-1)$ и поэтому может оказаться много больше $p-1$).

3. ЦП обладает всеми основными свойствами обычной бумажной подписи и поэтому она может рассматриваться наравне с последней и при судебном разбирательстве. (В этом случае используется так называемая

квалифицированная ЦП [29].) Однако в отличие от бумажной подписи, передача секретного ключа ЦП другим лицам делает их фактически уполномоченными выполнять эту подпись.

4. ЦП требует определенного объема памяти для ее хранения. Однако при больших объемах сообщения это относительное увеличение объема оказывается незначительным.

5. Время создания и верификации ЦП (даже при обеспечении долговременной стойкости) оказывается сравнительно небольшим.

6. Длины ключей для создания и верификации ЦП имеют порядок 768-1024 бит. В некоторых приложениях, где к длинам ключей предъявляются повышенные требования, используются специальные алгоритмы ЦП, например, на основе эллиптических кривых [25].

7. Подделка открытых ключей злоумышленниками может привести к ложной верификации ЦП. Для защиты от такой угрозы нужно использовать так называемые *центры сертификации ключей*, где их подлинность гарантируется доверием к этим центрам со стороны легальных пользователей [29].

8. Существуют различные разновидности ЦП, которые будут рассматриваться далее в разделе, посвященном криптографическим протоколам.

4.5. Некоторые стандарты ЦП

1) DSA (федеральный стандарт США). Является обобщением ЦП Эль-Гамиля с тем основным отличием, что для генерирования, формирования и проверки ЦП используются операции над числами не по одному, а по двум простым модулям;

2) ГОСТ Р 3410-94. Российский стандарт ЦП в котором используются модульные операции;

3) ГОСТ Р34.10-2012. Российский стандарт ЦП на основе использования ЭК.

Ниже приведем сокращенное описание процедур генерации ключей, формирования и проверки ЦП в обозначениях этого стандарта [25]. Зададим основные параметры ЦП:

p – модуль ЭК E, задаваемый уравнением $y^2 = x^3 + ax + b \bmod p$, где $a, b \in F_p$ (в наших обозначениях в $GF(p)$; $4a^3 + 27b^2 \neq 0$;

m – порядок группы точек ЭК E $(m \neq p)$;

q – простое число (порядок циклической подгруппы ЭК $2^{254} < q < 2^{256}$ или $2^{508} < q < 2^{512}$; $qP = 0$, где $P \neq 0$ точка на заданной ЭК.

Определим ХФ $\bar{h} = h(M)$ от сообщения M. Обозначим $e = \left[\bar{h}\right] \bmod q$, где $[x]$ – представление $\bar{x}$ числом.

Генерация ключей производится следующим образом. Случайно выбирается d – ключ формирования ЦП $(0 < d < q)$.

Вычисляется $Q = dP$ – ключ проверки ЦП.

Формирование ЦП производится следующими шагами:

1) вычислить $\bar{h} = h(M)$ и соответствующее $\bar{h}$ число e;

2) сгенерировать случайное число k, $0 < k < q$;

3) вычислить точку на ЭК $C = kP$ и определить $r = x_c \pmod q$, где x_c это x – координата точки C;

4) вычислить $s = (rd + ke) \pmod q$;

5) выбрать как ЦП к сообщению M пару чисел (r, s).

Проверка ЦП требует выполнения следующих операций:

1) проверить выполнение условий: $0 < r < q$, $0 < s < q$;

2) вычислить ХФ сообщения: $\bar{h} = h(M)$ и соответствующее ей число $e = \left[\bar{h}\right] \pmod q$;

3) вычислить $v = e^{-1} \pmod q$;

4) вычислить числа $z_1 = sv \pmod q$, $z_2 = -rv \pmod q$;

5) вычислить точку C на ЭК, где $C = z_1 P + z_2 Q$ и взять ее координату $R = x_c \pmod q$;

6) принять ЦП тогда и только тогда, когда $R = r$.

Из представленного описания стандарта видно, что он весьма похож на метод ECDSA, описанный в разд. 4.2.3, но при генерировании и проверке ЦП имеются некоторые отличия.

5. КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ

В настоящее время многие коммерческие сделки и деловые операции выполняются через Интернет, причем для их защиты от действий злоумышленников оказывается вполне достаточным использования таких изученных ранее, криптографических функций, как *конфиденциальность* и *аутентификация*, реализуемых при помощи криптосистем шифрования/дешифрования и цифровой подписи.

Однако существует множество других взаимодействий между двумя или более пользователями сети Интернет (называемых обычно протоколами), для которых обеспечение их безопасности не может быть реализовано только перечисленными выше средствами. Примерами подобных протоколов являются: электронные платежи, лотереи и аукционы, тайное голосование, анонимная покупка данных, выполнение совместных вычислений с сохранением индивидуальных данных в секрете и т.д.

Для обеспечения безопасности выполнения этих или иных действий, выполняемых дистанционно (через Интернет или в какой-либо другой локальной или корпоративной сети), в условиях присутствия в этих сетях недобросовестных пользователей (в том числе и среди законных участников этих процедур), используются специально для них разработанныетак называемые *криптографические протоколы* (КП).

В подразд. 5.1 проведем краткий обзор важнейших КП, а в подразд. 5.2 достаточно подробно рассмотрим такие КП, как разделение секретных данных, идентификация на основе нулевых знаний, поручительство информации, обманчивая передача, секретные совместные вычисления, неоспоримое шифрование и тайное голосование.

5.1.Обзор основных КП [14]

Таблица 5.1

Обманчивая передача, секретные совместные вычисления, неоспоримое шифрование

№ п/п	Название протокола	Задачи, решаемые после выполнения данного протокола
1	Разделение секретов	Метод, при котором организатор протокола вычисляет частные секреты («тени») k_i, $1 \le i \le n$ от исходного секрета k и секретным образом распределяет k_i пользователям так, чтобы выполнялось следующее условие: любые m или более пользователей, которые могут объединить свои тени k_i, легко восстанавливают k, но любая группа, состоящая из $m - 1$ или меньшего числа пользователей не может этого сделать. Знание $m - 1$ или меньшего числа теней вообще не дает никакой информации об исходном секрете

Продолжение табл. 5.1

№ п/п	Название протокола	Задачи, решаемые после выполнения данного протокола
2	Скрытный канал (фактически это стеганография, но используемая в существующих криптосистемах)	При помощи обмена совершенно неподозрительными сведениями, два участника хотят передать некоторую дополнительную информацию в присутствии наблюдателя, который не должен догадаться о самом факте ее присутствия в основном сообщении
3	Различные версии ЦП	
3.1	Неоспоримая ЦП	Подобно обычной ЦП, неоспоримая ЦП зависит от подписанного документа и секретного ключа автора ЦП. Однако в противоположность обычной ЦП, неоспоримая ЦП не может быть верифицирована без участия ее автора
3.2	Останавливаемая ЦП (обеспечивает ее стойкость, не зависящую от вычислительной мощности нарушителя)	Если подписавший хочет отменить свою ЦП, подозревая, скажем, подделку, то надо убедить в этом суд. Основная идея останавливаемой ЦП состоит в том, что для каждого открытого ключа имеется множество секретных ключей подписи и невозможно узнать, какой из них был использован при ЦП данного документа в действительности
3.3	Групповая ЦП	Только члены определенной группы уполномочены подписывать некоторые сообщения. Каждый член этой группы может проверить правильность данной подписи, но для «рядовых» членов группы невозможно определить авторство подписи и только специально уполномоченный «супервизор» сможет это сделать
3.4	«Слепая» ЦП	Необходимо подписать сообщение, не зная его содержания так, чтобы каждый впоследствии смог бы убедиться в правильности этой подписи. Если небезопасно подписывать сообщения «вслепую», то должна существовать возможность получать определенные сведения о подписанных сообщениях, сохраняя при этом основные свойства слепой подписи
3.5	Одновременное подписание контрактов	Два участника хотят подписать контракт, но ни один из них не должен сделать это раньше другого

Продолжение табл. 5.1

№ п/п	Название протокола	Задачи, решаемые после выполнения данного протокола
3.6	Заказная ЦП	Один участник хочет послать сообщение другому, но при условии, что тот сможет его прочитать только после того как пошлет отправителю квитанцию о получении этого сообщения
4	Поручительство информации	Один из участников хочет передать бит (в более общем случае цепочку бит) на хранение другому участнику, но так чтобы последний смог его прочитать позднее (по дополнительному распоряжению и при помощи посылки дополнительной информации от первого участника). Однако, с другой стороны, участник, сохраняющий бит, должен исключить возможность его изменения при поступлении дополнительных сведений со стороны первого участника
5	Доказательства с нулевым разглашением секретов	Доказывающий участник должен убедить проверяющего участника, что он обладает определенной информацией (например, секретным ключом или доказательством гипотезы Гольдбаха), но не выдать ему при этом абсолютно никаких сведений о самой этой информации, кроме, конечно, самого факта, что он эту информацию имеет. (Частным случаем этого протокола является протокол Фейге–Фиата–Шамира об идентификации пользователей, который подробно изучается далее)
6	Обманчивая передача	Некоторый пользователь обладает набором секретов (скажем, компрометирующих документов). Другой пользователь хочет купить один из них, но при условии, что продавец компроматов не узнает, какой именно документ у него покупают

Продолжение табл. 5.1

№ п/п	Название протокола	Задачи, решаемые после выполнения данного протокола
7	Тайное голосование	Это компьютерный аналог обычного тайного голосования, выполняемого дистанционно при помощи протокола, который должен удовлетворять следующим требованиям: - в голосовании принимают участие только авторизованные избиратели; - никто не может проголосовать более одного раза; - никто (включая избирательную комиссию) не может узнать результат голосования каждого из участников; - никто (включая избирательную комисию) не может незаметно изменить результаты голосования; - все авторизованные избиратели могут убедиться, что их голос правильно учтен в итоге голосования; - общеизвестным становится список всех проголосовавших. (Возможны и другие наборы требований.)
8	Совместные секретные вычисления	Группа участников хочет выполнить вычисление определенной функции от своих секретных данных, но так, чтобы эти индивидуальные секретные данные не стали бы известны другим участникам
9	Цифровая наличность (ЦН)	Использование кредитных карт не решает проблему анонимности платежей, поскольку позволяет отследить с какого счета снимаются и на чей счет переводятся деньги. Таким свойством обладает обычная бумажная наличность. Ее цифровой аналог (ЦН) должен обладать следующими свойствами: - ЦН может передаваться по компьютерным сетям; - ЦН не может быть скопирована и использована повторно; - никто не может отследить связи между покупателем и продавцом; - магазин не нуждается в наличии связи с банком покупателя (*off-line* покупки); - ЦН может быть передана (по желанию ее хозяина) другим пользователям; - ЦН может быть разменяна на более мелкие части, дающие в сумме ту же величину ЦН

Окончание табл. 5.1

№ п/п	Название протокола	Задачи, решаемые после выполнения данного протокола
10	Ограничение расстояния	Необходимо криптографически удостоверить расстояние между участниками протокола, выполняемого по компьютерным сетям. Существование данного протокола решает так называемую проблему «*мафиозной атаки*» (известной также под названием «*гроссмейстерской проблемы*» [14])
11	Анонимное широковещание	Группа пользователей некоторой сети передает по ней широковещательные сообщения, предназначенные определенным пользователям, и только они могут дешифровать эти сообщения. Условие этого протокола заключается в том, что никто, кроме отправителя и легального получателя сообщений, не может определить этих пользователей, в том числе и при помощи анализа телетрафика
12	Отслеживание предателей	Платные ТВ-программы могут быть зашифрованы при помощи ключей, которые передаются легальным подписчикам этих программ. Однако существует опасность, что некоторые из таких нечестных подписчиков (предателей) могут продать ключи другим пользователям, не оплатившим эти программы. Протокол обеспечивает возможность поиска таких «предателей» в случае, когда ТВ-декодер пользователей может быть открыт для необходимого контроля. Заметим, что такая же задача (но с использованием другой техники) решается частным видом стеганографии (fingerprinting)
13	Честные криптосистемы	Необходимо обеспечить конфиденциальность сообщений, однако по постановлению суда эти сообщения должны просто расшифровываться
14	Неоспоримое шифрование	Криптосистема позволяет выдать несекретное по сути сообщение, зашифрованное известной криптосистемой, если это необходимо из-за физического или правового воздействия на собственника криптосистемы, тогда как эти же криптограммы имеют другое (действительно секретное) расшифрование

5.2. Описание процедур выполнения некоторых КП

5.2.1. КП «разделение секретов»

В этом случае в качестве секретных данных могут рассматриваться, например, команды по применению ядерного оружия, либо ключи к зашифрованных файлам, содержащим важную информацию по рецептуре продуктов, различного рода «ноу хау» промышленных процессов и т. п. Разделение здесь необходимо для исключения возможности несанкционированного использования этих данных одной персоной. Разделение данных предполагает, что для выделения основного секрета необходимо объединение частных секретов (*теней*) нескольких лиц, причем в количестве не менее некоторой заданной величины. Очевидно, что тайный сговор достаточно большой группы лиц, владеющих частными секретами, для получения основного секрета значительно менее вероятен, чем появление одного такого «предателя». Это и обуславливает необходимость создания протокола разделения секретов. Возможно использование простейшей схемы разделения секретов – «*Все или никто*», в которой только объединение частных секретов *всех* их обладателей позволит им вычислить основной секрет. Такая схема просто реализуется при помощи чисто случайного генерирования двоичных цепочек секретов k_i, $i = 1, 2, ..., n-1$ всех пользователей кроме одного, тогда как последний получает частный секрет k_n следующего вида:

$$k_n = \oplus \sum_{i=1}^{n-1} k_i \oplus k ,$$

где k – основной секрет, а $\oplus$ – означает побитовое суммирование по mod 2. Однако в этом случае отсутствие (или потеря) даже одного из частных секретов приведет к невозможности восстановления основного секрета. Поэтому и возникает необходимость создания пороговой схемы, где для восстановления основного секрета достаточно объединить не менее чем m из n частных секретов.

Определение 5.1. *Пороговой (n, m) схемой* называется такой алгоритм формирования частных секретов k_i, $i = 1, 2, ..., n$ (*теней*) по основному секрету k, что при объединении m или более таких теней существует алгоритм, позволяющий восстановить в точности основной секрет, тогда как объединение менее чем m теней не дает абсолютно никакой информации об основном секрете k (рис. 5.1).

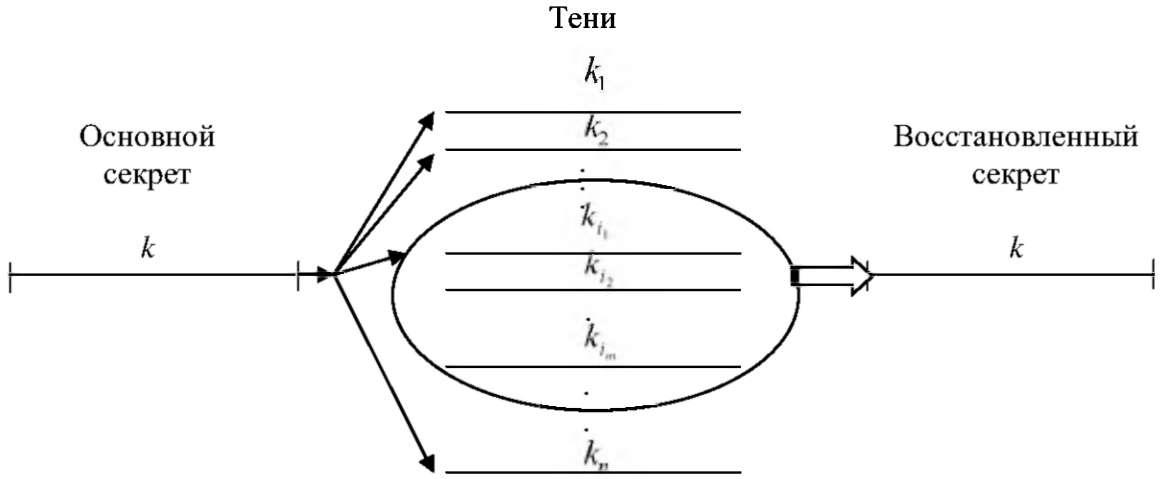


Рис. 5.1. Пороговая (*n*, *m*) схема

Существует множество различных способов построения пороговых схем. Рассмотрим далее описание одной из них, основанной на *интерполяции полиномов над конечными полями*.

Пусть основной секрет $k \in GF(p)$, где $GF(p)$ конечное простое поле, что всегда возможно для любого k при выборе необходимой величины p.

Выберем коэффициент a_0 полинома степени $m-1$ при его постоянном члене равным основному ключу k, а остальные его коэффициенты $a_1, ..., a_{m-1} \in GF(p)$ выберем чисто случайными. Тогда полином (m – 1)-й степени однозначно определяется всеми этими коэффициентами как:

$$h(x) = a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + ... + a_1 x + a_0 .$$

Частные ключи (тени) вычисляются по формуле:

$$k_i = h(x_i),\ i = 1, 2, ..., n,\ x_i \in GF(p),$$

например, можно взять $x_i = i,\ n < p$.

Затем k_i передаются каждому из n-пользователей по секретным каналам.

Если m (или более) пользователей $i_1, i_2, ..., i_m$ объединят свои индивидуальные ключи $k_{i_1}, k_{i_2}, ..., k_{i_m}$, то они смогут восстановить полином $h(x)$, используя интерполяционную формулу Лагранжа [14]:

$$h(x) = \sum_{S=1}^{m} k_{i_S} \prod_{\substack{j=1 \\ j \neq S}}^{m} \frac{(x - x_{i_j})}{(x_{i_S} - x_{i_j})} \bmod p .$$

Тогда основной ключ легко может быть найден как

$$a_0 = h(0) = \sum_{S=1}^{m} k_{i_S} C_S \bmod p ,$$

где $$C_S = \prod_{\substack{S'=1 \\ S' \neq S}}^{m} \frac{x_{i_{S'}}}{x_{i_S} - x_{i_{S'}}}.$$

Если же число теней оказывается менее m, то они не будут содержать никакой информации о коэффициенте a_0, поскольку для любого значения $a_0 \in GF(p)$ всегда найдется полином $(m-1)$-й степени $\tilde{h}(x)$, удовлетворяющий уравнениям: $\tilde{h}(x_i) = k_i,\ i = 1, 2, ..., m-1$.

Пример 5.1. Пусть требуется создать пороговую схему (5,3), предназначенную для пяти пользователей, в которой для восстановления основного секрета требуется три или более теней.

Тени получаются с помощью вычисления многочлена в пяти различных точках. В частном случае первой тенью может быть значение многочлена при $x = 1$, второй тенью – значение многочлена при $x = 2$ и т. д.

Пусть ключ k равен 13. Чтобы создать пороговую (5, 3)-схему, в которой любые три из пяти пользователей смогут восстановить k, сначала выберем простое число $p = 17$ (оно больше количества теней и больше основного секрета в данном примере). Предположим, что числа 2 и 10 оказались случайно выбранными коэффициентами полинома, который в этом случае будет равен

$$h(x) = 2x^2 + 10x + 13.$$

Пятью тенями тогда оказываются следующие числа:

$$k_1 = h(1) = (2 + 10 + 13) \bmod 17 = 8;$$
$$k_2 = h(2) = (8 + 20 + 13) \bmod 17 = 7;$$
$$k_3 = h(3) = (18 + 30 + 13) \bmod 17 = 10;$$
$$k_4 = h(4) = (32 + 40 + 13) \bmod 17 = 0;$$
$$k_5 = h(5) = (50 + 50 + 13) \bmod 17 = 11.$$

Эти тени распределяются среди пяти участников протокола. Допустим, 1-й, 3-й и 5-й из них, собрав свои тени, хотят восстановить основной секрет. Используя интерполяционную формулу Лагранжа, они находят:

$$h(x) = \left[\frac{8 \cdot (x-3) \cdot (x-5)}{(1-3) \cdot (1-5)} + \frac{10 \cdot (x-1) \cdot (x-5)}{(3-1) \cdot (3-5)} + \frac{11 \cdot (x-1) \cdot (x-3)}{(5-1) \cdot (5-3)} \right] \bmod 17.$$

Произведя все вычисления по модулю 17, получаем: $h(x) = 2x^2 + 10x + 13$. Свободный член в полученном полиноме 13 и есть восстановленный основной секрет.

Основное преимущество пороговой схемы на основе интерполяционных полиномов Лагранжа состоит в том, что при появлении новых пользователей не надо менять основной секрет, и если он является ключом, на

котором зашифрованы некоторые данные, то их не надо перешифровывать, а достаточно лишь вычислить для основного ключа дополнительные индивидуальные секреты.

Имеется множество различных обобщений [14] для пороговых схем разделения секретов, например:

- взвешенные секреты распределения порогов по группам пользователей;
- схемы с обнаружением фальсификации отдельными пользователями своих частных данных.

Еще одним интересным обобщением схемы с разделением секретов является так называемая (n, m, m_0) *рамп-схема*, в которой объединение m и более пользователей однозначно восстанавливает секрет, при объединении теней s пользователей в интервале $m_0 \le s < m$ основной секрет восстановливается лишь частично, а при числе объединенных пользователей меньшем чем m_0, восстановить его оказывается невозможно.

5.2.2. Доказательства с нулевым разглашением

С помощью протоколов из данного семейства один из пользователей компьютерной сети может доказать другому пользователю, что у него имеется некоторая информация, не раскрывая самой информации.

Например, один из участников доказал гипотезу Гольдбаха о том, что каждое четное целое число больше двух можно представить в виде суммы двух простых чисел, но опасаясь плагиата, он не хочет выдать проверяющему технические детали доказательства, тем не менее хочет убедить его, что доказательство, выполнено им математически корректно.

Доказательства с нулевым разглашением обычно принимают форму интерактивных протоколов. *Проверяющий* (V) задает *доказывающему* (P) ряд вопросов. Если P знает секрет, то он ответит на все вопросы V правильно. Если же секрет ему не известен, то у него есть лишь некоторая вероятность (обычно 0,5) ответить правильно. Тогда после значительного количества вопросов, V сможет достоверно убедиться, знает ли P секрет. Однако ни один из заданных V вопросов и ответов P на них не должен дать V ни малейших сведений об информации, которой обладает P.

Предположим, что P известна некоторая информация, которая является решением трудной проблемы. Базовый протокол нулевого разглашения состоит обычно из нескольких раундов следующего типа.

1. P использует свою информацию и случайное число для преобразования основной трудной проблемы в другую, изоморфную ей проблему. Затем он использует свою информацию и известное ему случайное число для решения новой трудной проблемы.

2. P передает V решение новой проблемы, используя протокол 4 *поручительства информации*.

3. P объясняет V сущность новой трудной проблемы, однако V не может использовать это знание для получения какой-либо информации о первоначальной проблеме или путях ее решения.

4. V просит P об одном из двух:

а) доказать ему, что новая и старая проблемы изоморфны;

б) открыть решение, полученное V на этапе шаге 2, и доказать, что это действительно решение новой проблемы.

5. P исполняет просьбу V.

6. P и V повторяют *n* раз шаги 1–5.

Математическое доказательство того факта, что протоколы подобного типа, действительно, обеспечивают отсутствие утечки к V какой-либо информации о решении основной проблемы и, кроме того, дают надежную гарантию для V, что P имеет такое решение, весьма сложно. Сами проблемы и случайное изоморфное преобразование должны выбираться осторожно, чтобы V не получил никакой информации о решении основной проблемы даже после многих повторений шагов протокола. Заметим, что далеко не все трудные проблемы можно использовать для доказательств с нулевым разглашением, но многие из них допускают это. Рассмотрим в качестве иллюстрирующего примера (который сам по себе не имеет важного практического применения) использование в качестве такой трудной задачи нахождение так называемого *гамильтогового цикла* на заданном графе.

Напомним сначала, что гамильтоновым циклом (ГЦ) называется замкнутая непрерывная линия на графе, которая проходит по ребрам графа через все его вершины только один раз, за исключением исходной вершины. Не каждый граф содержит ГЦ, и до сих пор не известно общих полиномиально сложных методов установления этого факта, а тем более нахождения самого этого цикла, даже если он существует.

Если два графа идентичны во всем, кроме наименования точек, то они называются *топологически изоморфными*. Кроме природы элементов графы, обладающие матрицами инциденций, могут различаться лишь способом упорядочивания ребер. Два графа будут изоморфны друг другу тогда и только тогда, когда их матрицы инциденций можно преобразовать друг в друга посредством перестановки строк и столбцов. Для графов очень больших размеров доказательство их изоморфности может потребовать много компьютерного времени. Это одна из так называемых *NP-трудных* проблем в теории сложности решений [9].

Предположим, что некоторый граф *G* известен как P, так и V, пусть также P удалось каким-то образом найти на нем ГЦ и он хочет доказать этот факт V, не выдавая ГЦ. Тогда рассмотренный выше протокол доказательства с нулевым разглашением принимает для данной задачи следующий вид:

1. P случайным образом переставляет нумерацию вершин графа G, получая другой граф H, который будет, очевидно, изоморфен G. Так как P знает этот изоморфизм и ему известен ГЦ на графе G, он легко находит такой же ГЦ на графе H. С другой стороны, если бы P не знал ГЦ на графе G, то он бы не смог в обозримое время найти его и на графе H. Более того, если бы P знал ГЦ на каком-то другом графе $\tilde{G}$, то он бы не смог доказать в обозримое время, что $\tilde{G}$ и G топологически изоморфны (даже если бы это было верно), поскольку доказательство этого факта для произвольных графов является, как уже отмечалось ранее, трудной задачей.

2. P посылает V копию графа H.

3. V просит P выполнить одно из двух:

а) доказать, что H и G изоморфны;

б) показать ГЦ на H.

4. P исполняет его просьбу и делает одно из двух:

а) доказывает, что H и G изоморфны, не показывая ГЦ на G или H;

б) показывает ГЦ только на H, не доказывая изоморфизма G и H.

5. P и V повторяют n раз шаги протокола 1–4.

Если P знает ГЦ на графе G, то он сможет правильно выполнить задания V на каждой из n итераций. Если же P этого не знает, то он не сможет выполнить требования V на всех шагах. Лучшее, что сможет сделать нечестный P, это построить такой граф $\tilde{H}$, который будет или изоморфным G, или имеющим известный ему ГЦ. Очевидно, что у P оказывается только 50 % шансов угадать, какое доказательство потребует от него V на шаге 3. Таким образом, задавая достаточно большое количество итераций n, можно обеспечить надежное разоблачение нечестного P.

С другой стороны, этот протокол не дает V никакой информации, помогающей ему из ответов P установить ГЦ графа G, поскольку P для каждого нового раунда протокола генерирует новый граф H.

Видно, что протокол, рассмотренный выше, требует обмена информацией между P и V, поэтому протоколы такого типа называются *интерактивными.* Достаточно неожиданным является тот факт, что протоколы с нулевым разглашением не обязательно должны быть интерактивными. Это означает, что P публикует все необходимые данные заранее, и каждый пользователь имеет возможность проверить, что P обладает некоторыми секретными знаниями, не получив из этих знаний никакой информации и даже не вступая в диалог с P! Описание *неинтерактивного* протокола для доказательства ГЦ заданного графа можно найти в [15].

5.2.3.Идентификация пользователей при помощи протокола с нулевым разглашением

Широкое распространение интеллектуальных карт для разнообразных коммерческих, гражданских и военных применений потребовало обеспечения безопасности идентификации таких карт и их владельцев. Во многих приложениях главная проблема заключается в том, чтобы при предъявлении интеллектуальной карты оперативно обнаружить обман и отказать нарушителю в допуске, ответе или обслуживании.

Схемы идентификации на основе паролей слабо соответствуют требованиям указанных приложений. Один из существенных недостатков такой идентификации заключается в том, что после того, как доказывающий передаст проверяющему пользователю свой пароль, проверяющий может, используя данный пароль, выдать себя впоследствии за проверяемого пользователя.

Протоколы идентификации на основе симметричных алгоритмов шифрования также имеют свои недостатки. Для работы таких протоколов идентификации необходимо, чтобы проверяющий и доказывающий с самого начала имели бы один и тот же секретный ключ. Следовательно, встает вопрос о распределении и доставке секретных ключей. При исполнении таких протоколов пользователь доказывает знание секретного ключа, производя с его помощью расшифрование запросов. Проверяющий пользователь имеет принципиальную возможность так сформировать запросы, чтобы передаваемые ответы могли бы быть обработаны с целью извлечения из них дополнительной информации о секретном ключе, с последующим возможным раскрытием этого ключа [15].

Широко известны способы идентификации на основе использования цифровой подписи (разд. 4). Преимущество идентификации на основе использования доказательств с нулевыми знаниями над остальными способами идентификации (в частности и на основе ЦП) заключается в том, что в ходе ее выполнения никакой информации о секретном ключе не «утекает» к проверяющему и ко всем посторонним наблюдателям, в то время, как, например, в случае идентификации на основе ЦП в ней присутствует информация о секретном ключе подписи, которую, хотя и вычислительно сложно, но принципиально можно найти. Кроме того, алгоритмы выполнения и проверки ЦП содержат в себе сложные операции модульного возведения в степень больших чисел, требующие при их программной реализации больших ресурсов процессорного времени, тогда как в алгоритме идентификации с нулевым разглашением (НР) применяются гораздо более простые модульные математические операции (возведение в квадрат и умножение), что позволяет значительно снизить требования к вычислительным ресурсам верификации.

Проверяющий может использовать концепцию HP для доказательства того, что некоторый пользователь обладает определенным секретным ключом, однозначно его идентифицирующим, и при этом избежать утечки какой-либо информации об этом ключе. Рассмотрим далее пример построения подобного протокола [15].

Пусть секретным ключом пользователя будет являться цепочка чисел $\overline{C} = (C_1, C_2, ..., C_k)$, где $1 \le C_j < n$, а $n = p \cdot q$ – модуль, где p и q большие простые числа, причем координаты $\overline{C}$ удовлетворяют уравнениям:

$$d_j \cdot C_j^2 = (\pm 1) \bmod n, \quad j = 1, 2, 3, ..., k, \tag{5.1}$$

где $d_1, d_2, ..., d_k$ – открытый идентификатор пользователя.

Предварительно некоторый центр формирует число $n = p \cdot q$, где p и q большие простые числа, причем $p = 3 \bmod 4$ и $q = 3 \bmod 4$; далее необходимость в обращении к центру для выполнения протокола идентификации отпадает.

Проверяющий (V) знает число n и знает, что секретный идентификатор определенного пользователя $\overline{C}$ удовлетворяет уравнению (5.1), но сам этот идентификатор для него неизвестен. При выполнении протокола идентификации V, по предъявленным ему данным, должен убедиться, что их автор имеет в своем распоряжении $\overline{C}$, но не получить больше об этом векторе никакой *дополнительной информации*. Однако вычислительная мощность V должна быть ограничена, поскольку в противном случае, V сможет найти $\overline{C}$ еще до выполнения протокола и следовательно выдавать себя в дальнейшем за пользователя P.

Протокол HP состоит из четырех шагов:

1. Проверяемый пользователь (P) генерирует случайное число r и вычисляет число

$$x = r^2 \bmod n. \tag{5.2}$$

Далее он посылает это число V.

2. V генерирует случайное подмножество $S \subseteq (1, 2, ..., k)$ и отправляет его P.

3. P вычисляет

$$T_c = \prod_{j \in S} C_j \bmod n \tag{5.3}$$

и посылает V число

$$y = r \cdot T_C \bmod n. \tag{5.4}$$

4. V вычисляет

$$X = y^2 \cdot T_d \bmod n, \tag{5.5}$$

где

$$T_d = \prod_{j \in S} d_j \bmod n, \tag{5.6}$$

и проверяет равенство

$$x = \pm X \bmod n. \tag{5.7}$$

Если равенство (5.7) выполняется, то производится следующая итерация (то есть повторение протокола с новыми случайными данными). Если же равенство (5.7) не выполняется, то проверяемый пользователь P не идентифицируется V.

Если справедливо равенство (5.1), (т. е. P действительно является обладателем секретного вектора $\bar{C}$), то уравнение (5.7) всегда выполняется.

Действительно,

$$X = y^2 \cdot T_d = r^2 \cdot T_c^2 \cdot T_d = r^2 = \pm x \bmod n. \tag{5.8}$$

Умножение случайного числа r на T_c на шаге 3 действительно необходимо, поскольку иначе, выбирая на каждой итерации подмножество $S = \{j\}$, т. е. на первой итерации $S = \{1\}$, на второй – $S = \{2\}$, и т. д. до $S = \{k\}$, V сможет вычислить весь секретный идентификатор $C_1, C_2, ..., C_k$.

Стойкость данной системы идентификации базируется на невозможности извлечения квадратных корней по $\bmod n$ при неизвестной факторизации n. Никакой другой информации о векторе $\bar{C}$ проверяющий V в процессе выполнения протокола не получает. (Предполагается, что всегда выполняется условие $\gcd(C_j, n) = 1$, поскольку в противном случае модуль n может быть легко факторизован.) С другой стороны, существует только один способ для P обмануть V, обеспечив себе идентификацию при отсутствии у него чисел C_j. Этот способ заключается в угадывании подмножества S и формировании $x = \pm r^2 \cdot T_d \bmod n$ и $y = r$. Тогда равенство $x = y^2 \cdot T_d \bmod n$ будет всегда тривиально выполняться. Вероятность угадывания множества S со стороны P равна 2^{-k} и тогда вероятность обмана при выполнении t итераций будет равна $2^{-k \cdot t}$. Соответственно, чем больше число таких итераций, тем меньше вероятность обмана со стороны P при выполнении такого протокола.

Заметим также, что дополнительные условия на простые множители n нужны для того, чтобы V мог бы быть уверен, что числа C_j, соответствующие числам d_j, существуют.

Пример выполнения процедуры идентификации. Пусть центр предоставил пользователю модуль $n = 19 \cdot 31 = 589$. Предположим, что идентифицируемый пользователь сгенерировал секретный идентификатор $\bar{C} = (90, 544, 460, 263, 567)$. Открытый идентификатор $d(p)$ вычисляется как решение уравнения (5.1). Это уравнение решается путем нахож-

дения обратных элементов к C_j^2 по модулю n, что дает, как легко проверить, вектор $\overline{d} = (472, 121, 253, 283, 359)$.

Выполним одну итерацию идентификации:

1. P генерирует случайное число $r = 859$, вычисляет $x = 859^2 \bmod 589 = 453$ и посылает его V.

2. V генерирует множество $S = (1, 4, 5)$ и отправляет его P.

3. P вычисляет $T_c = \prod_{j \in S} C_j = 90 \cdot 567 \cdot 263 = 13420890$ и посылает V число $y = 859 \cdot T_c \bmod 589 = 390$.

4. V находит $T_d = \prod_{j \in S} d_j = 472 \cdot 359 \cdot 283 = 47953784$, затем вычисляет $X = 390^2 \cdot 47953784 \bmod 589 = 453$, и проверяет равенство (5.7): $x = X$. Так как последнее равенство выполнилось, V делает вывод об успешной идентификации P на данной итерации.

5.2.4. Поручительство информации

Напомним сущность этого КП. Пользователь сетиA отдает на хранение (*поручительство)* пользователю B некоторую битовую цепочку данных M (в частном случае цепочку длины 1, т. е. один бит).B не может прочитать эту цепочку до определенного момента времени, задаваемогоA, однако и A не сможет позже изменить содержание цепочки M, хранящейся у B.Данная задача решается следующим простым протоколом:

1. B генерирует случайную битовую цепочку R и посылает ее по сети к A.

2. A объединяет Rсо своим сообщением M и посылаетB криптограмму $E = f_K(R, M)$, где $f_K(...)$ – алгоритм шифрования некоторым стойким блоковым шифром на секретном ключе K.

3. B хранитE до поступления команды от A.

4. В определенное времяA посылает B свой секретный ключ K, при помощикоторого B дешифрует E.

5.B проверяет присутствие своей цепочки R в дешифрованном сообщении.

Очевидно, что все задачи протокола решаются после выполнения данных шагов.

Для решения той же задачи можно использовать другой протокол, который не требует выполнения процедур шифрования/дешифрования и активности от пользователяB:

1. A генерирует случайные числа R_1, R_2.

2.A объединяет их со своим сообщением M, формируя цепочку $E = (R_1, R_2, M)$.

3. A выполняет хеширование E при помощи однонаправленной бесключевой хеш-функции $h(...)$, т. е. вычисляет $h = h(E)$.

4.A посылаетBцепочку h, R_1.

5.В определенное времяA посылает B цепочку $E = (R_1, R_2, M)$.

6.Bубеждается в правильности сохраняемого им сообщения M, выполняя хеширование цепочки E.

Безопасность данного протокола дляB определяется тем обстоятельством, что A не может найти другую такую цепочку (R_1, R'_2, M'), для которой $h(R_1, R'_2, M') = h(R_1, R_2, M)$.(Если бы A не посылалB цепочку R_1, то A имел бы возможность изменить обе величины R_1, R_2, а затем подделать сообщение M.)

5.2.5. Обманчивая передача

Реализация данного протокола, отвечающая всем требованиям его безопасности, в том числе и при активности участников, которые могут отклоняться от предписанных шагов протокола с целью обмана друг друга, слишком сложна для описания в этой книге.

Поэтому ограничимся лишь кратким описанием данного протокола для пассивных участников, строго следующих шагам протокола (обобщение на случай активных участников можно найти в [15]).

Пусть участник протокола A имеет k секретов $s_1, s_2, ..., s_k$, где каждый из секретов s_i представляет из себя цепочку бит произвольной длинны. Предполагается, что A анонсирует название этих секретов (скажем, «коррупционная сделка в N-ской компании»), а само содержание секретов сохраняется в тайне. B хочет купить один из этих секретов, предположим s_i (к другим у него нет интереса или недостаточно денег для покупки), однако он хочет это сделать так, чтобы владелец секретов A не узнал, что именно интересует B (иначе A может, например, предупредить компанию о возможной проверке). Тогда протокол может быть выполнен следующими шагами:

1. A передает B однонаправленную функцию (например, функцию шифрования РША $x^e \bmod n$, сохраняя в секрете p, q и d.

2. Если B решает купить секрет s_i, то он генерирует k случайных чисел $x_1, ..., x_k$, и посылает B цепочку чисел $y_1, ..., y_k$, где

$$y_j = \begin{cases} x_j, & \text{если} \quad j \neq i \\ f(x_j), & \text{если} \quad j = i \end{cases}.$$

3. *A* находит число $z_j = f^{-1}\left(y_j\right)$, $j = 1, 2, ..., k$ (для РША функциями f будут $z_j = y_j^d \bmod n$, $j = 1, 2, ..., k$) и посылает *B* $\overline{a}_j = \overline{z}_j \oplus \overline{s}_j$, где черта над буквой означает преобразование чисел в цепочку бит, а $\oplus$ – побитовое сложение по mod2.

4. *B*, зная, что $z_j = f^{-1}\left(f\left(x_j\right)\right) = x_i$ находит желаемый им секрет как $\overline{s}_i = \overline{z}_i \oplus \overline{a}_i$.

Видно, что *B* не получает никакой информации о z_j для $j \neq i$ и, следовательно, никакой информации о s_j для $j \neq i$. С другой стороны, у *A* нет никакой возможности отличить случайные числа x_j, $j \neq i$ от случайного числа x_i и поэтому *A* не может определить каков был выбор *B*.

(Очевидно, что активный *B* мог бы послать *A* несколько чисел в форме $f\left(x_j\right)$ и тогда получить от *A* больше секретов. Однако существуют видоизменения протокола, которые могут предотвратить такой обман [15].

5.2.6. Секретные совместные вычисления

Заметим, что в настоящее время подобные протоколы преобретают особое значение в связи с развитием в сети так называемых «облачных технологий» (т. е. использования удаленных вычислительных средств).

Общая постановка задачи для двух пользователей состоит в следующем: *A* имеет некоторую секретную величину *x*, *B* – секретную величину *y*, требуется послать на «облако» *C* некоторые данные от *A* и *B*, которые бы позволили рассчитать величину $f\left(x, y\right)$, где $f\left(.,.\right)$ – известная всем участникам функция и результат переслать *A* и *B*, но так, чтобы личные секреты *x* и *y* не были бы раскрыты не имеющим их участникам, включая *C*.

Свойства и описание таких протоколов для некоторых функций $f\left(\cdot\right)$ можно найти в [14, 15]. Однако поскольку это требует большого объема материала и выходит за рамки курса «Основы криптографии», мы ограничимся простейшим примером частного вида протокола, когда четверо участников *A*, *B*, *C*, *D*, имея свои секретные данные x_1, x_2, x_3, x_4, хотят вычислить функцию $f\left(x_1, x_2, x_3, x_4\right) = \left(x_1 + x_2 + x_3 + x_4\right)/4$. (Например, рассчитать среднюю зарплату четырех лиц, не выдав друг другу свои личные заработки). Предположим также, что все участники имеют какие-то связи друг с другом (или, хотя бы, $A \to B$, $B \to C$, $C \to D$, $D \to A$), а в роли участника анонсирующего результат, будем считать *A*. (Очевидно, что данный протокол легко обобщается для любого количества участников. Поэтому легко можно подсчитать «среднюю температуру по больнице», не

вскрывая секрета, скольких пациентов лихорадит, а сколько из них уже скончалось).

Описанный выше протокол для четырех участников реализуется следующими шагами:

1. A посылает B число $y_A = f_{e_B}\left(x_A \oplus r_A\right)$, где x_A – секретное число A, r_A – случайное число, e_B – открытый ключ B, $f\left(\cdot\right)$ – функция шифрования в некоторой криптосистеме с открытым ключом (например, в РША), $\oplus$ – операция сложения по $\bmod n$, где n – выбрано для криптографической системы с открытым ключом.

2. B посылает C число $y_B = f_{e_C}\left(g_{d_B}\left(y_A\right) \oplus x_B\right)$, где e_c – открытый ключ C, $g_{(\cdot)}\left(\cdot\right)$ – функция дешифрования для выбранной криптосистемы, x_B – секретное число B.

3. C посылает D число $y_D = f_{e_D}\left(g_{d_C}\left(y_B\right) \oplus x_C\right)$, где x_C – секретное число C.

4. D посылает A число $y_D = f_{e_A}\left(g_{d_D}\left(y_C\right) \oplus x_D\right)$, где x_D – секретное число D.

5. A оповещает всех участников о полученном результате

$$\overline{S} = \frac{1}{4}\left(g_{d_A}\left(y_D\right) \,!\, r_A\right) = \frac{1}{4}\left(x_A \oplus r \oplus x_B \oplus x_C \oplus x_D \,!\, r\right) = \frac{1}{4}\left(x_A \oplus x_B \oplus x_C \oplus x_D\right) =$$

$$= \frac{1}{4}\left(x_A + x_B + x_C + x_D\right),$$

где $!$ – операция вычитания по $\bmod n$; + обозначает арифметическое сложение. (Заметим, что последнее равенство будет верно, если $x_A \oplus x_B \oplus x_C \oplus x_D < n$.)

Из описанного протокола видно, что личные секреты A, B, C, D не разглашаются посторонним, а конечное число $\overline{S}$ соответствует желаемому результату. В [14] приведены примеры выполнения протоколов, которые позволяют вычислить функцию $\overline{i} = Arg\max x_i$, $i = 1, 2, ..., n$. (Например, вычислить, кто является старшим в группе из n лиц, не выдавая возраст каждого из них.) Другим примером протокола, представленного в [14], является распределение группы пользователей «по интересам», когда принадлежность к одной группе раскрывается лишь для ее членов, но не для членов других групп.

5.2.7. Цифровая наличность

Опишем кратко протокол, который реализует все требуемые свойства ЦН (разд. 5.1), за исключением требования 4 (режим off-line).

Такой протокол реализуется следующими шагами:

1. Пользователь A, желающий получить ЦН адекватную реальной сумме l_A, имеющейся на его счете в банке B, генерирует случайно цепочку x большой длины, «ослепляет ее», зашифровывает результат своим секретным ключом и посылает в банк B для его подписи. («Ослепление» цепочки x при использовании в ЦП РША означает вычисление $x' = x \cdot r^{e_B(l_A)} \bmod n$, где r – случайное число, и $e_B(l_A), n$ – открытый ключ B, соответствующей номиналу l_A).

2. B подписывает x' («вслепую» относительно x), т. е. выполняет операцию ЦП $= S = (x')^{d_B(l_A)} \bmod n$. При этом предполагается, что B предварительно расшифровал данные, полученные от A на шаге 1 «его» открытым ключом e_A, убедившись, что это действительно A, которая имеет счет в этом банке, не меньший чем l_A и поэтому он подписывает x' ЦП $d_B(l_A)$, зависящей от номинала l_A. Вместе с тем, B не знает числа $x \neq x'$, ввиду «ослепления» x. Дальше B отсылает S к A, уменьшая размер ее счета на l_A.

3. A снимает «ослепление» с S, выполняя операцию

$$S' = S / r = \frac{(x')d_B(l_A)}{r} = \frac{\left(x \cdot r^{e_B(l_A)}\right)^{d_B(l_A)}}{r} = x^{d_B(l_A)} (\bmod n)$$ и предъявляет для покупки в магазин (M) пару цифр (x, S').

4. Продавец M проверяет ЦП банка и ее соответствие заявленному номиналу l_A и отсылает (x, S') в свой банк C.

5. C перепроверяет ЦП банка B, проверяет в своем реестре, нет ли в нем уже потраченной ЦН (т. е. такой же пары (x, S') и если нет, то он увеличивает счет продавца M на l_A и отсылает ему извещение о подтверждении платежа.)

6. M отпускает A товар на величину номинала l_A.

Видно, что такой протокол обеспечивает анонимность покупателя, если, конечно, не принимать во внимание возможность отследить его электронный адрес по интернету или сфотографировать покупателя при личной покупке. (Впрочем, в первом случае покупателя может защитить анонимный канал (подразд. 5.2.9), а от второго не защищает даже и бумажная наличность.)

Однако при переходе из режима on-line (описанного выше) в режим off-line, на момент покупки невозможно иметь список всех израсходованных цепочек (x, S'). Ключевая идея решения этой проблемы состоит в последующем отслеживанием (с задержкой) нечестных покупателей, а также продавцов. Полное описание такого расширенного протокола можно найти в [11].

5.2.8. Неоспоримое шифрование

Данная задача решается тривиально, если используется идеальный шифр (одноразовое шифрование), т.е. когда криптограмма формировалась по правилу: $E = M \oplus K$, где K – чисто случайный двоичный ключ такой же длины как и криптограмма (подразд. 2.1).

В этом случае для любого «несекретного» сообщения M' можно иметь в запасе ключ $K' = E \oplus M'$, который выдается за истинный при принуждении.

Очевидно, что никакими криптографическими средствами нельзя доказать, что этот ключ не является истинным.

Однако маловероятно, что идеальный шифр будет использоваться для шифрования больших объемов данных. Поэтому в последние годы появилось много публикаций [26 и др.], где описывается решение подобной задачи для шифров, использующих рандомизацию при шифровании. В этом случае принуждаемый к раскрытию ключа может представить алгоритм дешифрования, который приведет имеющуюся криптограмму к «несекретному» сообщению.

5.2.9. Тайное голосование (ТГ)

Напомним основные требования, предъявляемые обычно к процедуре тайного голосования:

1) участвовать в выборах могут только те легитимные пользователи, которые были предварительно включены в списки, составленные *избирательной комиссией* (ИК);

2) результат голосования каждого легитимного участника должен сохраняться в тайне от всех участников процедуры голосования (в том числе от других легитимных пользователей, ИК и всех посторонних), за исключением, конечно, самого подавшего голос;

3) результат голосования каждого участника должен быть правильно учтен ИК и учтен лишь единожды;

4) при отсутствии легитимного избирателя в списке ИК или при неправильном учете результатов голосования в ИК, эти ошибки должны быть исправлены без нарушения тайны голосования;

5) ИК не может сфальсифицировать результаты тех легитимных избирателей, которые не захотели принять участие в голосовании.

Заметим, что для выполнения требований 3–5 необходимо участие самих избирателей, поскольку никто другой не заинтересован больше них в том, чтобы узнать, приняли ли они участие в голосовании, и правильно ли учтен их голос.

Рассмотрим сначала *принципиальные трудности*, возникающие при реализации протокола тайного голосования.

Ясно, что наиболее сложной для выполнения частью протокола при использовании только лишь телекоммуникационных каналов является сохранение тайны результата голосования как для всех посторонних пользователей сети (включая и легитимных избирателей), так и для ИК.

Защита результатов тайного голосования от перехвата со стороны всех пользователей, кроме ИК, может быть решена достаточно просто – путем передачи этих данных в ИК в зашифрованном виде. Однако это не решит проблему относительно ИК, поскольку она должна знать результаты голосования каждого легитимного участника, но она не должна быть в состоянии идентифицировать эти данные, связав их с именами пользователей. Для решения этой проблемы есть лишь одна возможность – использование так называемого *анонимного канала связи*. Это означает организацию такого телекоммуникационного канала, который в точности передает некоторые данные, но не позволяет определить адреса отправителей этих данных. Решение этого вопроса возможно при использовании:

- *ретранслятора*, обеспечивающего анонимность пересылки;
- *протокола анонимного вещания.*

Эти два способа имеют существенные недостатки. Первый требует участия в протоколе доверенных «третьих лиц», которые не должны раскрыть результаты голосования кому бы то ни было. Второй способ (описанный в [14]) требует специальной организации вещательного канала и выполнения весьма сложного и требующего большего времени протокола.

Общеизвестным вариантом решения является использование так *называемых MIX-серверов* [17]. В упрощенном варианте этот протокол имеет следующий вид:

1) пользователь A_i, который желает отправить сообщение M_i пользователю B_j так, чтобы нельзя было бы установить связь между A_i и B_j для всех, кроме этих пользователей, генерирует случайные числа $R_1, ..., R_k$ и отправляет криптограмму $C_i = f_{e_{S_1}}\left(R_1 \| f_{e_{S_2}}\left(R_2 \| ... f_{e_{S_k}}\left(R_k \| B_j \| f_{e_{B_j}}\left(M_i\right)\right)...\right)\right)$ первому серверу S_1, где $f_{(e)}(M)$ – функция шифрования в криптосистеме с открытым ключом e сообщения M, $\|$ – означает конкатенацию (объединение) нескольких сообщений;

2) сервер S_1 дешифрует криптограмму своим секретным ключом d_{S_1}, отбрасывает случайное число R_1 и полученное сообщение $f_{e_{S_2}}\left(R_2 \| ... f_{e_{S_k}}\left(R_k \| B_j \| f_{e_{B_j}}\left(M_i\right)\right)...\right)$ посылает на второй сервер S_2;

3) далее шаг 2 повторяется для 2-го, 3-го,..., k-го серверов, что позволяет k-му серверу опубликовать на открытом сайте сообщение $B_j \| f_{e_{Bj}}(M_i)$. (Если от A_i и других пользователей поступают сообщения предназначенные также другим пользователям $B_{j'}$, $j' \neq j$, то k-й сервер размещает их на сайте в лексикографическом порядке, что и оправдывает его название MIX сервера.);

4) пользователь B_j дешифрует криптограмму своим секретным ключом d_{B_j}, после чего он может также узнать и адрес отправителя A_i. Присутствие случайных чисел $R_1, R_2, ..., R_v$ необходимо для того, чтобы исключить возможность для посторонних дешифровать криптограммы при малом количестве сообщений (подразд. 3.1.5).

Легко проверить, что для описанного выше протокола никто кроме пользователей A_i и B_j не может установить связь между A_i и B_j, если хотя бы один из k серверов является «честным», т. е. исправно следует протоколу. Очевидно, что если все серверы не состоят в сговоре, то чем больше k, тем меньше вероятность нарушения анонимности.

В работе [27] описаны упрощенные протоколы, которые позволяют сократить длину криптограммы и упростить вычисления.

Однако даже если анонимный канал связи между избирателями и существует, это не решает полностью задачу выполнения протокола тайного голосования. Дело в том, что, принимая результаты голосования, ИК должна убедиться, что они присланы от легитимных избирателей, однако подтверждение их легитимности не должно идентифицировать самих избирателей. Это, казалось бы, неразрешимое противоречие, тем не менее, может быть решено с помощью выполнения протокола *слепой подписи,* описанной подробно в [14] и применимо к задаче тайного голосования, рассмотренной в следующем разделе. Кроме того, необходимо обеспечивать выполнение важного требования 3, которое может быть, в свою очередь, разбито на 2 требования:

- каждый легитимный участник не должен проголосовать более одного раза;
- голос каждого легитимного участника должен быть учтен (т. е. не может быть удален) ИК.

Что же касается требования 4, то оно оказывается менее жестким, поскольку реализуется при помощи выполнения повторной процедуры голосования, которая, однако, не должна открыть голоса его участников.

Наконец, принципиальные трудности могут возникнуть при выполнении требования 5, когда ИК может попытаться сфальсифицировать выгодные ей результаты голосования для легитимных, но не принявших участие в голосовании, избирателей. Если во время выполнения протокола избиратель не известит

об отказе продолжить голосование, то ИК может попытаться воспользоваться его голосом. Это для нее легко выполнить, так как именно сама ИК выдает разрешения на голосование. Поэтому в протоколе предусмотрены некоторые процедуры для сокращения возможностей ИК в использовании голосов, отказавшихся участвовать избирателей. Однако полным решением этой проблемы было бы извещение избирателя со стороны ИК об его отказе от голосования.

В следующем разделе описан протокол ТГ по телекоммуникационным каналам связи (ТКК) с использованием техники криптографии с открытым ключом. Стойкость ТГ протокола (т. е. его способность удовлетворять всем перечисленным выше требованиям в условиях возможных недружественных действий) будет полностью определяться стойкостью выбранного алгоритма криптосистемы с открытым ключом. Для упрощения будем в дальнейшем использовать в качестве такого базового метода КС РША.

При выборе соответствующих параметров (подразд. 3.1.7) необходимая стойкость может быть достигнута относительно всех, известных до настоящего времени, методов криптоанализа.

Описание протокола ТГ по ТКК

Весь протокол может быть разбит на три основные части:

1) инициализация протокола (распределение исходных данных);
2) выполнение основного протокола;
3) коррекция основного протокола.

Рассмотрим далее выполнение каждой из этих частей.

Инициализация протокола (рис. 5.2)

На этом этапе выполняются следующие процедуры:

• ИК составляет список легитимных участков голосования и помещает его на общедоступном сайте;

• ИК генерирует тройку чисел $n_{ик}$, $d_{ик}$, $e_{ик}$ (модуль, закрытый /открытый ключ) и публикует открытый ключ на общедоступном сайте в Интернете;

• каждый избиратель генерирует свою пару ключей $d_{и}$; $n_{и}$, $e_{и}$ (закрытый/открытый ключ) и публикует открытый ключ на общедоступном сайте в Интернете.

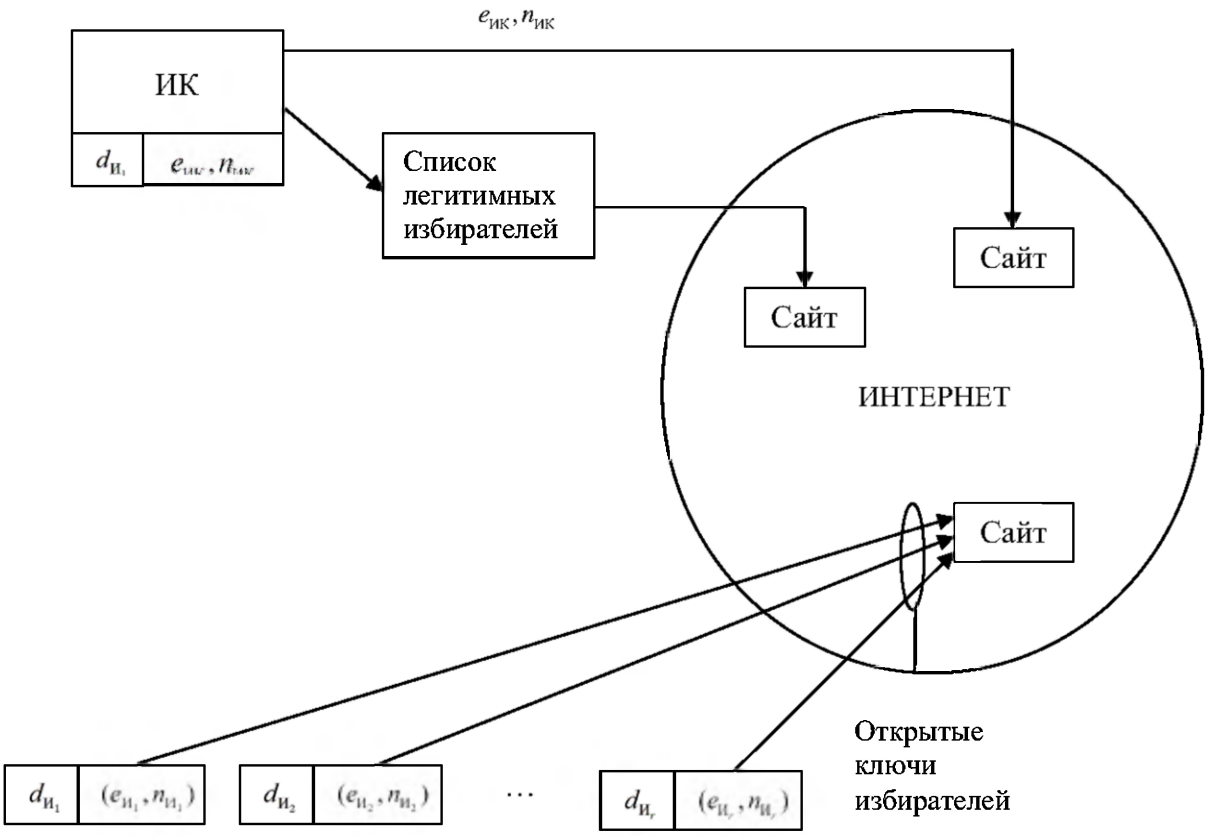


Рис. 5.2. Инициализация протокола тайного голосования

Выполнение основного протокола

Сначала перечислим шаги протокола (рис. 5.3) с повторением инициализации:

1) ИК публикует список всех правомочных избирателей, а также свой открытый ключ;

2) все избиратели генерируют свои ключи и делают открытые ключи общедоступными;

3) избиратель, желающий принять участие в голосовании, генерирует свой *идентификационный номер I* (ИН);

4) избиратель маскирует свой идентификационный номер и отправляет в избирательную комиссию *маскированный* ИН, зашифрованный дополнительно ключом избирателя;

5) ИК расшифровывает полученное сообщение;

6) ИК подписывает «вслепую» маскированный ИН;

7) избиратель демаскирует подписанный ИН;

8) избиратель создает бюллетень с результатом своего голосования, зашифровывает его и отправляет в ИК по *анонимному каналу*;

9) ИК публикует полученное шифрованное сообщение;

10) избиратель отправляет по анонимному каналу ключ для расшифровки сообщения;

11) ИК дешифрует сообщение и публикует результаты голосования.

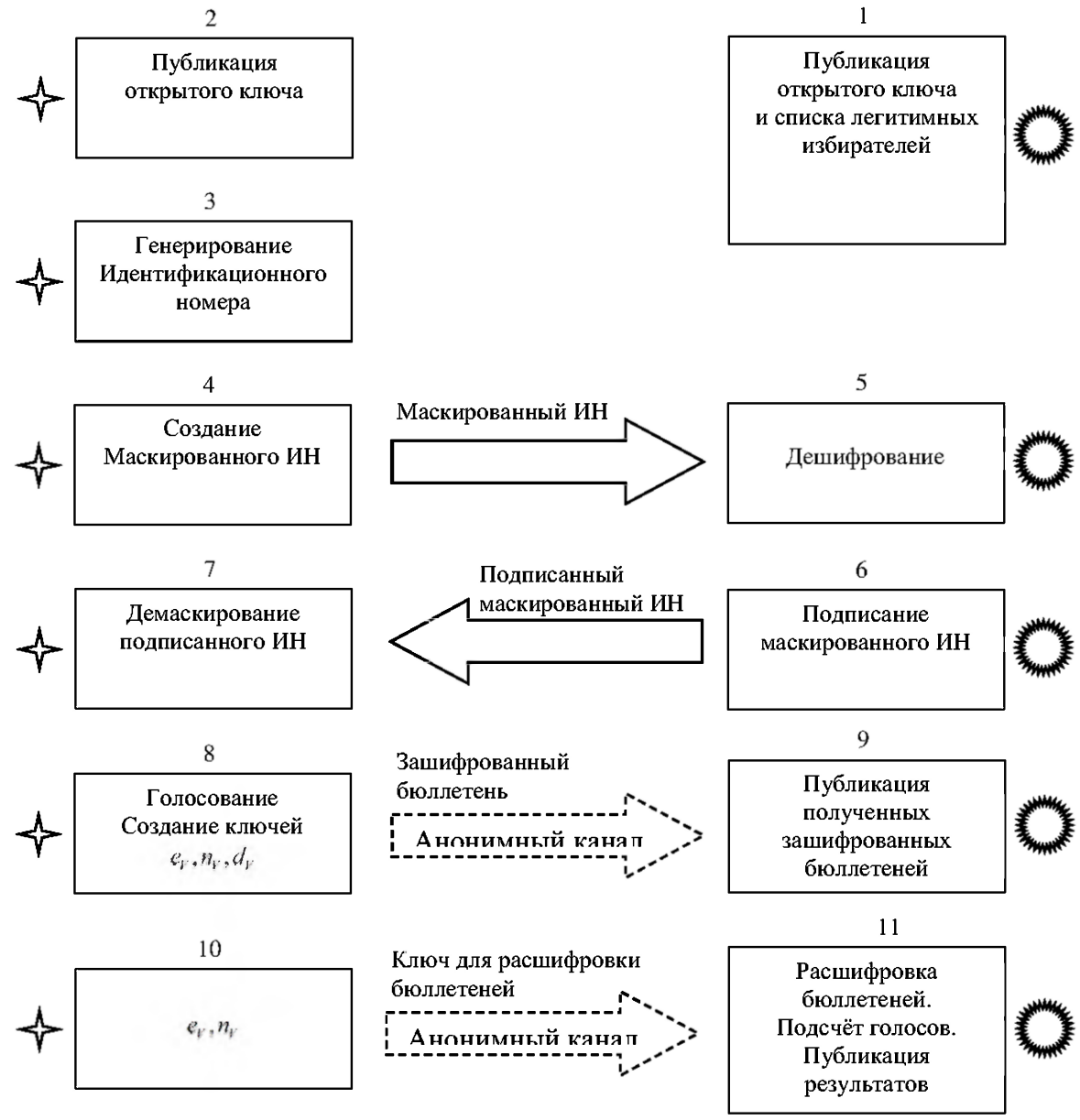


Рис. 5.3. Иллюстрация основных шагов протокола ТГ

Рассмотрим подробно шаги протокола ТГ.

1) ИК публикует на общедоступном сайте в Интернете пронумерованный список легитимных избирателей, подписанный своей цифровой подписью (ЦП), с использованием своего секретного ключа. (Заметим, что сквозная нумерация может быть заменена для удобства на какой-либо другой способ учета избирателей, обеспечивающий их быстрый поиск). В дополнение к этому ИК публикует на общедоступном сайте, заверенные своей ЦП сведения о порядке голосования, о времени выполнения каждого

шага протокола и структуре сообщений, которыми должны обмениваться с ней избиратели при выполнении процедуры голосования.

Рассмотрим теперь как обеспечивается *защита* от возможных недружественных действий участников протокола ТГ на этом шаге:

• *От ИК*. Так как ИК подписывает своей ЦП список избирателей, то она не заинтересована в том, чтобы внести в список нелегитимных избирателей. Также она не заинтересована в удалении из списка легитимных избирателей.

• *От избирателей*. Избиратель не будет отказываться голосовать под предлогом того, что ИК игнорирует легитимных участников (об этом далее).

• *От посторонних*. Если ИК не будет подписывать список легитимных избирателей, то злоумышленник может подменить список. Это приведет к тому, что легитимные избиратели, которые не нашли себя в списке, будут протестовать о не внесении их в список, или часть из них откажется голосовать.

2) Каждый избиратель, который желает принять участие в голосовании, должен найти в списке легитимных избирателей свою фамилию и соответствующий ей номер. (Не найдя себя в этом списке, избиратель должен действовать в соответствии с процедурой коррекции выполнения основного протокола, описанной далее). Если избиратель нашел свою фамилию и соответствующий номер в списке, то он должен сгенерировать случайной идентификацией номер I, который представляет собой цепочку цифр длиной l. Длина I выбирается таким образом, чтобы вероятность случайного выбора одного и того же идентификатора I у двух и более избирателей была бы пренебрежимо малой.

Этот ИН должен быть уникальным, чтобы ИК, получив впоследствии бюллетень с ним, была бы убеждена, что избиратель является одним из тех, кто имеет право на голосование. Однако ни ИК, ни кто-либо другой не должны знать, какой номер I имеется у данного избирателя, поскольку в противном случае результат голосования данного избирателя стал бы им известен.

Идентификационный номер I нужен для того, чтобы ИК знала, что данный избиратель имеет право на голос. В данном протоколе идентификационный номер I создает избиратель и дает ИК подписать его. Подписанный идентификационный номер – это более чем достаточный признак права на голосование. Проблема заключается в том, что ИК не должна знать какой идентификационный номер она подписала, хотя должна быть уверена, что обладателем этого ИН является легитимный избиратель. Эту проблему решает процедура слепой подписи.

3) Существует много способов выполнения подписи «вслепую». Рассмотрим далее один из самых простых. Идентификационный номер I умножается на *маскирующий множитель* (число) m. Однако так как его

нужно будет в дальнейшем «сократить», этот множитель заранее возводится в степень открытого ключа ИК – $e_{ик}$.

Избиратель маскирует сгенерированный им идентификационный номер I, подготавливая его для последующей «слепой» подписи со стороны ИК. Для этого он выполняет следующее преобразование:

$$I_m = m^{e_{ик}} \cdot I \bmod n_{ик}, \tag{5.9}$$

где $e_{ик}$, $n_{ик}$ – открытый ключ ИК, считанный избирателем с общедоступного сайта; m – случайно сгенерированное целое число из диапазона (1, 2, …, $n_{ик}$ – 1), которое является взаимно простым с $n_{ик}$, т. е. при выполнении условия $\gcd(m, n_{ик}) = 1$.

Для того чтобы найти значение ИН по маскированному номеру, ИК должна перебрать всевозможные m. Однако поскольку $n_{ик}$ является очень большим числом, время для нахождения правильного I оказывается нереализуемым.

Посторонний наблюдатель, также как ИК, не знает маскирующего множителя и, значит, не сможет узнать истинное значение I.

4) Для того чтобы ИК подписала каждому легитимному избирателю по одному идентификационному номеру, ИК должна знать, что они являются легитимными. Поэтому каждый избиратель зашифровывает свой номер n и маскированный I (т. е. I_m) своим секретным ключом $d_и$, т. е. он направляет ИК по открытому каналу следующее сообщение:

$$M_1 = \left(n, E_{d_и}\left(n, I_m\right)\right), \tag{5.10}$$

где n – порядковый номер этого избирателя в списке легитимных избирателей, который он считывал с общедоступного сайта; $E_{d_и}\left(n, I_m\right)$ – означает процедуру ассиметричного шифрования сообщений $\left(n, I_m\right)$ с использованием секретного ключа избирателя $d_и$.

В данном сообщении для ИК скрыт только ИН избирателя, тогда как его номер в списке легитимных избирателей известен.

Если злоумышленник попытается отправить сообщение на подпись от имени одного из избирателей, то он не сможет создать подобное сообщение, так как не знает секретного ключа избирателя.

5) ИК публикует все принятые сообщения M_1 на открытом сайте. Далее ИК дешифрует криптограмму, содержащую в сообщении M_1, используя для этого находящийся на общедоступном сайте, открытый ключ избирателя ($e_и$, $n_и$) с номером n. Если значение n, содержащееся в открытой части сообщения M_1 и в расшифрованной криптограмме совпадут, то ИК может быть уверена, что это сообщение действительно получено от n-го избирателя.

Публикация сообщений M_1 необходима для того, чтобы знать, кто из легитимных избирателей не будет голосовать. ИК не сможет воспользо-

ваться голосами тех, кто изначально не принимал участия в голосовании. Если сообщения M_1 не будут опубликованы, то ИК может имитировать избирателей. Так как в сообщении M_1 используется шифрование с секретным ключом избирателя, то подделать это сообщение невозможно. Значит, публикация этих сообщений определит голосующих и удалит те голоса, которыми могла бы воспользоваться нечестная ИК. Это весьма важно потому, что большая часть избирателей, отказавшихся от голосования, не станет генерировать ИН и посылать их в ИК.

6) Если некоторый избиратель является легитимным, то ИК подписывает его маскированный идентификационный номер I_m, выполняя процедуру цифровой подписи РША:

$$I_{sm} = I_m^{d_{\text{ик}}} \bmod n_{\text{ик}}. \tag{5.11}$$

После этого ИК отправляет I_{sm} по открытому каналу к n-му избирателю и помещает (вместе с номером n) на общедоступном сайте.

Даже если бы ИК не хотела этого, ей придется подписать сообщение и отправить его избирателю. При любых других вариантах обман со стороны ИК тривиально раскрывается.

С другой стороны, избиратель не сможет отказаться от того, что он получил подписанные данные.

Хотя подписанный маскированный ИН находится на общедоступном сайте, это не дает возможности определить истинный ИН без знания маскирующего множителя.

7) Избиратели производят демаскирование своих подписанных идентифицированных номеров, выполняя следующие преобразования над полученными от ИК значениями I_{sm}:

$$I_s = \left(I_{sm}\right) / m \bmod n_{\text{ик}}. \tag{5.12}$$

Действительно, подставляя (5.9) и (5.11) в (5.12), получаем:

$$\begin{aligned} I_{sm} / m &= \left(I_m\right)^{d_{\text{ик}}} \bmod n_{\text{ик}} \cdot m^{-1} \bmod n_{\text{ик}} = \left(m^{e_{\text{ик}}}\right)^{d_{\text{ик}}} \cdot I^{d_{\text{ик}}} \cdot m^{-1} \bmod n_{\text{ик}} = \\ &= m \cdot I^{d_{\text{ик}}} \cdot m^{-1} \bmod n_{\text{ик}} = I^{d_{\text{ик}}} \bmod n_{\text{ик}}. \end{aligned} \tag{5.13}$$

После выполнения преобразования (5.13), избиратель проверяет подлинность подписи ИК: $\left(I^{d_{\text{ик}}}\right)^{e_{\text{ик}}} \bmod n_{\text{ик}} = I$, используя известный открытый ключ ИК. Если это равенство не выполняется, то он выражает протест (об этом далее).

Таким образом, все легитимные избиратели после выполнения шага 7 будут иметь свои идентифицированные номера I, подписанные ИК, что и удостоверит их легитимность при выполнении последующих шагов протокола. С другой стороны, ИК и все посторонние не знают ИН избирателей, поскольку он присутствует только в маскированном виде.

8) Избиратели голосуют, включая результаты голосования в сообщение V. После этого каждый из избирателей генерирует пару собственных ключей (e_v, n_v и d_v) для шифрования/дешифрования результатов голосования. Далее каждый из избирателей формирует следующее сообщение:

$$M_2 = \left(P, E_{d_v}\left(I, I_s, V\right)\right), \tag{5.14}$$

где P – это любое число, используемое для удобного поиска M_2 на открытом сайте. (Оно даже может быть одинаковым у различных избирателей.) Избиратель посылает это сообщение по *анонимному каналу* к ИК. Необходимость в шифровании сообщения $\left(I, I_s, V\right)$ объясняется следующим образом. Если бы его передавали в отрытом виде, как, скажем, $M_2 = \left(P, I, I_s, V\right)$, то ИК могла бы удостоверить свою подпись и учесть результат голосования V как легитимный, однако существовала бы опасность *нарушения целостности* сообщения M_2 со стороны злоумышленников, т. е. замены его на $M'_2 = \left(P, I, I_s, V'\right)$, где $V' \neq V$, что соответствовало бы другому результату голосования, причем эта подмена не была бы обнаружена ИК.

Таким образом, ИК не знает, кто послал сообщение M_2, так как избиратели используют анонимные каналы для отправки этих сообщений. Даже если сообщение было послано от нелегитимного избирателя, ИК узнает об этом только после его расшифровки.

Поскольку каждый избиратель имеет только один подписанный ИН, то он сможет проголосовать всего один раз.

Лишь владельцы подписанных ИН смогут создать подобные сообщения, все же остальные сообщения будут на следующих шагах признаны ИК недействительными.

9) ИК публикует M_2 (в зашифрованном виде) на открытом сайте.

10) Избиратели направляют в ИК сообщение M_3 по анонимному каналу в следующем виде:

$$M_3 = \left(P, e_v, n_v\right). \tag{5.15}$$

ИК, получив это сообщение, сможет расшифровать M_2 и, следовательно, учесть голос неизвестного ей, но легитимного (поскольку его идентификатор ею же был подписан) избирателя. Если же этот голос «не устроит» нечестную ИК, то она может попытаться проигнорировать это сообщение. (Решение данной проблемы дано в коррекции к протоколу ТГ, ситуация 4).

Очевидно, что те, кто не имел ИН подписанных ИК, могут послать подобное сообщение, но их голос не будет учтен ИК.

11) ИК дешифрует криптограмму, содержащуюся в M_2, используя для этого ключ дешифрирования e_v, n_v. В результате ИК, получает I, I_s и V, может убедиться в легитимности результата голосования и учесть результат голосования V, представленный теперь в открытом виде, который она

затем опубликует (совместно с ИН) на открытом сайте для проверки правильности учета результата голосования каждым избирателем.

Дополнительные пояснения к протоколу тайного голосования

Использование анонимного канала как на шаге 8, так и на шаге 10 обязательно, поскольку в противном случае, ИК может связать I с конкретным избирателем и, следовательно, нарушить тайну голосования.

Если избиратель выполнил шаг 4, но затем решил не голосовать, то он должен известить об этом ИК. Избиратель создает в этом случае сообщение вида:

$$M_a = \left(n, E_{d_u}\left(n, r, I, I_s\right)\right),$$

где n – номер избирателя; r – признак отказа от голосования (значение его должно быть заранее определено); I – ИН (может отсутствовать в сообщении); I_s – подписанный ИН (может отсутствовать в сообщении).

Избиратель отправляет это сообщение в ИК и, кроме того выбирает по своему усмотрению избирателей, количество которых должно быть заранее определено, и отправляет каждому из них то же самое сообщение M_a. ИК при публикации результатов голосования, публикует дополнительную информацию об отказе избирателей от голосования, если такие имеются. В случае если избиратель уже получил подписанный ИН, то «раскрытие» подписанного ИН становится обязательным, для предотвращения использования данного ИН.

Нечестная ИК не может сфабриковать результат голосования для тех избирателей, которые изначально не участвовали в голосовании, поскольку голосующий избиратель на шаге 4 отправляет сообщение M_1, и тем самым подтверждает желание голосовать. Такое сообщение ИК сам создать не может, так как для этого используется секретный ключ избирателя. Разумно предположить, что большая часть отказавшихся голосовать избирателей не будут голосовать изначально. Однако остается малая часть избирателей, которая выполнит шаг 4 и затем на любом из шагов протокола (за исключением шага 10), откажется продолжать голосование. После шага 4 избиратель обращается к ИК только по анонимному каналу и определить, кто посылает сообщения в ИК невозможно. Это, казалось бы, означает, что нечестные пользователи могут попытаться послать сообщения в ИК, не являясь легитимными участниками. Однако для того чтобы их голоса были бы учтены, необходимы ИН, подписанные ИК. Поскольку ИК может создавать фальсифицированные ИН, то воспользовавшись отсутствием действий избирателя после шага 4, она может попытаться проголосовать за него. Обман ИК будет раскрыт, если большая часть избирателей, отказавшихся от голосования после шага 4, предоставит свои демаскированные подписанные ИН как доказательство того, что они не приняли участия в

голосовании. В то же время обман не раскроется, если избиратели, отказавшиеся продолжать голосование, не будут этого делать.

Следовательно, если избиратель отказывается голосовать, то узнать об этом можно только от самого избирателя, и ИК должна быть не единственной, кто узнает об этом. Исключения этой угрозы можно достигнуть, используя сообщение избирателя об отказе от голосования некоторому количеству других избирателей (число которых определяется до начала голосования). Представляется весьма вероятным, что кто-нибудь из этих «доверенных» избирателей проверит, учла ли ИК «отказы» от голосования и если нет, то он пошлет протест (см. ниже *коррекцию протокола*), содержащий в качестве подтверждения сообщение M_a. (Заметим, что сообщение M_a является неопровержимым доказательством того, что определенный избиратель отказался голосовать.)

Совершенно необходимо, чтобы на протяжении всех шагов протокола ТГ (вплоть до шага 11) избиратель не открывал бы свой ИН. Предположим противное, т. е., что на шаге 8 в сообщении M_2 вместо числа P открыто передается ИН, т. е.:

$$M_2 = \left(I, E_{d_v}\left(I_s, V\right)\right).$$

Тогда у ИК, пока не закончится этот шаг, остается время на создание сообщения с таким же ИН. Она подписывает ИН, получая I_s и затем создает V', в котором она заинтересована. Далее, имитируя избирателя, ИК создает ключи e'_v и d'_v, шифрует сообщение и отправляет его *самой себе*. Результатом таких нечестных действий ИК окажется, что у некоторых избирателей совпадут ИН, а это может означать, что некоторые избиратели решили проголосовать несколько раз с одним и тем же ИН. Так как никто не знает истинного владельца ИН, то доказать, что именно ИК создала это сообщение оказывается невозможным. Избиратель мог попытаться дважды проголосовать, и одно из его сообщений необходимо будет исключить. Однако органом, который исключает такие сообщения, является именно ИК и она может оставить свое сообщение, т. е. фактически проголосовать вместо избирателя. Впоследствии избиратель не сможет доказать, что он не отправлял два сообщения и поэтому уличить ИК в нечестных действиях будет невозможно.

По окончанию протокола может произойти конфликтная ситуация, состоящая в совпадении некоторых ИН. Это будет возможно в трех случаях:

а) у избирателей случайно совпали ИН;

б) нечестный избиратель с помощью своего ИН решил проголосовать несколько раз;

в) нечестная ИК создала дополнительный ИН и решила проголосовать, но он случайно совпал с ИН избирателя.

Понятно, что изменять ИН в этих случаях не имеет смысла, так как если совпадение номеров произошло из-за случая «б», то при замене ИН,

один избиратель сможет проголосовать несколько раз, поэтому в данном протоколе нельзя требовать замены ИН.

Так как вероятность совпадений ИН очень мала, то принятый голос можно отвести к разряду «сомнительных». Если два (или более) сомнительных голоса сделали один и тот же выбор, то разумно учесть в результатах только один из них. Если же выборы отличаются, то следует не учитывать ни один из голосов. Все операции с «сомнительными» голосами должны быть опубликованы вместе с результатами голосования.

Если «сомнительные» голоса являются решающими, то наилучшим способом выхода из ситуации будет проведение повторного голосования.

Вполне возможно, что какой-либо злоумышленник, зная открытый ключ ИК, может попытаться подделать подпись ИК. Допустим, злоумышленник подбирает такое a, что шифрование этого числа открытым ключом ИК приводит к индентификатору I, которое можно представить как некоторый ИН:

$$(a)^{e_{\text{ик}}} \bmod n_{\text{ик}} = I \,. \qquad (5.16)$$

Далее злоумышленник формирует на шаге 8 бюллетень, где вместо I_s (подписанного ИН) записывает значение a, т. е. формирует сообщение:

$$M_2 = \left(P, E_{d_v}\left(I, a, V\right)\right).$$

Поскольку ИК не знает отправителя этого сообщения, то она считает, что оно пришло от легитимного избирателя. При проверке подписи ИК выполнит операцию (5.16) и, следовательно, подтвердит достоверность подписи. Таким образом, злоумышленник сможет проголосовать, даже не являясь легитимным избирателем. Решением данной проблемы является уменьшение вероятности выполнения соотношения (5.16), где I допустимое значение ИН, до такой степени, что ею можно будет пренебречь. Это вполне возможно при ограничении значений I и увеличении величины $n_{\text{ик}}$.

Коррекция выполнения основного протокола

При выполнении данного протокола могут возникнуть следующие *конфликтные ситуации:*

1) некоторые избиратели оказались не включёнными в список легитимных избирателей;

2) подписи I_s под ИН у некоторых избирателей оказались неверными.

3) ИК не учла некоторых из тех, кто отказался голосовать после шага 4.

4) некоторые результаты голосования случайно не учтены или учтены неверно.

Если один из таких конфликтов будет иметь место, то избиратель (совместно с честной ИК) приступит к коррекции протокола голосования.

Коррекция ситуаций в случаях 1 и 2 тривиальна. Действительно, для этого избирателям достаточно обратиться к ИК по открытым каналам и выслать ей свои претензии, причем в случае 2 это сводится к просьбе повторения шагов протокола 2–7.

Рассмотрим ситуацию 3, когда ИК при публикации не учла голосов тех, кто отказался голосовать. (ИК должна была опубликовать список тех, кто перестал голосовать после шага 4.). Так как «отказавшийся» избиратель отсылает некоторому количеству «доверенных» избирателей сообщения об отказе от продолжения голосования, то избиратели, получившие сообщения M_a, обращаются к ИК с этими сообщениями и требуют повторного расчета результатов голосования. (Напомним, что сообщение M_a было подписано секретным ключом избирателя, а это является достаточным признаком его достоверности).

При возникновении ситуации 4, только сам избиратель сможет узнать, правильно ли учтен его голос. Если это не так, то он выражает протест и повторно отправляет по анонимному каналу сообщения M_2 и M_3, которые были посланы ранее на шаге 8 и 10. Так как ИК публиковала все сообщения M_2 на общедоступном сайте, то она не сможет проигнорировать эти сообщения. В случае же если ИК на шаге 10 проигнорировала сообщение M_3, или же сам избиратель не послал на этом шаге M_3, то он во время выполнения дополнительного протокола *протеста* имеет право на учет его голоса.

Как видно, все видимые пока конфликтные ситуации разрешаются без спорных вопросов. В целом, протокол тайного голосования является достаточно сложной процедурой, требующей для своего выполнения наличия качественных каналов связи между избирателями и ИК, времени и вычислительных ресурсов. При наличии большого количества избирателей (например сотен тысяч или миллионов) практическая реализация подобных протоколов требует дальнейшей проработки. Что же касается процедуры тайного голосования по телекоммуникационным каналам связи при ограниченном контингенте избирателей (скажем в рамках решения корпоративных проблем), то рассмотренный выше алгоритм электронного голосования по открытым каналам связи оказывается вполне приемлемым, правда, при надежном решении проблемы создания анонимного канала.

5.2.10. Краткие выводы по криптографическим протоколам

В подразд. 5.2.1–5.2.9 были рассмотрены алгоритмы выполнения девяти КП из достаточно большого списка, содержащего 14 таких КП, который в действительности мог бы быть еще и продолжен. Очевидно, что столь же подробное изучение всех оставшихся КП заняло бы слишком много места и изложение такого материала вышло бы за рамки книги «Основы криптографии». Описание способов выполнения большинства оставшихся протоколов можно найти в монографиях [14,15], а также в трудах основных международных конференций по криптографии [18]. Важно отметить, что для обоснования «не учебной» секретности КП существует достаточно развитая теория, использующая понятие *семантической стойкости* и другие математические понятия [19], только применение которых и может дать гарантии защищенности КП от всех мыслимых атак.

СПИСОК ЛИТЕРАТУРЫ К ЧАСТИ II

Основная

1. Виноградов, И. М. Основы теории чисел / И. М. Виноградов. – М. : Наука, 1965.

2. Koblitz, N. A. Course in Number Theory Cryptography / N. A. Koblitz. – New-York : Springer, 1987.

3. Menezes, A. J. Handbook of Applied Cryptography / A. J. Menezes [et al.]. – New-York, London, Tokyo : CRC Press, 1996.

4. Питерсон, У. Коды, исправляющие ошибки / У. Питерсон, Э. Уэлдон. – М. : Мир, 1976.

5. Henk C. A. van Tilborg. Fundamentals of Cryptography / Henk C. A. van Tilborg. – Boston, Dor-tech, London : Kluwer, 2001.

6. Agrawal, M. Primes in P / M. Agrawal, N. Kayal, N. Saxena. – Online News. – www.cse.iitk.ac.in/users/manindra/primality.ps, 2002.

7. Коржик, В. И. Основы защиты информации в компьютерных системах : методические указания к лабораторным работам. Часть 2 / В. И. Коржик, Д. В. Кушнир ; СПбГУТ. – СПб., 1999.

8. Болотов, А. А. Элементарное введение в эллиптическую криптографию / А. А. Болотов [и др.]. – М. : КомКнига, 2006.

9. Berlekamp, E. R. Ontheinternetintractabilityofcertaincodingproblems / E. R. Berlekamp [etal.] / IEEETrans. onIT. – 1978. – Vol. 24. – P. 384–386.

10. Chaum, D. Unconditionally – Secure Digital Signatures / D. Chaum, S. Roijakkers. – Crypto, 1998. – P. 206–214.

11. Коржик, В. И. Теоретические основы информационной безопасности телекоммуникационных систем : учебное пособие / В. И. Коржик, Д. В. Кушнир ; СПбГУТ. – СПб., 2000.

12. Феллер, В. Введение в теорию вероятностей и ее приложения : т. 1. /В. Феллер. – М. : Мир, 1976.

13 Молдавян, Н. А. Введение в криптосистемы с открытым ключем / Н. А. Молдавян, А. А. Молдавян. – СПб. : БХВ-Петербург, 2005.

14. Шнайер, Б. Прикладная криптография / Б.Шнайер. – М. : Триумф, 2002.

15. Salomaa, A. Public – key Cryptography / A. Salomaa. – Berlin : Springer, 1990.

16. Schneier, B. E-Mail Security / B. Schneier. – New-York : Wiley, 1995.

17. Sherwood, R. A protocol for Scalable Anonymous Communication / R. Sherwood [et al.] // Trans. Eurocrypt. – 2001. – P. 1–13.

18. International Conferences on Cryptography within the period 1979–1999. CD.

19. Мао, В. Современная криптография / В. Мао. – М., СПб., Киев : Вильямс, 2005.

20. Bennet, C. Generalized privacy amplification / C. Bennet [et al.] // IEEE Trans. on IT. – 1995. – Vol. 41, n 6. – P. 1915–1923.

21. Yakovlev, V. Achievability of key-capacity in a scenario of key sharing by public discussion and in the presence of passive eavesdropper / V. Yakovlev, V. Korjik, A. Sinuk // Lecture Notes in Computer Science. – 2003. – Vol. 2776. – P. 308–315.

22. Bennet, C. The dawn of a new era for quantum cryptography: the experimental prototype is working! / C. Bennet. G. Brassard // Sigact. – News. – 1989. – P. 78–82.

Дополнительная

23. Kleinjung, T. Factorization of a 768-bit RSA modulus /T. Kleinjung [et al.]. – http://eprint.iacr.org/2010/006.pdf.

24. The Elliptic Curve Digital Signature Algorithm. ISOstandart.

25. ГОСТ Р 34.10-2012. Национальный стандарт Российской Федерации. Информационная технология. Процессы формирования и проверки электронной цифровой подписи.

26. Canetti, R. Deniable Encription /R.Canetti [et al.] // CRYPTO. – 1997. – Proceedings. – P. 90–104.

27. Смарт, Н. Критография / Н. Смарт. – М. : Техносфера, 2005.

28. ГОСТ К 34.11–2012. Национальный стандарт Российской Федерации. Информационная технология. Криптографическая защита информации. Функция хеширования.

29. Федеральный закон от 06 апреля 2011 г. № 63-ФЗ. Об электронной подписи // Российская газета. – 2011. – 08 апр.

30. Atkinson,K. An introduction to numerical analysis : Second Edition / K. Atkinson. – New-York, Wiley, 1989.

31. PKCS#1 : RSA Cryptography Sandard v22.

32. Riesel, H.Prime Numbers and Computer Methods for Factorization : Second Edition / H.Riesel. – Reese, Springer Science, 2011.

ЧАСТЬ III

УПРАВЛЕНИЕКЛЮЧАМИ В КРИПТОГРАФИЧЕСКИХ СИСТЕМАХ

В ч.I были рассмотрены симметричные (одноключевые) криптосистемы, при этом предполагалось, что секретные ключи, необходимые для выполнения криптографических преобразований шифрования, аутентификации заранее распределены между корреспондентами (пользователями) сети. Отмечалась сложность задачи обеспечения корреспондентов секретными ключами. В ч.II рассмотрены асимметричные (двухключевые) криптосистемы преимущество которых перед симметричными заключается в том, что обеспечение корреспондентов осуществляется не секретными, а открытыми ключами.

В данной части книги мы рассмотрим детально вопросы, связанные с обеспечением корреспондентов необходимыми ключами и их использованием для выполнения криптографических преобразований, начиная от вопросов генерирования ключей, их распределения между корреспондентами и применения в криптоалгоритмах. Эти вопросы принято называть управлением ключами.

В самом широком смысле под *управление ключами* понимается совокупность технологий и процедур, обеспечивающих установление и поддержание отношения *криптографической связности* участников криптографического протокола. Под криптографической связностью будем понимать наличие у участников протокола ключевых данных (ключевого материала) достаточного для выработки сеансового ключа.

Содержание управления существенно зависит от вида криптографической системы, то есть от того какие ключи используются: симметричные или асимметричные. В разд.1 рассматриваются вопросы управления ключами в симметричных криптографических системах. В разд. 2 рассматриваются вопросы управления открытыми ключами асимметричных криптографических систем. В разд. 3 показано, как осуществляется управление ключами в реальных сетевых протоколах.

1. УПРАВЛЕНИЕ КЛЮЧАМИ В СИММЕТРИЧНЫХ КРИПТОГРАФИЧЕСКИХ СИСТЕМАХ

1.1. Модель управления ключами

Будем полагать, что управление ключами осуществляется в некоторой инфокоммуникационной сети, состоящей из M узлов (корреспондентов), соединенных каналами связи. В управлении ключами участвуют (рис.1.1):

-центр управления связью;

- центр распределения ключей;
- узлы связи.

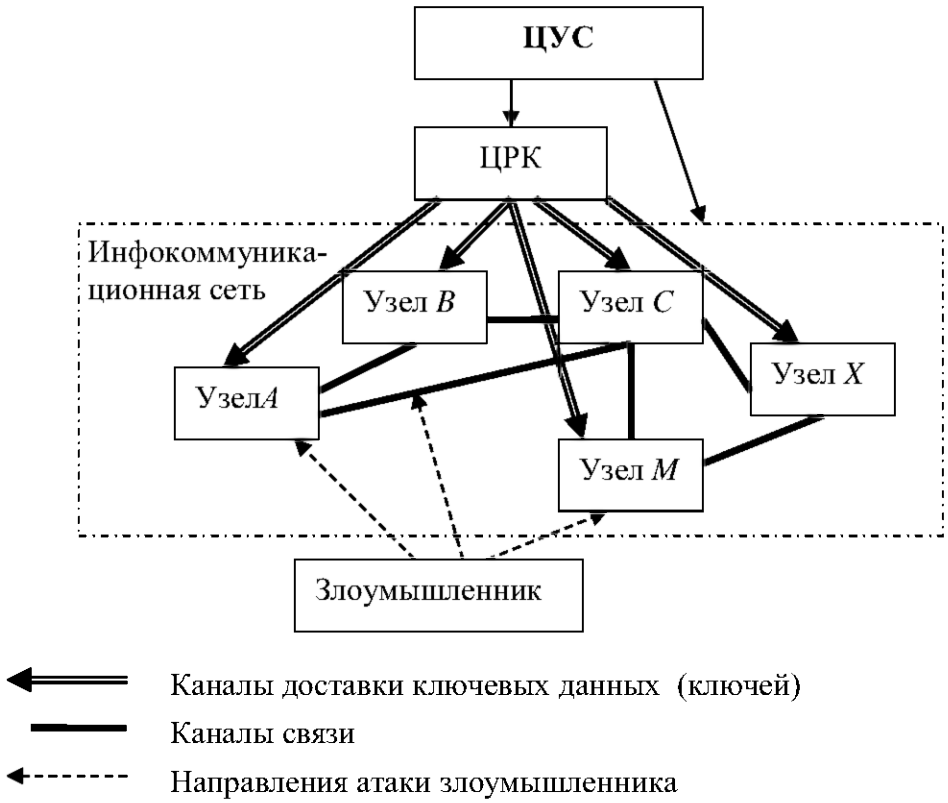


Рис.1.1. Модель управления ключами
в симметричной криптографической системе

Центр управления связью (ЦУС) создает сеть необходимой структуры, вырабатывает политику безопасности сети, в том числе политику управления ключами, включающую требования, предъявляемые к ключу, порядок его использования, способ распределения. При выработке политики учитывается модель злоумышленника, воздействующего на элементы сети.

Центр распределения ключей (ЦРК) формирует наборы ключевых данных (ключевого материала), представляющих собой структурированные случайные последовательности чисел, на основе которых обеспечивается криптографическая связность корреспондентов сети в соответствии с определенной ЦУС структурой сети. Также ЦРК осуществляет доставку ключей (ключевых данных) на узлы связи, используя доверенные (защищенные) каналы связи.

Узлы связи осуществляют: выработку сеансовых ключей из доставленных ЦРК ключевых данных, применение ключей в криптографических протоколах, уничтожение ключей после окончания срока их использования.

Управление ключами осуществляется в предположении действия в сети злоумышленника, который:

-может перехватывать любое сообщение, передаваемое по каналам сети;

-может получать и посылать сообщения любому корреспонденту сети, маскируясь под другого корреспондента.

- может выполнять действия, приводящие к компрометации ключей (ключевых данных) в отдельных узлах сети.

В некоторых случаях мы будем предполагать, что злоумышленник является законным корреспондентом сети и может инициировать контакт с другими корреспондентами. Также будем полагать, что злоумышленник не может воздействовать на ЦУС, ЦРК и доверенные каналы доставки ключей, так как они представляют собой защищенные от постороннего воздействия объекты.

1.2. Этапы жизненного цикла ключа

Управление ключами должно проводиться на всех этапах жизненного цикла ключа – от создания ключа до его уничтожения. Структура жизненного цикла ключей симметричных криптографических систем показана на рис. 1.2.

Рассмотрим основные этапы жизненного цикла ключа.

1. ЦУС на основе структуры сети, разрабатывает политику формирования и использования ключей, т. е. определяет способы обеспечения криптографической связности корреспондентов сети с учетом выполнения требований, предъявляемых к ключу и процессу его распределения.

К ключу симметричной криптосистемы можно предъявить следующие требования:

-заданная длина ключа – n бит;

- случайность и равновероятность символов ключа;

- секретность ключа на всех этапах жизненного цикла;

- целостность ключа, предполагающая, что ни один бит ключа не может быть искажен, или подменен;

-новизна (свежесть) ключа, предполагающая, что ключи должны периодически меняться. Частота изменения ключа определяется политикой использования ключей.

Далее в ЦРК выполняются 2–4 этапы.

2. Генерирование случайных последовательностей (чисел).

3. Формирование ключевых данных, построенных определенным образом в целях обеспечения связности корреспондентов и устойчивости сети при компрометации ключей.

Рис. 1.2. Структура жизненного цикла ключей симметричных криптосистем

4. Доставку ключей корреспондентам.

После получения ключевых данных от ЦРК корреспонденты сети выполняют следующие шаги:

5. Формирование с помощью данных, полученных от ЦРК, парных в т. ч. сеансовых ключей для работы друг с другом.

6. Штатное применение сеансовых ключей в соответствующем криптоалгоритме для шифрования или аутентификации.

7. Уничтожение сеансовых ключей, ключевых данных, носителей ключей после определенного времени их использования

Также следует отметить, что на некоторых этапах, например, после доставки ключевых данных корреспондентам, требуется осуществить учет ключей и их носителей и обеспечить их безопасное хранение.

Рассмотрим подробнее важные этапы жизненного цикла ключа.

1.3. Генерирование случайных последовательностей (чисел)

Формирование ключевых данных в первую очередь предполагает генерирование случайных чисел. Можно выделить три способа формирования случайных чисел:

-биометрический;

-программный;

-аппаратный.

Биометрический способ

При этом способе в качестве источника случайности могут использоваться различные биометрические характеристики человека.

Например, в программе PGP (PrettyGoodPrivacy) в качестве случайного параметра используются интервалы времени между случайными нажатиями пользователем клавиш на клавиатуре и значения нажатых клавиш. Эти параметры оцифровываются и записываются в 256-битный буфер. При необходимости формирования ключа случайные числа извлекаются из буфера.

Другим вариантом получения случайных чисел является отслеживание координат «мыши» при ее случайном перемещении пользователем. Основой недостаток биометрического способа заключается в невысокой производительности генерирования случайных чисел и их недостаточной «случайности».

Программный способ

При этом способе, в отличие от предыдущего, могут формироваться достаточно длинные последовательности двоичных или q-ичных псевдослучайных чисел путем преобразования по определенному алгоритму некоторых начальных данных, которые могут быть либо детерминированными, либо случайными числами. Известно достаточно много таких алгоритмов [8]. Рассмотрим некоторые из них.

Линейный конгруэнтный генератор. Псевдослучайная последовательность (ПСП) формируется согласно уравнению

$$x_{n+1} = (ax_n + b) \bmod m,$$

где целые числа x_n, x_{n+1} –n-й и n+1-й элементы ПСП; $m > 0$ – модуль; $0 < a, b < m$, x_0 – начальное заполнение. Максимальный период ПСП равен m при выборе параметров генератора согласно следующей теореме.

Теорема1.1 [8]. Линейная конгруэнтная последовательность, определяемая числами a,b,m имеет период равный m тогда и только тогда, когда

- m и b взаимно простые числа;

- $a - 1$ кратно p для каждого простого p, являющегося делителем m;

- $a-1$ кратно 4, если m кратно 4.

Аддитивный генератор Фибоначчи. Псевдослучайная последовательность (ПСП) формируется согласно уравнению

$$x_{n+1} = (x_{n-p} + x_{n-q}) \bmod m .$$

Теорема 1.2 [8]. Если многочлен $x^q + x^p + 1$ является примитивным над полем $GF(2)$, то последовательность, формируемая аддитивным генератором Фибоначчи, имеет максимальный период равный $2^{\log_2 m-1}(2^q - 1)$.

Генератор ПСП на основе линейного рекуррентного регистра. Линейные рекуррентные регистры были рассмотрены в п.3.2 ч. I данного пособия. Задавая случайное начальное состояние или случайные обратные связи ЛРР длины m, соответствующие примитивному полиному обратных связей (п. 3.2), можно сформировать ПСП с периодом $2^m - 1$. Обобщением ЛРР являются регистры, генерирующие q-ичные последовательности. Данный вид генераторов ПСП может быть усилен дополнительным применением нелинейных преобразований.

Генераторы ПСП на основе криптографических преобразований

Структурная схема такого генератора включает счетчик и блочный шифратор (рис. 1.3). Начальное состояние счетчика и ключ для блочного шифра задаются число случайными числами.

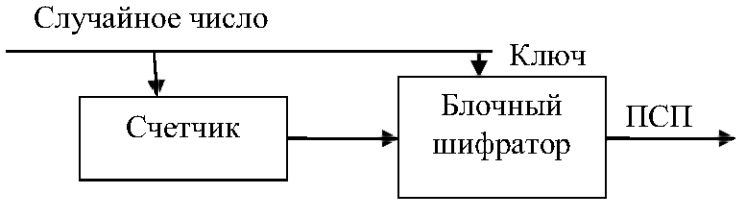


Рис.1.3. Схема генератора ПСП на основе блочного шифра

В дальнейшем с каждым тактом счетчик увеличивает свое состояние на 1. Числа с выхода счетчика шифруются блочным шифром. Выходные блоки шифратора образуют псевдослучайную последовательность чисел.

Аппаратный генератор на основе физического генератора шума

Структурная схема генератора показана на рис. 1.4.

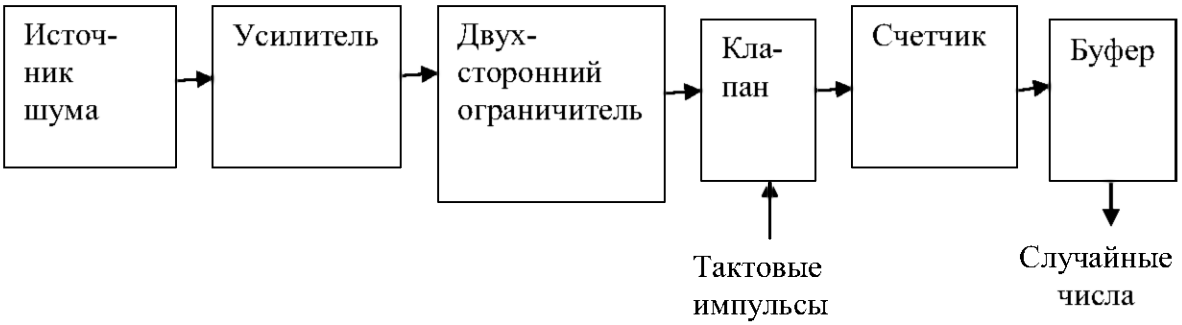


Рис. 1.4. Структурная схема аппаратного ГСЧ

В аппаратных ГСЧ источником случайности является какой-либо физический процесс. Наиболее часто для этих целей используют флуктуационные шумы, возникающие в полупроводниковых приборах. Источником флуктуационных шумов может быть шумовой диод, представляющий собой стабилитрон, рабочий режим которого выбран в неустойчивой области p–n перехода. Под действием многих факторов (изменения температуры, давления, влажности, питающего напряжения) на выходе диода появляются слабые флуктуационные колебания напряжения (флуктуационный шум). Эти колебания усиливаются и подаются на двухсторонний ограничитель амплитуды. На выходе ограничителя имеем двухполярную последовательность импульсов постоянной амплитуды и переменной (случайной) длительности. Последовательность этих импульсов управляет электронным клапаном. На второй вход электронного клапана поступает регулярная последовательность тактовых импульсов. В течение положительной полуволны сигнала на выходе ограничителя клапан открывается, и последовательность тактовых импульсов поступает в счетчик, который подсчитывает их количество. Во время отрицательной полуволны клапан закрыт, содержимое счетчика (случайнее число) переписывается в буфер.

Важнейшими параметрами любых ГСЧ являются их статистические характеристики, показывающие насколько генерируемые числа близки к чисто случайным. Подробное описание методов оценки этих параметров можно найти в [8,9]. Для улучшения статистических характеристик последовательностей, которые используются в качестве ключа, целесообразно применять экстракторы [21].

1.4. Распределение ключей для симметричных криптосистем с использованием центра распределения ключей и доверенных каналов доставки ключа на начальном этапе

Этапы формирования ключевых данных, их доставки корреспондентам и формирование на их основе сеансовых ключей корреспондентами являются зависимыми между собой и их целесообразно рассматривать как один укрупненный элемент жизненного цикла ключа, который назовем *распределением ключей.*

Распределение ключей наиболее сложный этап жизненного цикла ключа. Способы распределения ключей могут быть разными и определяться многими факторами, в первую очередь, ролью ЦРК в этом процессе. Можно выделить такие способы распределения ключей:

1) способ с использованием ЦРК и доверенных каналах доставки ключа на начальном этапе;

2) способ с использованием ЦРК в интерактивном режиме;

3) способы, на основе взаимного обмена данными между корреспондентами (без использования ЦРК).

1.4.1. Простейшие ключевые структуры

Рассмотрим некоторую сеть связи, содержащую M узлов (корреспондентов), в которой корреспонденты должны иметь возможность выполнения друг с другом криптографического протокола (шифрования, аутентификации).

В общем виде процедура распределения ключей реализуется следующим образом. Сначала в ЦРК генерируется ключевой материал (ключевые данные), которые по доверенным каналам связи, например, специальной связью доставляются на узлы. Корреспонденты на основе этих ключевых данных формируют сеансовые ключи и на этих ключах выполняют криптографический протокол.

Введем определения.

Ключевое отношение – отношение между двумя корреспондентами сети связи, при котором взаимодействующие стороны имеют ключевые данные, достаточные для выполнения криптографического протокола. В этом случае корреспонденты оказываются *криптографически связанными (криптосвязанными).*

Ключевая структура – совокупность ключевых отношений, определяющих структуру ключевого материала (ключевых данных), распределяемого в сети и обеспечивающего криптографическую связность корреспондентов сети.

Компрометация ключей – происшествие (захват, хищение, разглашение и др.), при котором ключ или ключевые данные становятся известными злоумышленнику.

Криптоживучесть – устойчивость ключевой структуры при компрометации ключей (ключевых данных) у отдельных корреспондентов.

Очевидно, что при данном способе распределения каналы доставки ключей имеют не высокую пропускную способность, поэтому формируемая в ЦРК ключевая структура должна удовлетворять следующим требованиям:

- малый объем ключевых данных (ключевого материала), доставляемых корреспонденту;
- малый объем ключевых данных, хранимых у корреспондента;
- простота формирования сеансового ключа из ключевых данных;
- устойчивость ключевой структуры при компрометациях ключей в отдельных узлах сети.

Простейшей ключевой структурой является структура, в которой все корреспонденты сети имеют один и тот же ключ длиной n_0. Назовем эту структуру *единый ключ*. Очевидно, что в этом случае объем генерируемого в ЦРК и распределяемого в сети по доверенным каналам ключевого материала совпадает с длиной ключа и равен n_0. Также очевидно, что данная ключевая структура не обеспечивает устойчивости к компрометации ключа, поскольку компрометация ключа на одном узле компрометирует ключи во всей сети.

Другим видом ключевой структуры (КС) является структура, в которой любая пара корреспондентов сети имеет свой парный ключ. Для построения такой КС ЦРК генерирует таблицу из M строк и M столбцов. Элемент K_{ij} таблицы есть ключ длиной n_0 бит для связи i-го и j-го корреспондентов. Естественно, что в таблице $K_{ij}=K_{ji}$. Первая строка таблицы содержит ключи, необходимые первому корреспонденту для взаимодействия с остальными корреспондентами сети. Вторая строка содержит все ключи, необходимые второму корреспонденту для связи с остальными. Остальные строки таблицы содержат ключи для других корреспондентов.

Набор ключей каждой строки образует ключевой материал, который доставляется каждому корреспонденту по доверенным каналам. Назовем данную ключевую структуру *сетевым набором.*

Несложно заметить, что объем ключевых данных, доставляемых каждому корреспонденту $(M-1)n_0$ бит, а объем ключевых данных генерируемых в ЦРК и доставляемых всем корреспондентам сети $M(M-1)n_0$ бит. При больших значениях M объем ключевых данных генерируемых в ЦРК и доставляемых всем корреспондентам сети приблизительно $M^2 n_0$ бит. То есть объем ключевых данных, распределяемых в сети по доверенным ка-

налам для ключевой структуры сетевой набор в M раз больше, чем для ключевой структуры Единый ключ.

С другой стороны такая ключевая структура имеет высокую устойчивость к компрометациям ключей. Действительно, компрометация ключевых данных на одном узле влияет только на связи этого узла с остальными и никак не влияет на криптосвязность остальных узлов между собой. Аналогичная ситуация будет иметь мест и при компрометации второго, третьего и т.д. узлов.

Рассмотрим далее принцип построения и характеристики ключевой структуры, которую назовем *базовым набором* [10, 11]. В этой КС объем ключевых данных, распределяемых в сети значительно меньше, чем в КС сетевой набор, с другой стороны эта КС может обеспечить устойчивость к заданному количеству компрометаций ключей в узлах сети.

1.4.2. Принцип построения и характеристики ключевой структуры Базовый набор

Распределение ключей осуществляется в три этапа.

Первый этап выполняется в ЦРК.

1. ЦРК генерирует открытые адреса (коды) пользователей:

$$\vec{P}_i = (P_{i1}, P_{i2}, \dots, P_{iL}),\ 1 \le i \le M, \tag{1.1}$$

где $P_{ij} \in GF\left(2^{n_0}\right)$, $1 \le i \le M$, $1 \le j \le L$; n_0 – длина ключа, необходимая для использования в определенном криптоалгоритме (в битах); L –целое число, как правило $L << M$, определяющее стойкость системы к компрометациям, величина которого будет определена в дальнейшем.

Особенностью адресов $\vec{P}_i$ является то обстоятельство, что их можно открыто передавать по каналам связи и хранить в несекретных справочниках без ущерба для безопасности связи.

2. ЦРК генерирует случайную секретную квадратную $L \times L$ матрицу $T = \{t_{ms}\}$, где $t_{ms} \in GF\left(2^{n_0}\right), 1 \le m, s \le L$.

По случайной матрице T, ЦРК формирует *базовые наборы* ключей корреспондентов:

$$\vec{S}_i = T\,\vec{P}_i',\ 1 \le i \le M, \tag{1.2}$$

где символ (′)– знак транспонирования вектор-строки – адреса пользователя.

На втором этапе происходит следующее:

1) ЦРК рассылает отрытым образом пользователям или публикует в виде файла-справочника их адреса $\vec{P}_i$, $1 \le i \le M$ (при этом должна быть исключена возможность подмены адресов злоумышленником);

2) ЦРК рассылает по доверенным каналам (без возможности перехвата злоумышленником) базовые наборы ключей $\vec{S}_i$, $1 \le i \le M$ всем корреспондентам (каждому только один соответствующий ключ);

3) ЦРК может уничтожить исходную случайную матрицу T, чтобы исключить последующую угрозу ее компрометации или хранить защищенным образом для того, чтобы иметь возможность расширять систему снабжения ключами при появлении новых пользователей.

Третий этап происходит между двумя корреспондентами без участия ЦРК. Рассмотрим порядок формирования парного (сеансового) ключа между i-ым и j-ым корреспондентами.

i-й корреспондент:

- находит (или извлекает из памяти) открытый адрес j-го корреспондента $\vec{P}_j$;

- формирует парный ключ для работы с j-м корреспондентом, используя свой базовый набор $\vec{S}_i$.

$$i: K_{ij} = \vec{P}_j \vec{S}_i. \tag{1.3}$$

j-й корреспондент:

- находит (или извлекает из памяти) открытый адрес i-го корреспондента $\vec{P}_i$;

- формирует парный ключ с i-м корреспондентом, используя свой базовый набор $\vec{S}_j$.

$$i: K_{ji} = \vec{P}_i \vec{S}_j. \tag{1.4}$$

Дадим характеристики способа распределения ключей на основе базовых наборов.

Теорема 1.3. Для того чтобы выполнялось равенство парных ключей от i-го пользователя j-му и наоборот, т. е. $K_{ij} = K_{ji}$ для K_{ij} и K_{ji}, определяемых уравнениями (1.3) и (1.4), при любых возможных наборах адресов $\vec{P}_i$ и $\vec{P}_j$, определяемых по (1.1), достаточно, чтобы матрица T была бы симметричной, т.е. выполнялось условие:

$$t_{sk} = t_{ks},\ 1 \le s,\ k \le L. \tag{1.5}$$

Доказательство. Из соотношений (1.2) и (1.3) получаем:

$$K_{ij} = \sum_{s=1}^{L} P_{js} \sum_{k=1}^{L} t_{sk} P_{ik} = \sum_{s=1}^{L} \sum_{k=1}^{L} P_{js} t_{sk} P_{ik}$$

$$K_{ji} = \sum_{s=1}^{L} P_{is} \sum_{k=1}^{L} t_{sk} P_{jk} = \sum_{s=1}^{L} \sum_{k=1}^{L} P_{is} t_{sk} P_{jk} \quad . \tag{1.6}$$

Если во втором уравнении сделать замену индексов $s \leftrightarrow k$ и учесть, что матрица T симметричная, т.е. $t_{sk}=t_{ks}$, то тогда $K_{ij} = K_{ji}$.

Таким образом, ЦРК может вырабатывать матрицу T как полностью случайную матрицу выше главной диагонали, и тогда из представления для парного ключа:

$$K_{ij} = \sum_{s=1}^{L} P_{js} \sum_{k=1}^{L} t_{sk} P_{ik} = \sum_{s=1}^{L} \sum_{k=1}^{L} P_{js} t_{sk} P_{ik} = \sum_{r=1}^{L} t_{rr} P_{jr} P_{ir} + \sum_{s<k}^{L} t_{sk} (P_{js} P_{ik} + P_{jk} P_{is}). \quad (1.7)$$

Видно, что все значения ключей $K_{ij} \in GF\left(2^{n_0}\right)$ окажутся равновероятными.

Рассмотрим стойкость данной ключевой структуры к компрометациям ключей у корреспондентов. Возможна компрометация, как базовых наборов, так и компрометация парных ключей. Оба вида компрометации являются связанными.

Теорема 1.4. Компрометация базового набора $\vec{S}_i$ эквивалента компрометации M–1 парных ключей K_{ij}, $1 \le i,j \le M$ корреспондента iи наоборот.

Доказательство. Пусть произошла компрометация базового набора $\vec{S}_i$, тогда, используя соотношение (1.3), мы можем вычислить все M – 1 парных ключей K_{ij}, $1 \le i,1 \le M$. Наоборот, если нам известно M – 1 парных ключей K_{ij}, $1 \le i$, j $\le M$ (фактически достаточно знать L таких ключей), решая системы уравнений (1.3) и (1.4), мы можем найти $\vec{S}_i$.

Очевидно, что компрометация базового набора более опасна, поэтому исследуем далее устойчивость данной ключевой структуры относительно компрометации базовых наборов у корреспондентов сети.

Теорема 1.5. Ключевая структура Базовый набор устойчива к компрометации L – 1 базовых наборов тогда и только тогда, когда множество адресов $\{\vec{P}_i\}$ обладает тем свойством, что любая система из L векторов этого множества линейно независима.

Доказательство. Открытые адреса корреспондентов представляют собой L-мерные векторы над полем $GF(2^n)$. Из свойства линейной независимости над полем $GF\left(2^{n_0}\right)$, следует, что для любого набора из L векторов выполняется условие

$$\lambda_1 \vec{P}_{i_1} + \lambda_2 \vec{P}_{i_2} + ... \lambda_L \vec{P}_{i_L} = 0 \quad (1.8)$$

тогда и только тогда, когда $\lambda_1=\lambda_2=...=\lambda_L=0$.

Покажем, что для полной компрометации всей системы достаточно скомпрометировать L базовых наборов любых пользователей системы.

Пусть скомпрометировано L базовых наборов $(S_{i1}, S_{i2},..., S_{iL})=I_{cp}$. Рассмотрим какой-либо нескомпрометированный базовый набор $\vec{S}_{il} \notin I_{cp}$. Тогда

$$\vec{S}_{il} = T\vec{P}'_{il} \ . \quad (1.9)$$

Выразим $\vec{P}_{il}$ через открытые адреса скомпрометированных пользователей

$$\vec{P}_{il} = \sum_{r=1}^{L} \lambda_r \vec{P}_{i_r} . \tag{1.10}$$

Подставляя (1.10) в (1.9), находим

$$\vec{S}_{il} = T \sum_{r=1}^{L} \lambda_r \vec{P}_{i_r} = \sum_{r=1}^{L} \lambda_r \vec{S}_{i_r} . \tag{1.11}$$

Из выражения (1.11) видно, что базовый набор однозначно находится достаточно простым образом как линейная комбинация скомпрометированных базовых наборов, в которой коэффициенты находятся из уравнения (1.11).

Докажем теперь обратную теорему. Если скомпрометировано L – 1 и менее базовых наборов каких-либо корреспондентов, то каждый из парных ключей, соответствующий нескомпрометированному корреспонденту, является равновероятным.

Пусть скомпрометированы L – 1 базовых наборов корреспондентов с номерами i_1, i_2,…, i_{L-1}. Тогда можно по ним вычислить $(L – 1)(M – 1)$ парных ключей, используя соотношение (1.6). Однако среди этого множества ключей, которое мы обозначим через $I_{i_1, i_2, \ldots, i_{L-1}}$, будут содержаться как совпадающие, так и линейно зависимые ключи, причем последние определяются как такие ключи K_{sk}, для которых справедливо равенство:

$$K_{sk} = \sum_{(i_r, j_r) \in I_{i_1, i_2, \ldots, i_{L-1}}} \lambda_r K_{i_r j_r} , \tag{1.12}$$

где λ_r – некоторые постоянные из $GF(2^n)$.

Рассмотрим множество $\hat{I}_{i_1, i_2, \ldots, i_{L-1}}$, состоящее из $(L+2)(L – 1)/2$ парных ключей, отобранных из множества $I_{i_1, i_2, \ldots, i_{L-1}}$ следующим образом. Сначала выбирается L произвольных парных ключей корреспондента i_1, затем любые L – 1 парных ключей корреспондента с номером i_2 и т.д., любые два ключа корреспондента с номером i_{L-1}.

Легко проверить, что каждый парный ключ из множества $I_{i_1, i_2, \ldots, i_{L-1}}$ может быть представлен как линейная комбинация ключей из множества $\hat{I}_{i_1, i_2, \ldots, i_{L-1}}$. Составим систему из $(L+2)(L – 1)/2$ уравнений следующего вида

$$K_{i_r j_r} = \sum_{m=1}^{L} t_{mm} \alpha^{i_r + j_r} + \sum_{s<k} \sum t_{sk} \left(\alpha^{s i_r + k j_r} + \alpha^{k i_r + s j_r} \right). \tag{1.13}$$

Очевидно, что в системе уравнений (1.13) учтены все сведения о скомпрометированных личных ключах корреспондентов i_1, i_2,…, i_{L-1}.

Любой не скомпрометированный парный ключ $K_{ij} \notin I_{i_1,i_2,\ldots,i_{L-1}}$ может быть представлен как:

$$K_{ij} = \sum_{m=1}^{L} t_{mm}\alpha^{i+j} + \sum_{s<k}\sum t_{sk}\left(\alpha^{si+kj} + \alpha^{ki+sj}\right). \qquad (1.14)$$

Выразим какой-либо из коэффициентов t_{sk} матрицы T через свободный член первого уравнения системы (1.13) и подставим его во все остальные уравнения системы (1.13) и в (1.14). Затем какой-либо другой коэффициент матрицы T выразим через свободный член второго уравнения и подставим его во все оставшиеся уравнения (1.13) и в (1.14). Продолжая эту процедуру вплоть до использования последнего уравнения системы (1.13) и учитывая то обстоятельство, что число неизвестных в симметричной матрице T равно $L(L+1)/2$, т.е. на единицу меньше чем число уравнений (1.13), получим представление:

$$K_{ij} = At_{s_0k_0} + B, \qquad (1.15)$$

где $t_{s_0k_0}$ – последний оставшийся не выраженный через другие коэффициент матрицы $T; A, B \in GF\left(2^n\right)$ и не зависят от элементов множества $\hat{I}_{i_1,i_2,\ldots,i_{L-1}}$.

Из представления (1.15) видно, что при любых $L-1$ скомпрометированных базовых наборах ключей $I_{i_1,i_2,\ldots,i_{L-1}}$, парный ключ K_{ij} оказывается равновероятным, так как равновероятен элемент $t_{s_0k_0} \in GF\left(2^n\right)$. Если число скомпрометированных ключей оказывается еще меньше, чем $L-1$, то ясно, что равновероятность любого нескомпрометированного парного ключа тем более выполняется. Что и требовалось доказать.

Таким образом, при построении данной ключевой структуры особое внимание стоит уделять выбору ее параметров. Наиболее важным является выбор открытых адресов, которые, как было показано выше, должны быть линейно независимыми в любой совокупности из L векторов.

Это условие можно выполнить, если формировать открытые адреса следующим образом:

$$\vec{P}_i = \left(\alpha^i, \alpha^{2i}, \ldots, \alpha^{Li}\right), \quad i \le 2^{n_0} - 1, \qquad (1.16)$$

где α – примитивный элемент поля $GF(2^n)$.

Составим матрицу из L адресов, выбранных таким образом

$$A = \begin{pmatrix} \alpha^{i_1} & \alpha^{i_2} & \alpha^{i_L} \\ \alpha^{2i_1} & \alpha^{2i_2} & \alpha^{2i_L} \\ \alpha^{Li_1} & \alpha^{Li_2} & \alpha^{Li_L} \end{pmatrix}$$

и покажем, что определитель этой матрицы не равен нулю. Это будет означать, что L адресов-векторов линейно независимы. Представим матрицу A в виде

$$A=\begin{bmatrix} 1 & 1 & \dots & 1 \\ \alpha^{i_1} & \alpha^{i_2} & \dots & \alpha^{i_L} \\ \alpha^{2i_1} & \alpha^{2i_2} & \dots & \alpha^{2i_L} \\ \dots & \dots & \dots & \dots \\ \alpha^{(L-1)i_1} & \alpha^{(L-1)i_2} & \dots & \alpha^{(L-1)i_L} \end{bmatrix} \begin{bmatrix} \alpha^{i_1} & & \dots & \\ & \alpha^{i_2} & \dots & \\ \dots & \dots & \dots & \dots \\ & & \dots & \alpha^{i_L} \end{bmatrix}.$$

Определитель этой матрицы равен

$$|A|=|V|\alpha^{i_1+i_2+\dots+i_L}, \qquad (1.17)$$

где $|V|$ – определитель Вандермонда.

Элемент α в определителе Вандермонда примитивен, поэтому сам определитель не равен нулю, т. е. $|V|\neq 0$ (в силу примитивности различные степени элемента α^i при $i<2^{n_0}-1$ не совпадают друг с другом).

При любых $i_1,i_2,\dots,i_L$, множитель в (1.17) также неравен нулю, следовательно, определитель матрицы A отличен от нуля.

Поэтому выбор открытого адреса $\vec{P}_i$ по (1.16) обеспечивает линейную независимость любых наборов из L или меньшего числа открытых адресов. Ясно, что любой из L+1 или большего числа адресов является линейно зависимым, так как пространство всех возможных адресов является L-мерным.

Проведем оценку объема ключевого материала, распределяемого в сети для данной ключевой структуры при требовании устойчивости сети к $L-1$ компрометациях базовых наборов.

Как следует из соотношений (1.3), (1.4) объем базового набора составляет n_0L бит. Объем распределяемого KM для всех корреспондентов сети будет составлять n_0LM бит.

В табл. 1.1 представлены характеристики для трех рассмотренных выше ключевых структур.

Таблица 1.1

Сравнительные характеристики ключевых структур

Ключевая Структура	Объем ключевых данных, доставляемых одному корреспонденту	Объем ключевых данных, доставляемых в сети	Устойчивость к компрометации
Единый ключ	n_0	n_0M	0
Сетевой набор	$n_0(M-1)$	$n_0(M-1)M$	$M-2$
Базовый набор	n_0L	n_0LM	$L-1$

Видим, что ключевая структура базовый набор требует распределения в L раз большего объема ключевого материала, чем $КС$ единый ключ. Но это значительно меньше, чем требуется в ключевой структуре сетевой набор, поскольку предполагается, что $L \ll M$. В то же время гарантированно обеспечивается устойчивость данной $КС$ к компрометации $L - 1$ любых базовых наборов у произвольных корреспондентов сети.

Заметим, что для данного способа построения ключевой структуры объем адреса корреспондента составляет $n_0 L$ бит.

1.5. Распределение ключей для симметричных криптосистем с использованием ЦРК в интерактивном режиме

1.5.1. Принципы построения протоколов распределения ключей

Рассмотрим сеть, содержащую M корреспондентов, имеющих открытые каналы связи друг с другом и с центром распределения ключей. Каждый корреспондент имеет предварительно распределенные по доверенным каналам ключи для связи с ЦРК. В отличие от предыдущего способа полагается, что ЦРК используется корреспондентами постоянно. Для связи между собой корреспонденты ключей не имеют.

Для создания сеансового ключа любая пара корреспондентов выполняет протокол обмена данными, с участием ЦРК. Поскольку обмен данными в ходе выполнения протокола осуществляется по открытым каналам связи, возможны многочисленные атаки со стороны злоумышленника. Известно достаточно много типов атак, отличающихся по цели, механизму воздействия, использованию ресурсов, области применения и проч.[2,5].

1. Атака с повторной передачей сообщения.
2. Атака подмены.
3. Атака «человек посередине».
4. Атака с помощью параллельного сеанса.
5. Атака с помощью отражения сообщений.
6. Атака с помощью чередования сообщений.
7. Атака на основе безымянных сообщений.
8. Атака на основе неправильного выполнения криптографических операций.
9. Атака с известным сеансовым ключем.
10. Атака на основе отказа в обслуживании.
11. Атака с помощью связывания.

Построение протокола должно исключать или минимизировать эти атаки.

В условиях активного воздействий злоумышленника корреспондентам необходимо решить ряд задач, связанных с обеспечением конфиденциальности, целостности передаваемых сообщений, исключить возможность использования злоумышленником старых данных. Также корреспондентам необходимо взаимно аутентифицировать друг друга. Поэтому такие протоколы распределения ключей называют протоколами создания *аутентифицированного ключа*. В решении этих задач накоплен большой опыт [1–7], который можно сформулировать в виде следующих требований (принципов) построения протоколов распределения аутентифицированного ключа:

1) принцип парной конфиденциальности;
2) принцип аутентичности;
3) принцип новизны (свежести);
4) принцип взаимности;
5) принцип анонимности.

Рассмотрим более подробно эти принципы.

Принцип парной конфиденциальности является основным и естественным для системы распределения симметричных ключей. Корреспонденты, участвующие в формировании ключа должны быть уверены, что ключ известен только им двоим. Согласно данному принципу предполагается, что передаваемые в ходе создания ключа данные должны быть зашифрованы. При наличии третьей стороны (ЦРК) это означает, что все корреспонденты безусловно ей доверяют.

Принцип аутентичности означает целостность ключа и предполагает имитозащиту передаваемых в протоколе данных, а также аутентификацию корреспондентов, участвующих в формировании ключа. Если формирование ключа осуществляется с участием доверенной стороны, то должна быть гарантия, что доверенная сторона не будет имитировать работу корреспондентов, участвующих в формировании ключа.

Принцип новизны (свежести) означает, что ключ сформирован заново именно для данного сеанса работы. Это общепризнанная практика использования ключей в криптографии, обусловленная целым рядом причин. Одна из них заключается в том, что если ключом пользуются два корреспондента, и если один из них, например A, уверен в выполнении всех требований по обращению с ключом в ходе его применения, то у него не может быть полной уверенности в том, что эти требования выполняет другой корреспондент B. Поэтому ключ принято периодически обновлять. Обычно обновление ключа требуется при каждом новом сеансе связи, а в некоторых случаях при передаче нового пакета.

Принцип взаимности. Корреспондент, участвующий в создании ключа, должен быть уверен, что ключ сформирован именно с назначенным

(желаемым) корреспондентом. Когда корреспондент A контактирует с корреспондентом B он должен быть уверен, что корреспондент B существует и выполняет криптографический протокол от начала до конца. Доказывающий корреспондент должен подтвердить свое существование, выполнив некую криптографическую операцию, после события, которое проверяющий считает последним. Выполнение данного принципа также называется аутентификацией существования корреспондента.

Принцип анонимности. Означает, что вовлеченные в протокол формирования ключа стороны хотят скрыть от посторонних, в том числе злоумышленника свои идентификаторы (имена). В строгом смысле анонимность означает возможность отрицать авторство сообщения и никакой «наблюдатель» в сети не может доказать, что протокол выполняется конкретной парой корреспондентов, создающих ключ. Заметим, что реализация данного принципа не всегда возможна и необходима.

Рассмотрим один из основных протоколов формирования сеансового аутентифицированного ключа между корреспондентами A и B, предложенный Нидхемом–Шредером (Needham, Schroeder) [12].

1.5.2. Протокол Нидхема–Шредера с использованием симметричных ключей

Предполагается, что все корреспонденты связаны друг с другом и с центром распределения ключей открытыми каналами связи. ЦРК в любой момент доступен всем корреспондентам. Каждый корреспондент имеет предварительно распределенные по доверенным каналам ключи для конфиденциальной связи с ЦРК, которые обозначим как K_A для корреспондента A, K_B для корреспондента B, K_C для корреспондента C и т. д. Для связи между собой корреспонденты ключей пока не имеют. Для создания сеансового ключа пара корреспондентов A и B выполняет следующий протокол обмена данными, с участием ЦРК.

1. Корреспондент A генерирует случайное число N_A и посылает в ЦРК сообщение A, B, N_A, где A, B идентификаторы взаимодействующих корреспондентов (рис. 1.5).

2. ЦРК генерирует случайный ключ K_{AB} и посылает A криптограмму, зашифрованную на ключе K_A: $E_{K_A}\left(N_A, K_{AB}, B, E_{K_B}\left(K_{AB}, A\right)\right)$. Эта криптограмма содержит вложенную криптограмму на ключе K_B для узла B.

3. Корреспондент A расшифровывает криптограмму, проверяет совпадение числа N_A переданного им и числа N_A полученного из расшифрованной криптограммы и устанавливает, что ему прислан ключ K_{AB} для

связи с корреспондентом B. Корреспондент A вложенную криптограмму $E_{K_B}(K_{AB}, A)$ посылает корреспонденту B.

Корреспондент B расшифровывает криптограмму, устанавливает что ему прислан ключ для связи с корреспондентом A.

В итоге корреспонденты A и B имеют одинаковые сеансовые ключи, на которых они могут выполнять криптографический протокол. Далее A и B обмениваются тестовыми сообщениями, подтверждающими владение одинаковым ключом.

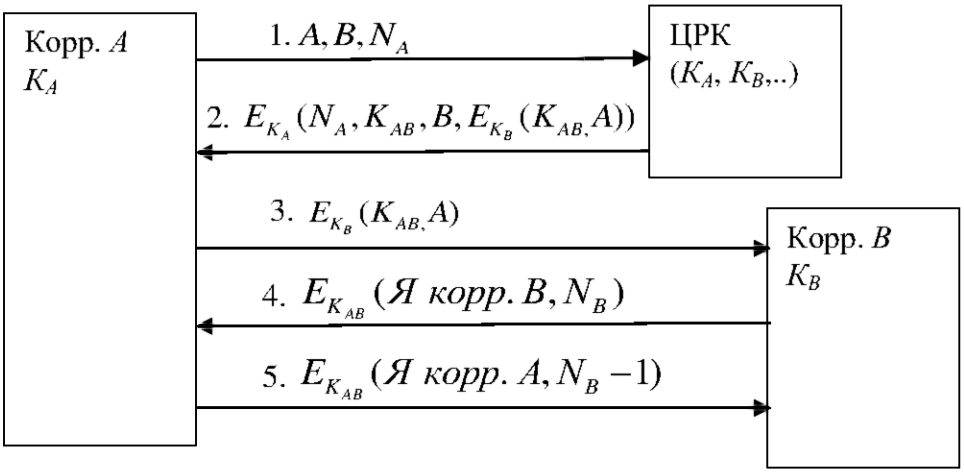


Рис. 1.5. Протокол распределения ключей Нидхема–Шредера

4. Корреспондент B посылает A криптограмму

$$E_{K_{AB}}(Я\ корр.\ B, N_B).$$

5. Корреспондент A расшифровывает эту криптограмму и отвечает B

$$E_{K_{AB}}(Я\ корр.\ A, N_B - 1).$$

Корреспондент B расшифровывает эту криптограмму и проверяет, что он получил число $N_B - 1$ в ответ на переданное им число N_B.

На этом действие протокола Нидхема–Шредера (НШ) заканчивается. Заметим, что здесь и далее мы используем термин *расшифрование*, как синоним термину *дешифрование*, введенному в первой части.

Пересылка сеансового ключа K_{AB} в зашифрованном виде обеспечивает его конфиденциальность. Числа N_A, N_B в протоколе обеспечивают «свежесть» сообщений, которыми обмениваются участники протокола для каждого сеанса. Аутентификация сторон осуществляется путем демонстрации своих криптографических возможностей через процедуру шифрования-расшифрования, поскольку правильное восстановление числа $N_A(N_B)$ доказывает, что корреспондент владеет ключом, что и аутентифицирует его.

Однако не все требования к протоколам аутентифицированного распределения ключей выполняются. Злоумышленник может осуществить ряд атак. Рассмотрим их.

1. Атака по нарушению принципа взаимности.

На втором этапе протокола НШ корреспондент A расшифровывает криптограмму, полученную от ЦРК, и проверяет совпадение случайного числа N_A, содержащегося в ней, со случайным числом N_A, которое он отправлял ЦРК. Совпадение чисел убеждает A, что ЦРК выполнил шифрование после получения N_A. То есть A убеждается в том, что ЦРК существовал после получения N_A.

Однако этого нельзя сказать о корреспонденте B, он не имеет никаких доказательств существования ЦРК. ЦРК может доказать свое существование, если корреспондент B пошлет ему случайное число N_B, которое ЦРК вставит в свое сообщение для корреспондент В вместе с сеансовым ключом. Однако это приведет к увеличению количества обменов в протоколе, что нежелательно.

Другим решением этой задачи может быть использование ЦРК меток времени T, которые он встраивает в передаваемые сообщения. Рассмотрим протокол НШ с введением меток времени:

1. $A \rightarrow$ЦРК: A, B.
2. ЦРК$\rightarrow A$: $E_{K_A}\left(B, K_{AB}, T, E_{K_B}\left(K_{AB}, A, T\right)\right)$.
3. $A \rightarrow B$: $E_{K_B}\left(K_{AB}, A, T\right)$.
4. Так же, как в исходном протоколе.
5. Так же, как в исходном протоколе.

Получив сообщения на 2-м и 3-м этапах, корреспонденты A и B проверяют неравенство $\left|T_{\text{текущее время}} - T\right| \le \Delta T$, где ΔT допустимое расхождение времени у корреспондентов с учетом задержек, связанных с распространением сигнала в канале связи и обработкой принятых сообщений.

При выполнении неравенства сообщение принимается. Так как метки времени зашифрованы имитация ЦРК злоумышленником будет затруднена, сама же метка предотвращает атаку с повторением. Метки времени требуют точных системных часов, что создает новую проблему, решение которой является нетривиальным.

2. Атака на основе использования старого сеансового ключа.

Предположим, что нарушитель каким-либо образом узнал ключ K'_{AB}, который был использован на одном из предыдущих сеансов связи (например, из-за небрежного хранения ключа одним пользователем).

Злоумышленник активизируется на 3-м этапе протокола НШ.

3’. Он перехватывает сообщение, направленное от A к B, и заменяет его материалом перехвата старого сеанса между A и B – $E_{K_B}\left(K'_{AB}, A\right)$.

4’. Корреспондент B расшифровывает криптограмму, устанавливает кем было отправлено сообщение и посылает корреспонденту A сообщение: $E_{K'_{AB}}\left(Я\ корр.\ B, N_B\right)$.

5’. Злоумышленник отвечает B криптограммой

$$E_{K'_{AB}}\left(Я\ корр.\ A, N_B - 1\right).$$

В результате атаки корреспондент B думает, что он имеет ключ для связи с корреспондентом A, но на самом деле этот ключ является старым и известным злоумышленнику. При этой атаке нарушаются принципы конфиденциальности и взаимности.

Более удачным решением по защите от этой атаки является протокол Отвея–Рииса, к рассмотрению которого мы перейдем после обсуждения следующего замечания.

Заметим, что в рассмотренном выше НШ протоколе в качестве криптографического преобразования, используемого для защиты пересылаемых данных, использовано шифрование, обеспечивающее конфиденциальность передаваемых данных.

Однако известно, что шифрование не решает задачу имитозащиты передаваемых сообщений. Поскольку для шифрования может быть использован блочный шифр, то необходимо защитить блоки защифрованного сообщения каким-либо методом, гарантирующим целостность данных. Такие методы известны. Это либо применение кодов аутентификации сообщений (*MAC*), либо ключевых хэш-функций типа *HMAC* (п. 4 ч. 1).

Рассмотрим модификацию протокола НШ, с введением дополнительной имитозащиты передаваемых блоков. Обозначим *MAC*-код сообщения M, полученный с помощью ключа K, как $MAC_K\left[M\right]$.

1. $A \rightarrow$ЦРК: A, B, N_A

2. ЦРК$\rightarrow A$: $E_{K_A}\left(K_{AB}\right), N_A, A, B, MAC_{K_A}\left[E_{K_A}\left(K_{AB}\right), N_A, A, B\right]$

$E_{K_B}\left(K_{AB}\right), T, A, B, MAC_{K_B}\left[E_{K_B}\left(K_{AB}\right), T, A, B\right]$

3. $A \rightarrow B$: $E_{K_B}\left(K_{AB}\right), T, A, B, MAC_{K_B}\left[E_{K_B}\left(K_{AB}\right), T, A, B\right]$

4. $B \rightarrow A$: $N_B,\ MAC_{K_{AB}}\left[N_B\right]$

5. $A \rightarrow B$: $N_B - 1,\ MAC_{K_{AB}}\left[N_B - 1\right]$

Видим, что в этой модификации протокола сообщения, передаваемые на шагах 2–5 протокола, защищены от модификации. Наличие чисел N_A и T убеждает корреспондентов в свежести ключа и взаимной аутентичности сторон.

1.5.3. Протокол Отвея–Рииса

Предложен Otway D., Rees O. [13] в целях устранения недостатка протокола НШ, связанного с возможностью использования старых сеансовых ключей (рис. 1.6). Основная идея данного протокола заключается в отслеживании на каждом этапе обмена порядкового номера идентификатора сеанса.

1. Корреспондент A генерирует сообщение, состоящее из порядкового номера N сеанса, собственного идентификатора и идентификатора вызываемого корреспондента B, добавляет к этому сообщению случайное число N_A и шифрует все сообщение на ключе K_A. Криптограмму и исходное сообщение корр, A передает корреспонденту – A, B, N, $E_{KA}(A, B, N, N_A)$.

2. Корреспондент B генерирует сообщение, состоящее из случайного числа N_B, порядкового номера N сеанса, идентификаторов корреспондентов A и B. Сообщение шифруется на ключе K_B. Затем B отправляет криптограмму B, $E_{KB}(A, B, N, N_B)$ в ЦРК вместе с зашифрованным сообщением A, B, N, $E_{KA}(A, B, N, N_A)$ от корреспондента A.

3. ЦРК генерирует случайный сеансовый ключ. Затем он создает два сообщения. Одно сообщение, включающее случайное число корреспондента A и сеансовый ключ, шифруется на ключе K_A. Второе сообщение, включающее случайное число корреспондента В и сеансовый ключ, шифруется на ключе K_B.

ЦРК посылает корреспонденту B оба сообщения вместе с порядковым номером сеанса: N, $E_{KA}(N_A, K_{AB})$, $E_{KB}(N_B, K_{AB})$.

4. Корреспондент B посылает корреспонденту A криптограмму, зашифрованную на его ключе и порядковый номер сеанса: N, $E_{KA}(N_A, K_{AB})$.

5. Корреспондент A расшифровывает сообщение, восстанавливает сеансовый ключ и число N_A.

Если все случайные числа совпадают, а порядковый номер сеанса не изменился по ходу протокола, то A и B убеждаются во взаимной аутентификации друг друга и получают общий сеансовый ключ.

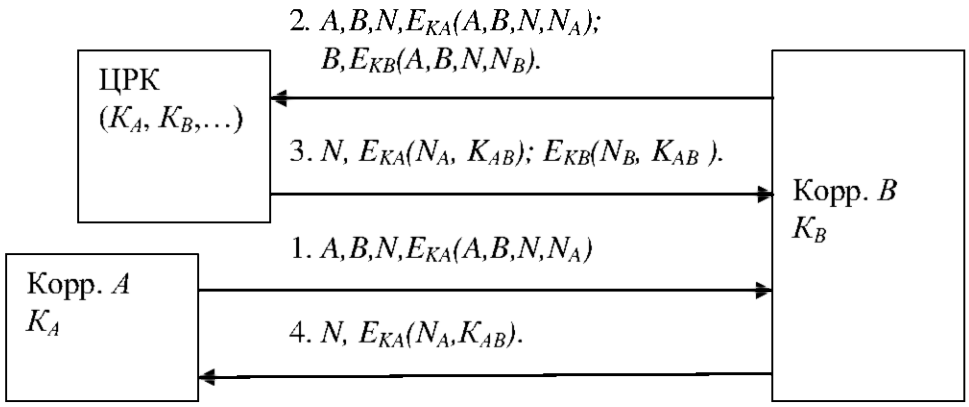


Рис. 1.6. Протокол распределения ключей Отвея–Рииса

Однако в [2] отмечается, что практическая реализация этого протокола может быть подвержена некоторой атаке. Действительно, если шифрование на шаге 3 будет осуществляться по блокам, например, используя режим ECB, то эти блоки злоумышленник может модифицировать, так что это будет незаметно при расшифровке, что и приведет к уязвимости протокола. Поэтому следует очень внимательно подходить к выбору алгоритмов шифрования при реализации протокола, например, выбрав режим CBC.

1.5.4. Протокол Нидхема–Шредера с использованием открытых ключей (НШО)

Введем предварительно дополнительные обозначения.

Обозначим закрытый ключ и открытый ключ корреспондента A как SK_A, PK_A соответственно. Операции шифрования сообщения M и расшифрования криптограммы E на этих ключах обозначим $E_{PK_A}(M)$ и $D_{SK_A}(E)$ соответственно. Цифровую подпись сообщения M, получаемую с помощью ключа SK_A обозначим как $S = sign_{SK_A}\{M\}$. Верификацию цифровой подписи с помощью открытого ключа PK_A обозначим как $Ver_{PK_A}\{S\}$.

Так же, как в протоколе НШ с симметричными ключами, мы рассматриваем сеть, содержащую M корреспондентов (A, B,…), имеющих открытые каналы связи друг с другом и с центром распределения ключей, который в любой момент доступен каждому корреспонденту. ЦРК имеет открытые ключи каждого корреспондента, а каждый корреспондент имеет свой закрытый ключ и открытый ключ ЦРК. Цель протокола состоит в возможности создания общего сеансового ключа для любой пары корреспондентов.

Протокол НШО имеет следующие шаги (рис. 1.7).

1. $A \rightarrow$ЦРК: A, B.

2. ЦРК$\rightarrow A$: $Sign_{SK\ \text{ЦРК}}\{PK_B, B\}$.

3. Корреспондент A проверяет подпись $Ver_{PK\ \text{ЦРК}}\{PK_B, B\}$, генерирует одноразовое случайное число N_A и используя полученный от ЦРК открытый ключ корреспондент B посылает ему криптограмму

$A \rightarrow B$: ЦРК, $E_{PK_B}(N_A, A)$.

4. Корреспондент B расшифровывает криптограмму $D_{SK_B}\left(E_{PK_B}(N_A, A)\right)$, определяет идентификатор вызывающего корреспондента и посылает запрос в ЦРК

$B \rightarrow$ЦРК: B, A.

5. ЦРК$\rightarrow B$: $Sign_{SK\ \text{ЦРК}}\{PK_A, A\}$.

6. Корреспондент B проверяет подпись $Ver_{PK\ \text{ЦРК}}\{PK_A, A\}$, генерирует одноразовое случайное число N_B и используя полученный от ЦРК открытый ключ корреспондент A посылает ему криптограмму

$B \rightarrow A$: ЦРК, $E_{PK_A}(B, N_A, N_B)$.

7. Корреспондент A расшифровывает криптограмму $D_{SK_A}\left(E_{PK_A}(B, N_A, N_B)\right)$, и посылает B криптограмму

$A \rightarrow B$: $E_{PK_B}(N_B, A)$.

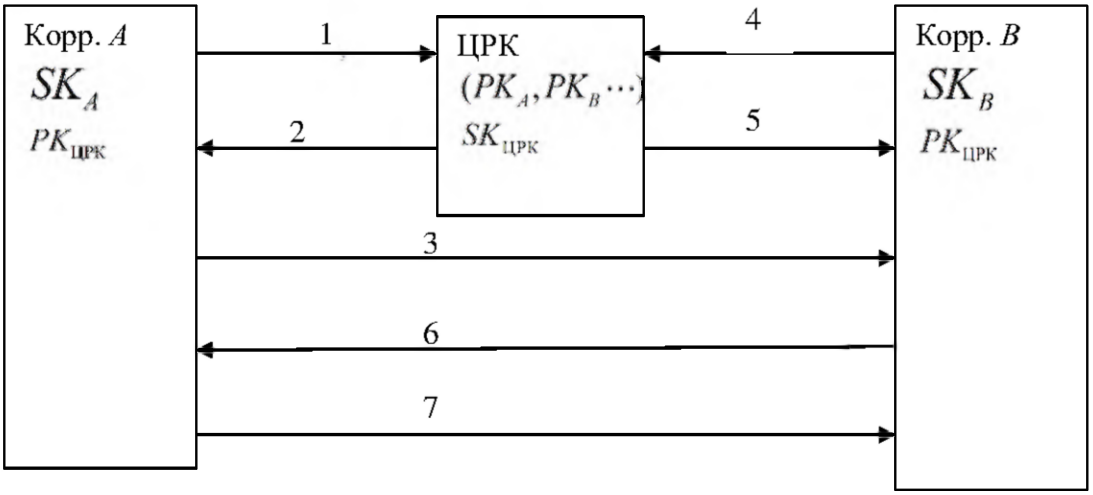


Рис. 1.7. Протокол Нидхема–Шредера с использованием асимметричных ключей

По числу N_A корреспондент A убеждается в правильности выполнения всех этапов протокола, в том числе убеждается в существовании корреспондента B поскольку только он мог расшифровать сообщение на этапе 3, содержащее число N_A. Аналогично корреспондент B убеждается в существовании корреспондента A поскольку он вернул ему число N_B, предварительно расшифровав криптограмму ЦРК и зашифровав его с помощью ключа PK_B (шаг 7).

Далее корреспонденты A и B, обладая случайными числами N_A и N_B, по какому-либо алгоритму формируют общий секретный ключ, например, $K_{AB} = N_A + N_B$, для шифрования в симметричной криптосистеме.

Рассмотрим атаки злоумышленника на данный протокол.

Заметим, что в этом протоколе на этапах 3, 6, 7, используются операции шифрования с открытым ключом. Поскольку, как уже отмечалось выше, шифрование не решает автоматически задачу имитозащиты, то на этих этапах потенциально возможны атаки, связанные с навязыванием ложных

сообщений. Одной из таких атак является атака «с помощью параллельного сеанса», предложенная Лоу [14]. Суть атаки можно пояснить рис. 1.8.

1-й сеанс A–C	2-й сеанс C (под видом A) – B
1. A→ЦРК: A, C	
2. ЦРК→A: $Sign_{SK\ \text{ЦРК}}\{PK_C, C\}$	
3. A→C: $E_{PK_C}(N_A, A)$	3’. $C(A)$→B: $E_{PK_B}(N_A, A)$
4. C→ЦРК: C, A	4’. B →ЦРК: B, A
5. ЦРК→C: $Sign_{SK\ \text{ЦРК}}\{PK_A, A\}$	5’. ЦРК→C: $Sign_{SK\ \text{ЦРК}}\{PK_A, A\}$
6. C→A: $E_{PK_A}(B, N_A, N_B)$	6’. B→$C(A)$: $E_{PK_A}(B, N_A, N_B)$
7. A→C: $E_{PK_C}(N_B, A)$	7’. A→C: $E_{PK_B}(N_B, A)$

Рис. 1.8. Атака на протокол НШ с открытыми ключами
с помощью параллельного сеанса

Предположим, что злоумышленник является законным корреспондентом сети – C. Атака начинается после того, как корреспондент A выходит на связь со злоумышленником. В этой атаке корреспондент C может поддерживать два самостоятельных сеанса протокола НШО: 1-й сеанс это почти корректный протокол с корреспондентом A; 2-й сеанс – злоумышленник, маскируясь под корреспондента A, выполняет протокол НШО с корреспондентом B. Нас будут интересовать шаги 3, 6, 7 протоколов, т. е. те, в которых используется шифрование.

На шаге 3 злоумышленник получает зашифрованное случайное число от корреспондент A, и расшифровывает его своим закрытым ключем. Далее он продолжает выполнять 4 и 5 шаг протокола и одновременно открывает 2-й сеанс с корреспондентом B. Для этого он шифрует случайное число корреспондента A (мы предполагаем, что открытый ключ корреспондента у него есть) и посылает его корреспонденту B. В соответствии с протоколом НШ корреспондент B выполняет 4’и 5’шаги протокола. Затем на шаге 6’ он, полагая, что работает с корреспондентом A, посылает ему свое случайное число N_B, зашифрованное на ключе корреспондента A. Эту криптограмму получает злоумышленник. Расшифровать он ее не может, поэтому он пересылает ее корреспонденту A в качестве сообщения шага 6 1-го сеанса. Корреспондент A расшифровывает криптограмму, а восстановленное случайное число N_B шифрует на открытом ключе злоумышленника и отправляет ему. Теперь злоумышленник может расшифровать это число и переслать корреспонденту B на 7-м шаге 2-го сеанса. На этом выполнение протоколов в обоих сеансах завершается. В итоге корреспондент A создал ключ с корреспондентом C (злоумышленником), а злоумышленник под видом корреспондента A создал ключ с корреспондентом B,

который будет полагать, что работает с корреспондентом A. Эта атака показывает недостаточность применения шифрования в протоколе, требующем аутентификации и является примером нарушения принципа взаимности при формировании ключа.

Для устранения этой атаки необходимо связать случайные числа с идентификаторами корреспондентов, что не позволит манипулировать ими в разных сеансах. Для этого необходимо использовать цифровую подпись. Тогда сообщения, передаваемые на шагах 3, 6, 7 (рис. 1.7), можно представить в виде

3. $A \rightarrow B$: ЦРК, $E_{PK_B}\left(Sig_{SK_A}\{N_A, A\}\right)$.

То есть пересылаемое от A к B сообщение – случайное число N_A, сгенерированное корреспондентом A, подписывается им, затем шифруется на открытом ключе корреспондента B.

Аналогично

6. $B \rightarrow A$: ЦРК, $E_{PK_A}\left(Sign_{SK_B}\{B, N_A, N_B\}\right)$

7. $A \rightarrow B$: $E_{PK_B}\left(Sig_{SK_A}\{N_B, A\}\right)$

Однако и в этой модификации протокола у злоумышленника остается небольшая лазейка вмешательства в протокол. Действительно, злоумышленник C, будучи законным корреспондентом сети, может начать взаимодействовать с корреспондентом A и получить от него сообщение на 3-м шаге

3. $A \rightarrow C$: ЦРК, $E_{PK_C}\left(Sig_{SK_A}\{N_A, A\}\right)$.

Далее корреспондент C расшифровывает криптограмму, восстанавливает сообщение N_A из подписи и передает сообщение корреспонденту B под видом A.

3'. C (под видом A)$\rightarrow B$: ЦРК, $E_{PK_E}\left(Sig_{SK_A}\{N_A, A\}\right)$

То есть злоумышленник C может инициировать протокол взаимодействии с B. Однако переслать сообщение от B к A на этапе 6 своего сеанса злоумышленник не сможет, поскольку содержание криптограммы подписано корреспондентом B.

Чтобы исключить возможность и этой атаки, на этапе 3'достаточно поменять очередность выполнения операций шифрования и цифровой подписи. Тогда шаги 3, 6, 7 протокола можно записать в виде

3. $A \rightarrow E$: $Sig_{SK_A}\left\{E_{PK_E}(N_A, A)\right\}$.

6. $B \rightarrow A$: $Sig_{SK_B}\left\{E_{PK_A}(B, N_A, N_B)\right\}$.

7. $A \rightarrow B$: $Sig_{SK_A}\left\{E_{PK_B}(N_B, A)\right\}$.

Некоторые другие более изощренные атаки описаны в [2, 5].

1.6. Способы распределения ключей на основе взаимного обмена данными между участниками протокола

В предыдущих параграфах рассматривались способы распределения ключей на основе использования ЦРК и существования защищенных каналов доставки ключей от него к корреспондентам.

Однако известна большая группа способов распределения ключей, когда корреспонденты формируют общие сеансовые ключи, обмениваясь друг с другом информацией, не используя доверенные каналы и ЦРК. В этом случае задача распределения ключей сводится к задаче согласования ключей между корреспондентами. К таким способам относятся:

- способы распределения ключей для симметричных систем на основе ассиметричной криптографии;
- способы распределения ключей для симметричных систем на основе квантовых каналов;
- способы распределения ключей на основе использования каналов связи с шумом;
- способы распределения ключей на основе рандомизации параметров канала связи.

Рассмотрим первую группу способов. Три другие находятся на стадии научных исследований [22–28] и кратко описаны в первой части книги.

Отсутствие ЦРК, как естественного регулятора обмена сообщениями в сети, создает условия для многочисленны атак со стороны злоумышленников, в том числе являющихся корреспондентами сети.

Протокол формирования ключей на основе асимметричной криптографии должен строиться на принципах, изложенных в п. 1.4.2.

Эволюция протоколов, использующих ассиметричную криптографию, показана на рис. 1.9.

Рис. 1.9. Эволюция протоколов согласования ключей на основе ассиметричной криптографии

Протокол Диффи- Хеллмана

Первым и наиболее распространенным протоколом, формирования ключа на основе асимметричной криптографии является протокол Диффи–Хеллмана [15], который был рассмотрен в п. 3.3 ч. 2.

В этом протоколе общий ключ вырабатывается на основе взаимного обмена несекретными сообщениями α^x и α^y между корреспондентами *A* и *B*, передаваемыми по открытым каналам связи.

Однако практическое использование данного протокола небезопасно, поскольку он подвержен атаке «человек посередине» (рис. 1.10). Рассмотрим данную атаку более подробно.

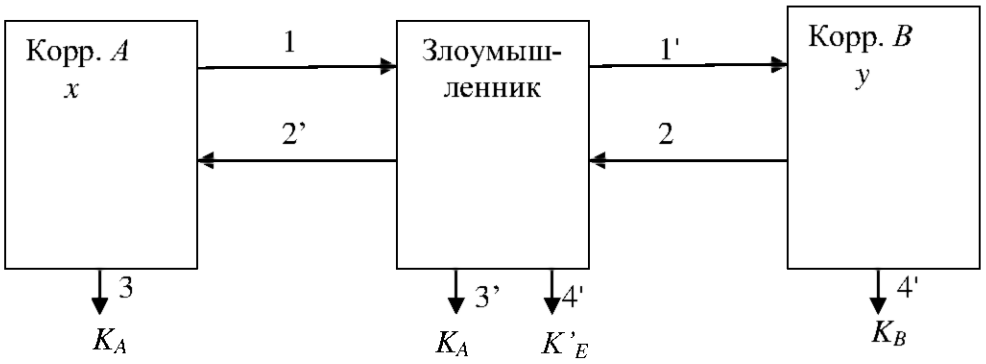


Рис. 1.10. Атака человек посередине в схеме Диффи–Хеллмана

1. Корреспондент *A* генерирует случайное число $x \in [1, p-1]$, вычисляет $g_A = \alpha^x \bmod p$ и посылает полученное значение корреспонденту *B*.

2. Корреспондент *B* генерирует случайное число $y \in [1, p-1]$, вычисляет значение $g_B = \alpha^y \bmod p$ и посылает полученное значение корреспонденту *A*.

1'. Злоумышленник *E* перехватывает g_A сохраняет его в памяти, генерирует случайное число $e \in [1, p-1]$, находит $g_E = \alpha^e \bmod p$ и посылает его корреспонденту *B* по видом корреспондента *A*.

2'. Злоумышленник *E* перехватывает g_B сохраняет его в памяти, генерирует случайное число $e' \in [1, p-1]$, находит $g_E = \alpha^{e'} \bmod p$ и посылает его корреспонденту *A* по видом корреспондента *B*.

3. Корреспондент *A* вычисляет значение сеансового ключа

$$K_A = g_E^x \bmod p = \alpha^{ex} \bmod p .$$

3'. Злоумышленник находит

$$K_E = g_A^e \bmod p = \alpha^{xe} \bmod p .$$

4. Корреспондент *B* вычисляет значение сеансового ключа

$$K_B = g_E^y \bmod p = \alpha^{e'y} \bmod p .$$

4’. Злоумышленник находит

$$K'_E = g_B^{e'} \bmod p = \alpha^{ye'} \bmod p .$$

Очевидно, что $K_A = K_E$ и $K_B = K'_E$.

Таким образом, злоумышленник сформировал общий ключ K_E с корреспондентом A и общий ключ K'_E с корреспондентом B. Если теперь корреспондент A будет посылать зашифрованное сообщение на ключе K_A, то злоумышленник расшифрует это сообщение на ключе K_E и повторно зашифрует на ключе K'_E. Далее он отправит криптограмму корреспонденту В, который ее расшифрует на ключе K_B. При этом корреспондент A считает, что он непосредственно работает с корреспондентом B, а корреспондент B полагает, что он работает непосредственно с корреспондентом A. Хотя фактически обмен между корреспондентами A и B идет под контролем злоумышленника E.

Видим, что ни один из принципов построения протоколов аутентифицированного распределения ключей (п. 1.5.1), здесь не выполняется.

Поэтому рассмотрим усовершенствованные протоколы и проведем анализ реализации в них изложенных выше принципов, сформулировав их в виде следующих требований:

1) конфиденциальность ключа;

2) имитозащищенность ключа и аутентификация корреспондентов;

3) взаимное согласование ключа;

4) анонимность корреспондентов, формирующих ключ.

Для упрощения анализа требование новизны рассматривать не будем, полагая, что оно выполняется введением в передаваемые сообщения случайных чисел.

Протокол STS (Station to Station)

Протокол был предложен Диффи и соавторами [16] для устранения главного недостатка протокола *DH* неустойчивости к атаке «человек посередине».

Суть усовершенствования состоит в том, что числа α^{δ} и α^{y}, которыми обмениваются корреспонденты A и B, подписываются цифровыми подписями этих корреспондентов. Для реализации протокола необходимо провести предварительное распределение ключей, чтобы выполнялись следующие условия.

1. Корреспонденты A и B имеют закрытые ключи подписи (SK_A, SK_B) соответственно.

2. Корреспонденты A и B владеют сертификатами открытых ключей, подписанные удостоверяющим центром (УЦ):

$$Cert_A = Sig_{\text{УЦ}}(A, PK_A), \quad Cert_B = Sig_{\text{УЦ}}(B, PK_B).$$

Понятие сертификата ключа будет рассмотрено в п. 2, сейчас же для простоты будем полагать, что сертификат, представляет собой открытый ключ корреспондента, подписанный специальным органом – удостоверяющим центром (УЦ), которому доверяют все корреспонденты сети.

3. Корреспонденты *A* и *B* имеют открытый ключ удостоверяющего центра $PK_{\text{УЦ}}$ для верификации сертификатов.

Протокол STS может быть записан так.

1. Корреспондент *A* генерирует случайное число $x \in [1, p-1]$, вычисляет $g_A = \alpha^x \bmod p$ и посылает полученное значение корреспонденту *B*.

2. Корреспондент *B* генерирует случайное число $y \in [1, p-1]$, вычисляет значение $g_B = \alpha^y \bmod p$ и посылает корреспонденту *A* сообщение

$$g_B, Cert_B, E_K\left(Sig_{SK_B}\{g_A, g_B\}\right),$$

где $K = (g_A)^y \bmod p = \alpha^{xy} \bmod p$ – ключ, вычисленный корреспондентом *B*; $Sig_{SK_B}\{g_A, g_B\}$ – подписанное корреспондентом *B* сообщение, содержащее значения g_A, g_B.

3. Корреспондент *A*, получив g_B, находит ключ $K = (g_B)^x \bmod p = \alpha^{yx} \bmod p$, расшифровывает криптограмму $E_K()$ и верифицирует подпись $Sig_{SK_B}\{g_A, g_B\}$, используя для этого открытый ключ корреспондента *B*, содержащийся в сертификате $Cert_B$. Правильная верификация подписи свидетельствует о подлинности значений g_A, g_B. Далее корреспондент *A* отправляет корреспонденту *B* сообщение $Cert_A$, $E_K\left(Sig_{SK_A}\{g_A, g_B\}\right)$.

4. Корреспондент *B*, приняв это сообщение, выполняет действия аналогичные корреспонденту *A* и убеждается в подлинности значений g_A, g_B, а следовательно, и аутентичности общего ключа *K*.

Проведем анализ выполнения требований к протоколу *STS*, изложенных ранее.

Поскольку сообщения на шагах 2 и 3 передаются в зашифрованном и подписанном виде, то атака «человек посредине» не приведет к нарушению конфиденциальности и аутентичности, формируемого ключа.

Наличие шифрования также не позволяет злоумышленнику заменить подпись корреспондента *B* и соответственно провести атаку путем замены сертификата и подписи на третьем шаге протокола, что привело бы к нарушению взаимной согласованности. Именно обмен зашифрованными сообщениями на общем ключе и их правильное расшифрование позволяют корреспондентам *A* и *B* надеяться, что они оба сформировали идентичные ключи.

Однако, как было показано при анализе протокола НШО, шифрование на симметричном ключе не обеспечивает имитозащищенности передаваемых сообщений и в некоторых случаях оставляет лазейку для осуществления атаки нарушения взаимности формирования ключа.

Рассмотрим эти случаи. Предположим, что в протоколе *STS* используется шифрование на основе гаммирования. Пусть в целях обеспечения анонимности на 3-м шаге корреспондент *A* передает корреспонденту *B* сообщение $E_K\left(Cert_A, Sig_{SK_A}\{g_A, g_B\}\right)$. Тогда злоумышленник может перехватить эту криптограмму и передать корреспонденту B сообщение

$$E_K\left(Cert_A, Sig_{SK_A}\{g_A, g_B\}\right) \oplus \left(Cert_A \oplus Cert_E, 0^l\right),$$

где 0^l – последовательность из l нулей на позициях, соответствующих подписи $Sig_{SK_A}\{g_A, g_B\}$ в криптограмме.

Расшифровав эту криптограмму, корреспондент *B* получит сертификат корреспондента *E* и используя открытый ключ в нем будет пытаться проверить подпись $Sig_{SK_E}\{g_A, g_B\}$.

Таким образом, корреспондент *A* будет думать, что он установил связь с корреспондентом *B*, а *B* будет полагать, что установил связь с *E*, но по каким-то причинам подпись не верифицируется. Далее он будет обращаться к *E* для выяснения причины, что в итоге равносильно не выполнению взаимной согласованности формирования ключа.

Другой тип атаки состоит в следующем. Злоумышленник, как корреспондент сети, представляется корреспонденту *B* под своим именем, а корреспонденту *A* представляется как *B*.

На 1-м шаге злоумышленник получает от корреспондента *A* сообщение $g_A = \alpha^x \bmod p$ и от своего имени передает его корреспонденту *B*.

На 2-м шаге корреспондент *B* формирует сообщение $g_B, Cert_B, E_K\left(Sig_{SK_B}\{g_A, g_B\}\right)$, которое злоумышленник передает корреспонденту *A*.

На 3-м шаге корреспондент *A* посылает злоумышленнику сообщение $Cert_A, E_K\left(Sig_{SK_A}\{g_A, g_B\}\right)$.

Поскольку это сообщение подписано корреспондентом *A* и зашифровано, переслать его корреспонденту *B* от своего имени он не может. На этом выполнение протокола приостанавливается. В итоге *A* считает, что распределил ключ с корреспондентом *B*, а *B*, не получив ответа от корреспондента A, прерывает выполнение протокола.

Дальнейшие попытки корреспондента *A* установить связь с корреспондентом *B* будут пресекаться, поскольку последний полагает, что вступил в контакт со злоумышленником, а не с корреспондентом *A*,

и никто не может сообщить корреспонденту A о неправильном завершении протокола.

Хотя злоумышленнику и не удалось овладеть секретным ключем, ему удалось нарушить согласованность взаимодействия корреспондентов A и B.

Более того, если предположить, что корреспондент A является сервером, обслуживающим многих клиентов, то он резервирует ресурс для дальнейшего взаимодействия с этим корреспондентом. И если сервер подвергается DDOS-атаке, то за счет неоправданного резервирования ресурсов его мощность резко падает. DDOS-атака достигает цели.

Эти недостатки протокола *STS* частично устранены в протоколе международной организации по стандартизации (International Organization for Standartization – ISO).

ISO протокол

Рассмотрим сначала версию протокола без шифрования.

1. $A \rightarrow B$: $Cert_A, g_A$
2. $B \rightarrow A$: $Cert_B, g_B, Sig_{SK_B}\{g_A, g_B, A\}$
3. $A \rightarrow B$: $Sig_{SK_A}\{g_A, g_B, B\}$.

Далее корреспонденты A и B находят общий ключ по алгоритму Диффи–Хеллмана $K_A = (g_B)^x$, $K_B = (g_A)^y$.

Конфиденциальность ключа основывается на алгоритме Диффи–Хеллмана. За счет использования подписей обеспечивается аутентификация значений g_A и g_B, которыми обмениваются корреспонденты.

Атака злоумышленника, нацеленная на нарушение связности формирования ключа, нейтрализуется введением идентификаторов корреспондентов в подписываемые сообщения на 2-м и 3-м шага протокола.

Таким образом, первые три требования, предъявляемые к протоколу, выполняются.

Перейдем к анализу выполнения требования анонимности выполнения протокола.

Как следует из протокола, в сообщениях, передаваемых на 1-м и 2-м шагах, присутствуют идентификаторы (сертификаты) корреспондентов, которые необходимы для проверки подписей. Наличие идентификаторов не позволяет обеспечь анонимность выполнения протокола.

Следующей модификацией протокола эта задача решается по отношению к пассивному злоумышленнику

1. $A \rightarrow B$: g_A
2. $B \rightarrow A$: $g_B, E_K(Cert_B)$
3. $A \rightarrow B$: $E_K\left(A, Cert_A, Sig_{SK_A}\{g_A, g_B, B\}\right)$
4. $B \rightarrow A$: $E_K\left(Sig_{SK_B}\{g_A, g_B, A\}\right)$,

где $K = \alpha^{yx} \bmod p$ ключ, сформированный по алгоритму Диффи–Хеллмана.

Как видно, идентификаторы корреспондентов на 2-м и 3-м шагах передаются в зашифрованном виде. Шифрование на симметричном ключе в свою очередь может создавать проблемы, связанные с имитозащитой передаваемых сообщений, и как в протоколе *STS* приводить нарушению связности формирования ключа. Эта проблема может быть решена введением в передаваемые сообщения аутентификаторов *MAC*-кодов.

Обратим также внимание на то, что в последнем варианте ISO-протокола потребовалось четыре обмена для формирования общего ключа.

Протокол SIGMA

Название протокола происходит от двух слов SIGnature (подпись) и Massage Authentication Code (код аутентификации сообщений) [17, 18]. Эти преобразования составляют основу протокола.

Базовый протокол SIGMA

Также как и ранее p – простое число, α – примитивный элемент поля $GF(p)$. Предполагается, что корреспонденты имеют свои закрытые ключи и сертификаты открытых ключей.

1. $A \rightarrow B$: g_A
2. $B \rightarrow A$: $g_B, B, Sig_{SK_B}\{g_A, g_B\}, MAC_{K_a}[B]$
3. $A \rightarrow B$: $A, Sig_{SK_A}\{g_A, g_B\}, MAC_{K_a}[A]$.

Выход протокола: сессионный ключ $K_s = \alpha^{yx} \bmod p$ и ключ для выполнения аутентификации K_a.

Первый базовый элемент протокола $Sig_{SK}\{g_A, g_B\}$ обеспечивает защиту значений Диффи–Хеллмана g_A, g_B от подмены и модификации, защита от атак повтора обеспечивается тем, что экспоненты g_A, g_B каждый раз меняются или в подписываемое сообщение вносятся случайные числа.

Второй базовый элемент протокола состоит в применении *MAC* аутентификаторов, полученных на основе ключа K_a. За счет использования *MAC* стороны демонстрируют обладание общим ключем, что обеспечивает выполнение требования взаимной согласованности и защиту от атаки нарушения связности.

Однако, как видно базовый протокол не обеспечивает защиту идентификаторов и, следовательно, анонимность корреспондентов. Поэтому рассмотрим далее модификации протокола, в которых решается эта задача.

Протокол SIGMA-I

Протокол обеспечивает защиту идентификатора инициатора протокола.

Исходные данные: те же, что в предыдущем протоколе.

1. $A \rightarrow B$: g_A

2. $B \rightarrow A$: $g_B, E_{K_e}\left(A, Sig_{SK_B}\{g_A, g_B\}, MAC_{K_a}[B]\right)$

3. $A \rightarrow B$: $E_{K_e}\left(A, Sig_{SK_A}\{g_A, g_B\}, MAC_{K_a}[A]\right)$

Выход протокола: сессионный ключ $K_s = \alpha^{yx} \bmod p$, ключ для выполнения аутентификации K_a, ключ шифрования K_e.

Как видно, этот протокол отличается от базового тем, что во 2-м и 3-м сообщениях используется шифрование на ключе K_e, полученном из ключа K_s. Для пассивного злоумышленника, обеспечивается конфиденциальность идентификаторов *A* и *B* обеих корреспондентов за счет их шифрования.

Если злоумышленник активный, например, он выдает себя за корреспондента *A*, то при приеме g_B во втором сообщении он может сформировать ключи K_s, K_e и расшифровать второе сообщение. Далее в третьем сообщении атака продолжена не будет, поскольку злоумышленник не имеет ключа подписи корреспондента *A*. Однако частично атака достигает цели, так как, расшифровав второе сообщение, злоумышленник узнает идентификатор корреспондента, с которым начал устанавливать связь.

Пусть наоборот, активный злоумышленник выдает себя за корреспондента *B*. Тогда, получив от корреспондента *A* сообщение g_A, он может сформировать второе сообщение, в котором подпись $Sig_{SK_E}\{g_A, g_B\}$ не будет соответствовать истинной подписи. Это будет обнаружено корреспондентом *A* и он прекратит выполнять протокол, не раскрыв своего идентификатора.

Таким образом, при активной атаке злоумышленника протокол *SIGMA-I* обеспечивает защиту только идентификатора инициатора протокола.

Рассмотрим другой вариант базового протокола *SIGMA*.

Протокол SIGMA-R

Протокол обеспечивает защиту идентификатора респондера. Исходные данные: те же, что в базовом протоколе.

1. $A \rightarrow B$: g_A

2. $B \rightarrow A$: g_B

3. $A \rightarrow B$: $E_{K_e}\left(A, Sig_{SK_A}\{g_A, g_B\}, MAC_{K_a}[A]\right)$

4. $B \rightarrow A$: $E_{K_e}\left(A, Sig_{SK_B}\{g_A, g_B\}, MAC_{K_a}[B]\right)$

Выход протокола: сессионный ключ $K_s = \alpha^{yx} \bmod p$, ключ для выполнения аутентификации K_a, ключ шифрования K_e.

В этом протоколе, в отличие от базового корреспондент *B* задерживает отправку своего идентификатора до четвертого сообщения. До этого (в 3-м сообщении) он проводит идентификацию и аутентификацию корреспондента *A*. Такая перестановка, как несложно показать, обеспечивает конфиденциальность идентификатора ответчика к активной атаке злоумышленника.

Из-за симметрии протокола злоумышленник может применить атаку отражения, т. е. сообщение, посланное от корреспондент *A* к корреспонденту *B* он может перенаправить обратно к корреспонденту *A*. Для защиты от этой атаки корреспондент *A* должен послать $MAC_{K_a}\left[\vec{0}, A\right]$, а корреспондент *B* должен послать $MAC_{K_a}\left[\vec{1}, B\right]$, где $\vec{0}$ или $\vec{1}$ последовательность нулевых или единичных бит. Или же корреспонденты *A* и *B* должны использовать разные ключи аутентификации.

Заметим, что данный протокол требует использования четырех обменов.

Таким образом, мы показали, что протокол *SIGMA* удовлетворяет основным требованиям к протоколам распределения ключей на основе использования ассиметричной криптографии. Требования конфиденциальности, имитозащищенности, взаимной согласованности обеспечиваются как для пассивного, так и активного злоумышленника. Требование анонимности выполняется при пассивном злоумышленнике как для инициатора, так и для ответчика. Если злоумышленник активный, то обеспечивается анонимность либо инициатора протокола, либо ответчика.

2. УПРАВЛЕНИЕ ОТКРЫТЫМИ КЛЮЧАМИ

Различные виды криптосистем с открытыми ключами (РША, Рабина, Диффи–Хеллмана, Эль-Гамаля, Мак-Элис, на эллиптических кривых) были рассмотрены во второй части книги. Их общая черта заключается в том, что каждый пользователь имеет, по крайней мере, одну пару ключей:

- закрытый или секретный ключ, который обозначим в данной части, как *SK* (secret key);
- открытый ключ, который обозначим, как *PK* (public key).

Рассмотрим вопросы, связанные с управлением этими ключами.

Управление закрытыми ключами в силу их секретности, может осуществляться так же, как это было рассмотрено выше применительно к ключам симметричных криптосистем. Более того, поскольку закрытые ключи являются, фактически личными ключами корреспондентов, то нет

необходимости их передачи другим участникам криптопротокола. Поэтому задача управления закрытыми ключами существенно упрощается.

Рассмотрим вопросы управления открытыми ключами.

К открытым ключам предъявляются такие требования:

1. Параметры открытого ключа определяются параметрами закрытого ключа, т. е. $PK = f(SK)$, где $f(\cdot)$ – однонаправленная функция, отображающая пространство закрытых ключей в пространство открытых ключей. Длина отрытого ключа определяется выбором параметров криптосистемы с открытыми ключами в целом.

2. На всех этапах жизненного цикла ключа должна обеспечиваться целостность открытого ключа.

3. Должна быть гарантия принадлежности открытого ключа владельцу закрытого ключа.

Естественно, что требования к конфиденциальности открытого ключа не предъявляются.

Этапы жизненного цикла открытого ключа во многом определяются принятой системой распределения этих ключей между корреспондентами.

Наибольшее распространение получили следующие способы:

- обмен открытыми ключами по каналу связи с использованием дополнительного канала проверки подлинности ключа;
- использование третьей (четвертой и т. д.) доверенной стороны для подтверждения подлинности ключа («паутина доверия»);
- использование сертификатов открытых ключей.

Рассмотрим эти способы.

В небольших сетях, корреспонденты, используя соответствующее программное обеспечение, могут самостоятельно сгенерировать пары (*SK*, *PK*) ключей. Для обеспечения криптографической связности они обмениваются друг с другом открытыми ключами. Далее, используя известную им хэш-функцию, корреспонденты находят хэш-код $h(PK)$. Затем, используя другой канал связи, например, телефон, электронную почту, корреспонденты проверяют совпадение хэш-кодов отправленного и принятого открытого ключа. Совпадение хэш-кодов будет свидетельствовать об аутентичности ключей, а узнавание голоса корреспондента будет гарантировать принадлежность открытого ключа его владельцу. Применение хэш-функции здесь необходимо для уменьшения длины сообщения, передаваемого по каналу проверки.

Другой способ распределения открытых ключей основывается на использовании доверенных лиц, которыми являются сами корреспонденты сети. Такая система распределения получила название *паутина на доверии (Web on Trust)*. Например, она применяется в криптографической системе защищенного обмена электронными сообщениями PGP (Pretty Good Privacy).

Рассмотрим способ распределения ключей в этой системе (рис. 2.1).

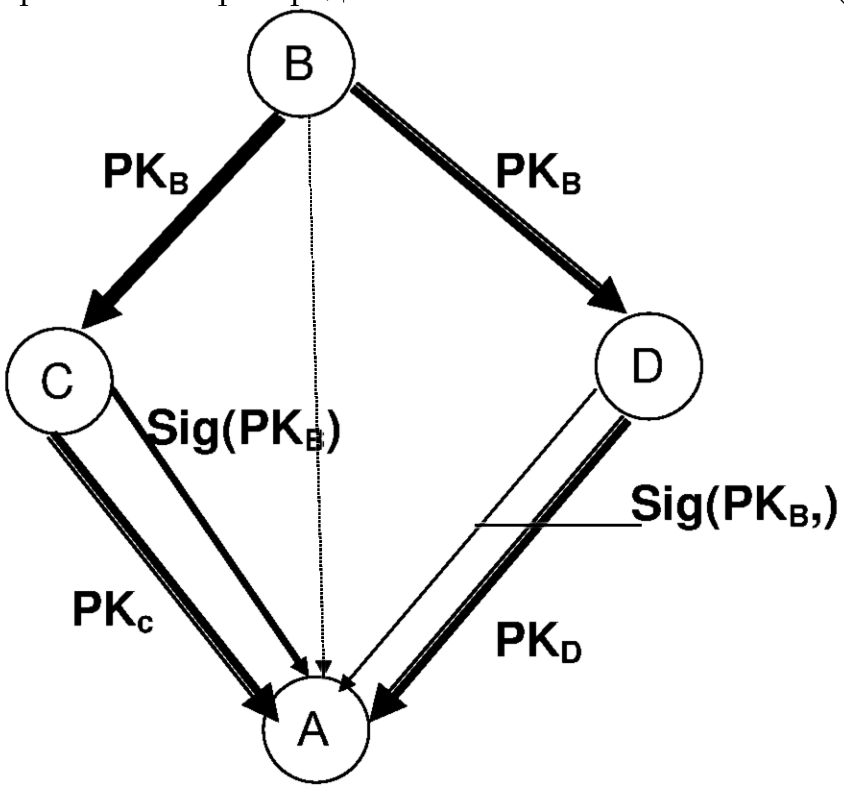


Рис. 2.1. Распределение ключей с помощью доверителей

Корреспонденты сети A, B, C, … создают пары ключей (SK_A, PK_A), (SK_B, PK_B) и т. д. Предположим, что в результате предварительного обмена корреспондент A имеет открытый ключ корреспондента C PK_C и доверяет ему. Корреспондент C имеет открытый ключ корреспондента B PK_B и доверяет ему (это показано на рис. 2.1 жирными линиями). Тогда, если корреспонденту A необходимо взаимодействовать с корреспондентом B, он просит корреспондента C выслать ему ключ корреспондента B и подтвердить свое доверие к этому ключу. Для этого корреспондент C подписывает своим закрытым ключом открытый ключ корреспондента B $Sig_{SK_C}(PK_B)$, тем самым он выражает свое доверие к этому ключу. Корреспондент A проверяет подпись, используя PK_C, и если подпись верна, также доверяет этому ключу. Далее, используя этот ключ, он может послать зашифрованное сообщение корреспонденту B.

Если корреспондент A хочет иметь более убедительные доказательства подлинности PK_B, он может попросить подписать этот ключ другого корреспондента, например D, которому он также доверяет, и имеет его открытый ключ. Положительная верификация двух подписей открытого ключа PK_B дает большую уверенность корреспонденту A в подлинности ключа PK_B.

Далее корреспондент *A* может сам выступить в качестве доверителя открытого ключа корреспондента *B*.

Теоретически большое количество подписей под ключем пользователя должно вызывать большее доверие к этому ключу. Так создается в сети «паутина» взаимного доверия ключам корреспондентов.

Рассмотрим далее другой способ распределения ключей на основе инфраструктуры сертификации открытого ключа (Public Key Certification Infrastructure – *PKI*).

Инфраструктура открытых ключей представляет собой совокупность логически связанных программ, политик, процедур, протоколов, форматов данных, с помощью которых обеспечивается управление открытыми ключами на всех этапах их жизненного цикла. Принцип функционирования *PKI* состоит в следующем.

Рассмотрим инфокоммуникационную сеть, состоящую из большого количества корреспондентов (абонентов), которые хотят осуществлять защищенный обмен сообщениями, например, используя *VPN* технологию.

В сети создается специальный орган – удостоверяющий центр (УЦ), которому доверяют все корреспонденты. УЦ генерирует пару ключей $\left(SK_{\text{УЦ}}, PK_{\text{УЦ}}\right)$ и сообщает всем корреспондентам свой открытый ключ. Подлинность $PK_{\text{УЦ}}$ обеспечивается за счет массовости его рассылки, периодического обновления и общедоступности. Права, обязанности, порядок аккредитации УЦ в РФ устанавливаются Федеральным законом РФ от 6 апреля 2011 г. № 63-ФЗ «Об электронной подписи».

Корреспонденты сети проходят регистрацию в УЦ и на их открытые ключи оформляются сертификаты. Рассмотрим порядок создания сертификата открытого ключа корреспондента *A*. Имея соответствующие программно- аппаратные средства корреспондент *A* генерирует пару ключей $\left(SK_A, PK_A\right)$. Свой открытый ключ PK_A он представляет в УЦ вместе с другими документами (паспорт, письмо-ходатайство от организации, другие данные). УЦ проверяет подлинность представленных документов, проверяет соответствие PK_A и SK_A (для этого он предлагает корреспонденту *A* расшифровать закрытым ключем сообщение защифрованное на его открытом ключе). Далее на представленный корреспондентом *A* открытый ключ оформляется сертификат $Cert\left(PK_A\right)$.

Сертификат это специальная структура данных, содержащая открытый ключ корреспондента, служебную информацию об именах корреспондента и УЦ, назначении и параметрах открытого ключа, электронную цифровую подпись УЦ открытого ключа и всей служебной информации и служащая для хранения, распространения и передачи по каналам связи открытого ключа корреспондента без опасения его несанкционированного изменения.

Сертификат корреспондента A помещается в специальное хранилище, к которому имеют доступ другие корреспонденты сети, согласно установленному УЦ регламенту.

Если, например, корреспондент B намерен послать зашифрованное сообщение корреспонденту A, он обращается с запросом в хранилище сертификатов, получает сертификат $Cert(PK_A)$, проверяет его подлинность путем верификации подписи УЦ на нем. Дополнительно проверяется не был ли этот сертификат аннулирован. Далее, используя PK_A из сертификата, корреспондент B шифрует на этом ключе сообщение и посылает корреспонденту A. Корреспондент A, имея закрытый ключ SK_A, расшифровывает, полученную криптограмму.

В настоящее время предложено несколько моделей *PKI* [19]. Рассмотрим наиболее распространенную модель *PKIX*, использующую сертификаты согласно стандарту ITU X.509.

Общая структура *PKIX*, показана на рис. 2.2. Основными ее элементами являются:

- конечные субъекты (пользователи);
- удостоверяющий центр;
- центр регистрации;
- хранилище сертификатов;
- цифровые сертификаты;
- списки аннулированных сертификатов;
- архив сертификатов.

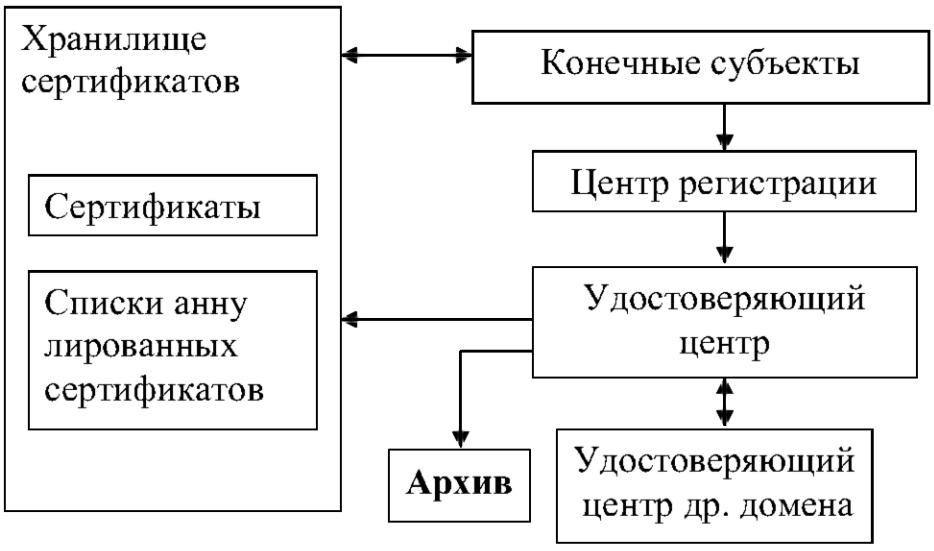


Рис. 2.2. Структура *PKI*

Конечные субъекты определяются в системе как пользователи, которые используют сервисы или функции *PKI*. Субъекты подразделяются на владельцев сертификатов и пользователей сертификатов.

Владельцы сертификатов используют свои закрытые ключи для ЦП и расшифрования данных, зашифрованных на их собственном открытом ключе. Пользователи сертификатов (люди, устройства, приложения), используют открытые ключи владельцев для проверки ЦП, шифрования данных, формирования секретного ключа в процессе согласования ключей.

Удостоверяющий центр – основной элемент *PKI* выпускает в обращение сертификаты и управляет ими. УЦ может выпускать следующие сертификаты: сертификаты пользователей, сертификаты УЦ (самоподписанные сертификаты), сертификаты подчиненных УЦ, кросс-сертификаты, необходимые для взаимного признания сертификатов конечных субъектов других доменов.

УЦ включает свое имя, даты начала и конца действия открытого ключа в сертификат. В случае компрометации закрытого ключа пользователя, ключ выводится из действия, а информация об этом факте публикуется в специальной структуре данных – *списке аннулированных сертификатов* (CAC).

Центр регистрации (РЦ) – создается при большом количестве пользователей. Его основное назначение – регистрация пользователей, проверка предоставленных пользователем данных, которые заносятся в сертификат. Обобщенные данные о пользователе РЦ передает в УЦ для выпуска сертификата.

Хранилище (реестр) сертификатов – это база данных, где хранятся действующие сертификаты и списки аннулированных сертификатов.

К хранилищу предъявляются следующие требования:

- простота и стандартность доступа;
- регулярность обновления информации;
- простота управления;
- встроенная защищенность.

Рекомендуется хранилище строить по директориальному принципу. Целесообразно размещение реестра на сервере каталогов с обеспечением доступа к нему на основе протокола LDAP v.2 , главное достоинство которого быстрое обслуживание запросов.

Цифровые сертификаты – это сертификаты открытых ключей субъектов *PKI*, соответствующие стандарту ITU X.509.v3 (рис. 2.3).

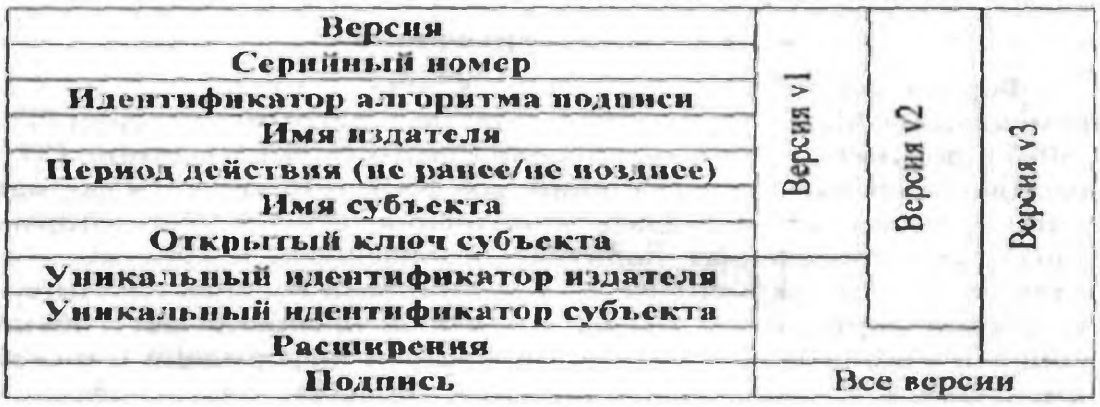


Рис. 2.3. Формат сертификата открытого ключа по стандарту ITU X.509.v3

В сертификате семь обязательных и три служебных поля.

К обязательным полям относятся:

- серийный номер сертификата;
- идентификатор алгоритма подписи;
- период действия;
- имя субъекта сертификата;
- открытый ключ субъекта;
- подпись удостоверяющего центра.

Необязательные поля:

- два уникальных идентификатора;
- дополнения.

Дополнения несут важную информацию, их содержание определено в рекомендациях ITU X.509.v3. Наиболее важные из них:

- key usage (отражает область применения закрытого ключа, соответствующего указанному в сертификате открытому ключу). Например, это формирование и проверка ЦП, шифрование других ключей, шифрование и расшифровка данных, формирование ЦП сертификатов, списков CAC и пр.);

- CRL Distribution Point – указывает местоположение списка аннулированных сертификатов;

- Subject Alternative Name – альтернативные имена владельца сертификата: DNS-имя, IP-адрес, URL-адрес, адрес электронной почты.

Списки аннулированных сертификатов – средство уведомления субъектов PKI об аннулировании цифровых сертификатов. Согласно стандарту ITU X.509.v2 список представляет структурированную двоичную запись и содержит серийные номера аннулированных сертификатов, метки времени, соответствующие моментам их аннулирования, причины аннулирования, ЦП удостоверяющего центра и расширения.

Любой субъект может проверить сертификат путем сверки его номера с серийными номерами, аннулированных сертификатов в CRL.

Архив сертификатов. Предназначен для долговременного хранения от имени УЦ и защиты информации обо всех изданных сертификатах. Данные из архива могут быть востребованы при урегулировании споров по поводу надежности ЦП, которыми заверялись документы.

3. УПРАВЛЕНИЕ КЛЮЧАМИ В СТАНДАРТНЫХ СЕТЕВЫХ ПРОТОКОЛАХ

В настоящее время широкое распространение получили виртуальные защищенные сети (Virtual privatenet work – VPN). Понятие «виртуальная защищенная сеть» является обобщенным понятием, которое описывает сочетание различных технологий, позволяющих объединять отдельные доверенные компьютеры, узлы, сети в единую защищенную сеть, посредством сетей и каналов, к которым нет доверия. Обеспечение защиты информации

в VPN основывается на использовании криптографической защиты в сочетании с туннелированием. Для построения VPN сетей (каналов) разработаны стандартные сетевые протоколы: IPSec, SSL/TLS, PPTP, L2TP и др. [20]. На основе материалов, представленных в предыдущих параграфах, проанализируем, как в этих протоколах решаются вопросы управления ключами.

3.1. Протокол IPSec/IKE

Протокол IPSec работает на сетевом уровне модели OSI и позволяет решить такие задачи защиты информации:

- аутентификация конечных точек (хостов, маршрутизаторов);
- автоматическое снабжение конечных точек ключами;
- обеспечение конфиденциальности передаваемых сообщений;
- обеспечение целостности передаваемых данных и служебной информации.

Архитектуру IPSec составляют три протокола:

IKE (Internet Key Exchange) – протокол согласования параметров и обмена ключами);

AH (Authentication Header) – протокол аутентифицирующего заголовка;

ESP (Encapculating Security Payload) – протокол защиты содержимого.

Рассмотрим далее протокол IKE v.2 (RFC), который функционирует на начальном этапе установления соединения между двумя узлами и выполняет следующие задачи:

- согласование параметров безопасного соединения между двумя узлами;
- аутентификацию сторон;
- выработку ключей для безопасного соединения и протоколов AH и ESP.

Взаимодействие сторон по протоколу IKE осуществляется путем проведения двух начальных обменов: обмена по установлению параметров соединения – IKE_SA_UNIT и обмена аутентификации сторон – IKE_AUTH, в ходе которых между взаимодействующими сторонами устанавливаются безопасное соединение и вырабатываются сеансовые ключи.

Стороны, участвующие в обменах, будем обозначать как *инициатор (i – initiator)* и *ответчик (r – responder)*. Каждый обмен содержит две пары сообщений: *запрос*, посылаемый от инициатора к ответчику и *ответ*, посылаемый от ответчика к инициатору. Каждое сообщение содержит заголовок и несколько блоков данных.

Рассмотрим сначала назначение основных блоков, передаваемых в сообщениях обмена.

Блок данных SA (безопасное соединение), содержит в себе такие параметры: вид протокола (AH, ESP, IKE); тип преобразования: шифрование, псевдослучайная функция, проверка целостности, тип группы Диффи–Хеллмана; алгоритм криптопребразо-

вания; атрибуты преобразования (например, размер ключа). Алгоритмы для разных типов криптопреобразований приведены в табл. 3.1

Блок данных $KE_i\,(KE_r)$ используется для обмена значениями Диффи–Хеллмана вида $g = \alpha^x \bmod p$.

Таблица 3.1

Типы и алгоритмы криптопреобразований в протоколе IPSec

Тип преобразования	Алгоритмы преобразования
Шифрование (ENCR)	DES, DES IV32, DES IV128, 3DES, IDEA, RC5, CAST, Blowfish, 3IDEA, AES CBC, AES CTR
Псевдослучайная функция (PRF)	HMAC MD5, HMAC SHA1, HMAC TIGER, AES 128 CBC
Проверка целостности (INTEG)	HMAC MD5 96, HMAC SHA1 96, DES MAC, KPDK MD5, AES XCBC 96
Тип группы Диффи–Хеллмана (DH)	Указывается номер группы

Блок данных $ID_i\,(ID_r)$ – позволяет корреспондентам объявить друг другу свои идентификаторы, например, IP-адрес, адрес электронной почты, доменное имя и проч.).

Блок данных (CERT) – обеспечивает транспортировку сертификатов или другой информации, связанной с сертификатами.

Блок данных (CERTREQ) – запрос на получение сертификата.

Блок данных (AUTH) – содержит данные, используемые для аутентификации. Эти данные формируются на основе ЭЦП RSA, ЭЦП DSS, кода целостности.

Блок данных $N_i,(N_r)$ – содержит случайные числа, генерируемые передающей стороной в диапазоне от 16 до 256 октетов, которые используются для обеспечения свежести переданных данных.

Рассмотрим далее процедуры выработки ключей для создаваемого безопасного соединения в обменах согласования параметров и аутентификации сторон. Решение этих задач в протоколе IKE может быть выполнено двумя способами:

- на основе имеющихся у корреспондентов сертификатов открытых ключей;
- на основе предварительно распределенного по доверенному каналу секретного ключа.

Рассмотрим сначала первый способ.

В ходе первого обмена инициатор и ответчик посылают друг другу блоки:

1. $i \rightarrow r \quad SA_{i1},\ KE_{i1},\ N_i$
2. $r \rightarrow i \quad SA_{r1},\ KE_{ir1},\ N_r,$

где в блоке данных SA_{i1} инициатор предлагает криптографические алгоритмы, режимы их применения и другие параметры, в блоке SA_{r1} ответчик указывает только те параметры, которые он будет поддерживать далее в

данной безопасной ассоциации, KE_{i1} и KE_{ir1} это значения Диффи–Хеллмана $g_i = \alpha^{x_i} \bmod p$, $g_r = \alpha^{x_r} \bmod p$ соответственно, N_i, N_r – случайные числа инициатора и ответчика.

На основании значений Диффи–Хеллмана каждая из сторон генерирует порождающий ключ SKEYSEED, используя преобразование:

$$SKEYSEED = prf\left(N_i \mid N_r, \alpha^{x_i x_r}\right),$$

где *prf*() – псевдослучайная функция, $N_i \mid N_r$ строка бит, полученная конкатенацией случайных чисел N_i, N_r.

Далее из порождающего ключа формируется семь ключей:

K_d – ключ, используемый для выработки ключей в дочерней безопасной ассоциации;

K_{a_i}, K_{a_r} – ключи, используемые для аутентификации последующих обменов;

K_{e_i}, K_{e_r} – ключи, используемые для шифрования;

K_{p_i}, K_{p_r} – ключи, используемые для генерации блоков данных AUTH.

Эти ключи вычисляются путем применения псевдослучайной функции $prf + \left(K_d, K_{a_i}, K_{a_r}, K_{e_i}, K_{e_r}, K_{p_i}, K_{p_r}\right) = prf + \left(SKEYSEED, N_i \mid N_r, SP_i, SP_r\right)$, где SP_i, SP_r – индекс параметров защиты – 32-разрядная метка, которая получается путем хэширования блоков SA_{i1}, KE_{i1}, N_i и SA_{r1}, KE_{ir1}, N_r соответственно.

Псевдослучайная функция *prf*+, определяется следующим образом

$$prf + \left(K, S\right) = T1 \mid T2 \mid T3 \ldots,$$

где $T1 = prf\left(K, S \mid 0 \times 01\right)$, $T2 = prf\left(K, T1 \mid 0 \times 02\right)$, $T3 = prf\left(K, T2 \mid 0 \times 03\right)\ldots$

Необходимые ключи берутся из выходной строки путем выбора требуемого количества бит, не принимая во внимание границы блоков Ti.

Например. Если требуемыми ключами являются 256-битный ключ AES и 160-битный ключ HMAC, а функция *prf* генерирует 160-битные блоки, то ключ AES будет выбираться из $T1$ и начала $T2$, в то время как ключ HMAC будет выбираться из остатка $T2$ и начала $T3$.

Отметим, что сформированный сторонами порождающий ключ SKEYSEED и все производные от него ключи пока не являются защищенными от атаки «человек посередине». Также заметим, что в первоначальном обмене IKE_SA_UNIT стороны не показали свои идентификаторы, чем обеспечили анонимность обмена.

Второй обмен сообщениями между инициатором и ответчиком выполняется путем обмена блоками, в соответствии с протоколом SIGMA-R, описанном в п. 2.

3. $i \rightarrow r \quad E_{K_e}\left(ID_i, [CERT], [CERTREQ], ID_r, AUTH_i, SA_{i2}, TS_i, TS_r\right)$;

4. $r \rightarrow i \quad E_{K_e}\left(ID_r, [CERT], AUTH_r, SA_{r2}, TS_i, TS_r\right)$.

Блоки данных, передаваемые в обоих сообщениях, шифруются на ключах K_{e_i}, K_{e_r} соответственно, блоки *CERT* содержат открытые ключи, необходимые для верификации блоков *AUTH*. Инициатор в первом сообщении передает два селектора трафика TS_i, TS_r и передает также блок SA_{r2}, чем обеспечивает согласование параметров следующего безопасного соединения (по протоколу ESP или AH). Ответчик в своем сообщении объявляет свой идентификатор ID_r, представляет согласованные параметры безопасного соединения в блоке SA_{r2}, а блоком *AUTH* подтверждает аутентичность второго сообщения.

Наиболее важными в данном обмене являются блоки *AUTH*, которыми обеспечивается выполнение требований безопасного формирования ключа обеими корреспондентами, Эти блоки имеют следующую структуру

$$AUTH_i = Sig_i\{SA_{i1}, KE_i, N_i, N_r, prf(K_{pi}, ID_i)\}$$

$$AUTH_r = Sig_r\{SA_{r1}, KE_r, N_r, N_i, prf(K_{pr}, ID_r)\}$$

Видим, что блок *AUTH* содержит ЦП от сообщения, состоящего из блоков, посланных в начальном обмене, к которым присоединен одноразовый номер противоположной стороны и значение псевдослучайной функции (MAC) от блоков идентификации (С. 254).

В данном протоколе в отличие от классического варианта протокола SIGMA-R блок $\text{MAC} = prf\left(K_p, ID\right)$ внесен в состав подписываемого сообщения, что позволяет уменьшить длину блока *AUTH*.

Второй способ выработки сессионного ключа (на основе предварительно распределенного по доверенному каналу секретного ключа) отличается от рассмотренного выше тем, что в первом и втором обменах блоки CERT и CERTREQ не используются. А блок аутентификации формируется без ЦП следующим образом

$$AUTH = prf(prf(Shared\ Secret, "KeyPadforIKEv.2")),$$

где строка "*KeyPadforIKEv*.2" представляет собой 17 символов кода ASCII. Распределенный ключ между корреспондентами может быть строкой переменной длины. Для обеспечения стойкости эта строка должна быть случайной, а ее длина должна быть не меньше длины формируемого ключа.

После успешного выполнения второго обмена создается безопасное соединение, называемое CHILD SA, в рамках которой стороны могут осуществить обмен данными по протоколу ESP или AH.

Ключи для криптографических преобразований в этом безопасном соединении генерируются следующим образом

$$KeyMAT = prf + \left(K_d, N_i, N_r\right).$$

Из полученной цепочки бит первые октеты составляют ключ шифрования, а следующие октеты составляют ключ аутентификации.

3.2. Протокол SSL/TLS

Протокол SSL/TLS обычно применяется для обеспечения безопасности передачи данных в WEB соединениях. Данный протокол располагается над транспортным протоколом и по иерархии семиуровневой модели OSI относится к протоколу сеансового уровня. Имеются варианты практической реализации, когда протокол выполняется на уровне приложений, например, в рамках протокола передачи гипертекстовых файлов (HTTP), облегченного протокола служебных каталогов (LDAP), протокола доступа к сообщениям сети (IMAP).

Протокол обеспечивает:

- аутентификацию сервера, гарантируя, что пользователь попал на тот Web-узел, который хотел посетить;
- аутентификацию клиента в соответствии с заявленным им именем;
- создание защищенного канала, в котором информация передается между клиентом и сервером в зашифрованном и аутентифицированном виде.

Протокол имеет двухуровневую структуру, включающую два протокола: протокол записи и протокол рукопожатия,

Протокол записи разделяет сообщения на блоки, сжимает их (при необходимости), аутентифицирует каждый блок, вычисляя аутентификатор (HMAC), шифрует блоки с помощью симметричного алгоритма и передает корреспонденту. Адресат получает зашифрованные блоки данных, расшифровывает, проверяет аутентичность, разархивирует блоки (если они подвергались сжатию) и собирает блоки в целое сообщение.

Протокол квитирования предшествует протоколу записи. В ходе его выполнения на клиентской машине и сервере устанавливается сессия, в ходе которой:

- согласуются алгоритмы шифрования, аутентификации, сжатия;
- осуществляется аутентификация сторон;
- формируется мастер-ключ и на его основе генерируются сеансовые ключи.

Рассмотрим более подробно протокол квитирования, акцентируя внимание на процедуру выработки ключей.

Протокол включает четыре обмена. В ходе первого обмена Клиент и Сервер посылают друг другу сообщения Client Hello и Server Hello, включающие блоки:

$C \rightarrow S$: Client Hello $\left(N_c, ID_{session}, CipherSuite_c\right)$,

$S \rightarrow C$: Server Hello $\left(N_s, ID_{session}, CipherSuite_s\right)$,

где N_c, N_s – случайные числа, $ID_{session}$ – идентификатор сессии, блок *CipherSuite* – комплект шифров.

Блок комплект шифров, посылаемый клиентом, содержит набор криптографических опций, определяющих способ формирования ключей и параметры криптографического преобразования (алгоритм шифрования, алгоритм вычисления MAC, длину хэш-кода, вектор инициализации и др.). Из предложенного набора сервер выбирает единственную схему обмена ключами и криптографического преобразования и сообщает о ней клиенту в сообщении Server Hello.

Содержание сообщений, передаваемых во 2-м и 3-м обменах, существенно зависит от согласованного в первом обмене метода формирования ключей. В протоколе SSL/TLS поддерживаются следующие способы формирования ключей (рис. 3.2).

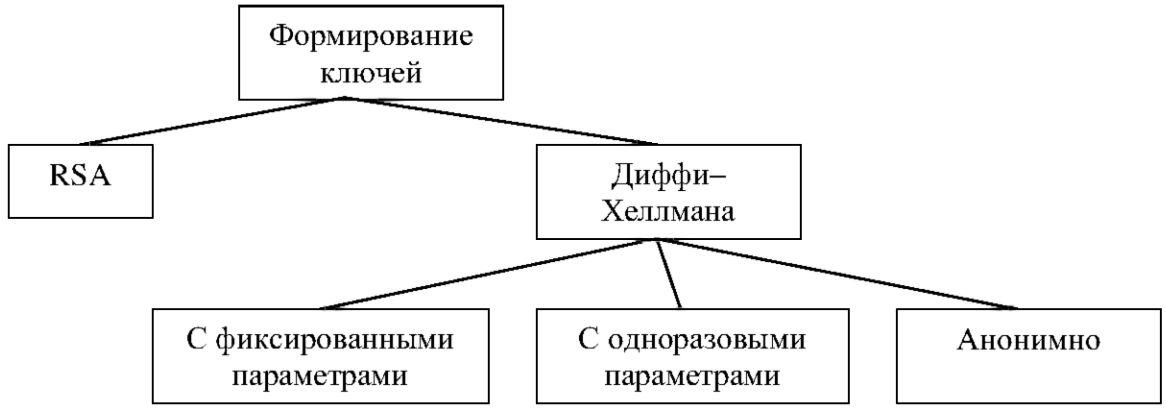


Рис. 3.2. Способы формирования ключей в протоколе SSL/TLS

При использовании способа RSA секретный ключ (строка бит) формируется пользователем, шифруется открытым ключом сервера, согласно алгоритму RSA и передается серверу. Сервер расшифровывает криптограмму своим закрытым ключом и выделяет строку бит, посланную клиентом.

При использовании способа Диффи–Хеллмана возможны три варианта.

Для способа с фиксированными параметрами получателю передается тройка чисел (p, α, α^y), где p – модуль (простое число); α – порождающий элемент большой группы по модулю p; α^y – открытый ключ отправителя. Эта тройка чисел включается в сертификат, подписанный удостоверяющим центром.

Для способа с одноразовыми параметрами и анонимного в каждом сеансе передается тройка чисел (p, α, α^y), где y – случайное число для каждого сеанса. В случае одноразовых параметров осуществляется аутентификация передаваемого сообщения.

Второй обмен содержит четыре сообщения, передаваемые от сервера клиенту:

S→C: *Server Certificate*
S→C: *Server Key Exchange*
S→C: *Certificate Request*
S→C: *Server Hello Done*

Сообщение *Certificate(Сертификат)* используется для подтверждения подлинности открытого ключа сервера. Для способа обмена ключами DH с фиксированными па-

раметрами это сообщение содержит значение $\alpha^y \bmod p$, включенное в сертификат. Для анонимного способа это сообщение не передается.

Сообщение *Server Key Exchange* – используется для обмена ключами.

Для способа RSA, когда сервер имеет ключ RSA, предназначенный только для подписи, прямой обмен ключами не возможен, В этом случае сервер генерирует временную пару ключей (SK'_{serv}, PK'_{serv}). После этого открытый ключ PK'_{serv} подписывается закрытым ключем сервера $Sig_{serv}\{PK'_{serv}\}$ и включается в сообщение *Server Key Exchange.*

Для способа DH с фиксированными параметрами это сообщение не передается, так как ключ уже передан в сертификате.

Для способа DH с одноразовыми параметрами сообщение содержит подпись $Sig_{serv}\{p, \alpha, \alpha^y\}$ тройки чисел (p, α, α^y). Для анонимного способа DH тройка чисел (p, α, α^y) передается без подписи.

В сообщении *Certificate Request* сервер запрашивает сертификат клиента.

Сообщение *Server Hello Done* означает переход сервера в режим ожидания.

Заметим, что верификация подписей на сертификатах и подписанных сервером сообщениях обеспечивают аутентификацию сервера.

В третьем обмене клиент посылает серверу три сообщения:

C→S: *Certificate*

C→S: *Client Key Exchange*

C→S: *Certificate verify*

Если сервер запросил сертификат, то ему отправляется сообщение *Certificate.* Если у клиента сертификата нет, передается уведомление *No Certificate.*

Содержание сообщения *Client Key Exchange* зависит от выбранного способа обмена ключами и может быть следующим:

- для способа RSA клиент генерирует 48-байтовый предварительный ключ, шифрует его с помощью открытого ключа сервера, взятого из сертификата или с помощью временного открытого ключа PK'_{serv}, полученного в сообщении *Server Key Exchange*;

- для способа DH с одноразовыми параметрами или анонимного способа передается тройка чисел (p, α, α^x), где x – секретный ключ клиента;

- для способа DH с фиксированными параметрами это сообщение пустое, так как значения Диффи–Хеллмана были переданы в сертификате.

Сообщение *Certificate verify* подтверждает цифровой подписью подлинность данных в предыдущем сообщении.

В табл. 3.2 показаны сообщения, передаваемые во 2-м и 3-м обменах для разных способов формирования ключа.

В ходе четвертого обмена клиент и сервер передают друг другу по два сообщения:

C→S: *Change cipher spec*

C→S: *Finished*

S→C: *Change cipher spec*

S→C: Finished

Таблица 3.2

Сообщения, передаваемые во 2-м и 3-м обменах
для разных способов формирования ключа

	Способ обмена ключами			
Передаваемое сообщение	RSA	Диффи-Хеллмана		
		Фиксиров параметры	Одноразовые параметры	Анонимно
Certificate	$Cert(PK_{serv})$	$Cert(\alpha^y)$	$Cert(PK_{serv})$	не передается
Server key exchange	$Sig_{serv}\{PK'_{serv}\}$	не передается	$Sig_{serv}\{p, \alpha, \alpha^y\}$	(p, α, α^x)
Certificate	$Cert(PK_{client})$	$Cert(\alpha^y)$	$Cert(PK_{client})$	не передается
Client key exchange	$E_{PK'_{serv}}(premasterkey)$	не передается	(p, α, α^x)	(p, α, α^x)
Certificate verify	не передается	не передается	$Sig_{serv}\{p, \alpha, \alpha^y\}$	не передается

Сообщения *Change cipher spec* означают переход к работе протокола Записи, в котором будут использоваться согласованные в ходе протокола квитирования параметры. Сообщения *Finished* являются тестовыми сообщениями для проверки работоспособности установленных криптографических алгоритмов с согласованными параметрами. Сообщение *Finished* представляет конкатенацию двух значений HMAC кода, вычисленного на основе мастер ключа с использованием хэш-функций MD5, SHA-1.

$$MD5(master\ \sec ret \,|\, Pad2 \,|\, MD5(handShake\ massage \,|\, sender \,|\, master\ \sec ret \,|\, Pad1),$$

$$SHA(master\ \sec ret \,|\, Pad2 \,|\, SHA(handShake\ massage \,|\, sender \,|\, master\ \sec ret \,|\, Pad1),$$

где $Pad1$ и $Pad2$ – заполнители, *handshake massage* включает в себя все сообщения, переданные ранее, *sender* – код отправителя сообщения (клиент или сервер). Совпадение HMAC кодов подтверждает, что процессы обмена ключами завершились успешно. Далее клиент и сервер могут переходить к обмену данными на уровне приложения.

Вычисление ключей шифрования, аутентификации и хэш-функции осуществляется на основе главного ключа, который в свою очередь находится из предварительного ключа следующим образом.

Если для обмена ключами использовался способ RSA, то предварительный ключ – строка бит 48 байт = 384 бита, сгенерированная клиентом, переданная в зашифрованном виде серверу.

Если для обмена ключами использовался способ DH, то предварительный ключ вычисляется по формуле $pre\_master\_key = \alpha^{xy} \bmod p$.

Главный ключ находится из предварительного ключа с помощью двух хэш-функций MD5 и SHA-1

$$master\_\sec ret = MD5(pmk \mid SHA(A \mid pmk \mid R_c \mid R_s \mid MD5(pmk \mid SHA(BB \mid pmk \mid R_c \mid R_s)) \mid$$
$$MD5(pmk \mid SHA(CC),$$

где R_c, R_s – случайные числа в сообщениях Client Hello и Server Hello соответственно, $pmk = pre\_master\_key$.

Ключ шифрования, аутентификации и хэш-функции вычисляются по вышеприведенной формуле, в которой следуем заменить pmk на $mk = master\_\sec ret$. Процедура выполняется до тех пор, пока не будет сгенерирована последовательность достаточной длины для получения всех ключей.

Проводя анализ протокола SSL/TLS с позиции выполнения требований по обеспечению безопасности распределения ключей (п. 3) можно сделать такие выводы. Конфиденциальность формирования ключей обеспечивается, поскольку для способа RSA предварительный ключ передается в зашифрованном виде. Для способа DH конфиденциальность выполняется при передаче значений Диффи–Хеллмана в составе сертификата или в подписанном виде. При этом устраняется атака «человек посредине» и обеспечивается имитозащищенность передаваемых сообщений. Для анонимного способа ни конфиденциальность, ни имитозащищенность не обеспечиваются. Вместе с тем можно заметить, что требования обеспечения взаимной связности сторон, формирующих ключ, и их анонимности не выполняются. Также для способа RSA, существует опасность имитационной атаки на зашифрованный ключ при его передаче от клиента к серверу.

3.3. Формирование ключа в протоколах PPTP, L2TP

Протоколы PPTP (Point to point protocol), L2TP (Layer 2 tunneling protocol) предназначены для построения защищенных VPN туннелей, обеспечивающих подключение удаленных пользователей к своим сетям.

Протокол PPTP позволяет создавать защищенные каналы для обмена данными по различным сетевым протоколам (IP, IPX, NetBeU, Apple Talk и др.). Пакеты с данными этих протоколов инкапсулируются внутрь пакета TCP/IP и переносятся на другой конец линии в зашифрованном виде.

Шифрование осуществляется по алгоритмам DES или MMPE. Основой алгоритма DES является блочный шифр, а в основе алгоритма MMPE лежит шифр гаммирования RC4.

Выбор алгоритма шифрования, длина ключа и другие параметры устанавливаются между взаимодействующими сторонами при обмене управляющими сообщениями. После установки режима обмена начинается сеанс по передаче пакетов зашифрованных данных. Только те пакеты, номера протоколов которых лежат в диапазоне 0×0021–0×00fa, под-

лежат шифрованию. Аутентификация передаваемых пакетов не обеспечивается.

Ключи для выполнения шифрования формируются из паролей, заранее распределенных между удаленными пользователями и сервером удаленного доступа сети пользователя.

Протокол MMPE(Microsoft Point to Point Encryption) поддерживает работу с ключами 40 и 128 бит.

40-битный ключ формируется следующим образом:

- из пароля генерируется 64-битное значение хэш-функции Lan Manager. Полученная строка еще раз преобразуется с помощью хэш-функции SHA;

- старшие 24 бита устанавливаются в значение 0×01269e.

128-битный ключ формируется следующим образом:

- объединяется 64-битовое значение хэш-кода Windows NT и 64-битовое случайное число, выработанное сервером и переданное пользователю в ходе процедуры аутентификации по протоколу MS-CHAP;

- полученная строка бит хэшируется с помощью хэш-функции SHA.

Существует возможность обновления ключа, после каждого переданного пакета, но это снижает эффективность шифрования, поскольку для выработки ключа требуется время.

Протокол L2TP позволяет создавать защищенные каналы для обмена данными разных сетевых протоколов (IP, IPX, NetBeU, Apple Talk и др.) не только по IP, но и по другим сетям, таким как ATM, X.25, FR. Протокол L2TP использует два вида пакетов (управляющие и информационные) одинакового формата. Управляющие сообщения используются при установлении, поддержке и аннулировании туннелей и вызовов. Информационные сообщения используются для инкапсуляции PPP-кадров, пересылаемых по туннелю.

Протокол L2TP не определяет конкретных методов криптозащиты и позволяет использовать разные методы шифрования. Если защищается туннель в IP-сетях, то используется протокол IPSec/ESP. Ключи шифрования и аутентификации создаются во время IKE сессии, на основе предварительно распределенных сертификатов открытых ключей.

Протокол L2TP, использующий для защиты данных протокол IPSec, обеспечивает более высокую степень защиты данных, чем протокол PPTP, так как используются более совершенные алгоритмы шифрования, например, 3DES или AES с большой длиной ключа, а для аутентификации данных используются хэш-функции HMAC-MD5 или HMAC-SHA-1 с большой длиной хэш-кода. Распределение ключей осуществляется на основе достаточно совершенного протокола IKE (п. 3.1).

Список литературы к части III

Основная

1. Шнайер, Б. Прикладная криптография / Б. Шнайер. – М. : Триумф, 2002.

2. Мао, В. Современная криптография / В. Мао. – СПб., Киев : Вильямс, 2005.

3. Menezes, A. J. Handbook of Applied Cryptography / A. J. Menezes. – New-York, London, Tokyo : CRC press, 1997.

4. Boyd, C. Protocols for Authentication key Establishment / C. Boyd, A. Mathuria. 2003.

5. Черемушкин, А. Криптографические протоколы. Основные свойства и уязвимости / А. Черемушкин. 2007.

6. Столингс, В. Основы защиты сетей. Приложения и стандарты / В. Столингс. – М., СПб., Киев : Вильямс, 2002.

7. Смит, Р. Аутентификация: от паролей до открытых ключей / Р. Смит. – М., СПб., Киев : Вильямс, 2002.

8. Иванов, М. Теория, применение и оценка качества генераторов псевдослучайных последовательностей / М. Иванов, И. Чугунков. – М. : КУДИЦ-ОБРАЗ, 2003. – 240 с.

9. Maurer, U. A universal statistical test for random bit generators. Journal of cryptography / U. Maurer, v. 5, 1992, pp. 89–105.

10. Коржик, В. И. Распределение ключей для конфиденциальной связи, основанное на использовании линейных преобразований / В. И. Коржик, Ю. Меринович, И. Слепенков // Информатика и вычислительная техника. – № 3. – 1994. – С. 26–27.

11. Blom, R. Non-Public Key Distribution / R. Blom // Advances in Cryptology. – Proc. Eurocrypt'82. Plenum. N.Y., 1983. pp. 231–236.

12. Needham, R. Using encryption for authentication in large networks of computers / R. Needham, M. Schroeder // Commbnication od the ACM, 21 (12), 1978, pp. 993–999.

13. Otway, D. Efficient and timely mutual authentication. Operating System Review / D. Otway, O. Rees, 21 (1), 1987, pp. 8–10.

14. Lowe, G. An attack on the Needham-Schroeder public-key authentication protocol / G. Lowe // Information Proceeding Letters 56(3), 1995, pp. 133.

15. Diffie, W. New directions in cryptography. IEEE Trans. Information Theory / W. Diffie, M. Hellman, IT-22(6), 1976, pp. 644–654.

16. Diffie, W. Authentication and authenticated key exchange / W. Diffie, van Oorschot P., M. Wiener // Designs, Codes and cryptography, 1992, pp. 107–125.

17. Cannetti, R. Security Analysis of IKE's Signature-Based Key-Exchange Protocol / R. Cannetti, H. Krawczyk. Crypto 2002. LNCS Vol. 2442.

18. Krawczyk, H. SIGMA: The «SIGn and-Mac» Approach to Authenticated Diffie-Hellman and Its Use in the IKE Protocols / H. Krawczyk. Crypto 2003. LNCS Vol. 2729. pp. 400–425.

19. Горбатов, В. Основы технологии PKI / В. Горбатов, О. Полянская. – М. : Горячая линия – Телеком, 2004.

20. Запечников, С. Основы построения виртуальных частных сетей : учебное пособие для вузов / С. Запечников, Н. Милославская, А. Толстой. – М. : Горячая линия – Телеком, 2003.

21. Raz, R. Extracting all the randomness and reducing the error in trevisan's extractors / R. Raz, O. Reingold, S. P. Vadhan. – J. Comput. Syst. Sci., vol. 65, no. 1, pp. 97–128, 2002.

Дополнительная

22. Maurer, U. Secret-key agreement over unauthenticated public channels. Part III. Privacy amplification. IEEE Transactions on Information Theory / U. Maurer, S. Wolf, vol. 49, no. 4, pp. 839–851, 2003.

23. Yakovlev, V. Key Distribution Protocols Based on Noisy Channel in Presence of Active Adversary: Conventional and New Versions with Parameter Optimization. IEEE Transaction on Information Theory / V. Yakovlev, V. Korzhik, G. Morales-Luna. Vol. 54. No 6, June 2008. pp. 2535–2549.

24. Yakovlev, V. Optimization of Key Distribution Protocols Based on Extractors for Noisy Channels within Activ Adversaries. 6 th International Conference of «Mathematical Methods, Models and Architectures for Computer Network Security» / V. Yakovlev, V. Korzhik, M. Bakaev, G. Morales-Luna. LNCS 2012. N. 7531, P. 51–64.

25. Aono, T. Wireless secret key generation exploiting reactance – domain scalar response of multipath fading cannels. IEEE Trans. Antennas Propag / T. Aono, K. Higushi, T. Ohira, B. Komiyam, H. Sasaoka. Vol. 53. pp. 3776–3784. Nov. 2005.

26. Яковлев, В. Распределение ключей в беспроводных локальных сетях на основе использования антенн со случайно изменяемой диаграммой направленности в условиях многолучевого распространения радиоволн. Часть 1. модель канала распределения ключей. Проблемы информационной безопасности. Компьютерные системы / В. Яковлев, В. Коржик, Ю. Ковайкин. – СПб. : СПбГПУ, № 4. – 2010. – С. 51–64.

27. Яковлев, В. Распределение ключей в беспроводных локальных сетях на основе использования антенн со случайно изменяемой диаграммой направленности в условиях многолучевого распространения радиоволн. Часть 2. Система распределения ключей и ее оптимизация. Проблемы информационной безопасности. Компьютерные системы / В. Яковлев, В. Коржик, Ю. Ковайкин ; СПбГУТ. – СПб., № 1. – 2011. – С. 87–99.

28. Yakovlev, V. Secret Key Agreement Over Multipath Channels Exploiting a Variable-Directional Antenna International Journal of Advanced Computer Science and Applications (IJACSA-ISSN 2156 5570) / V. Yakovlev, V. Korzhik, Y. Kovajkin, G. Morales-Luna, January 2012. Volume 3. – No. 1. – P. 172–178.

ЗАКЛЮЧЕНИЕ

Настоящая книга, состоящая из трех частей, содержит материал по «продвинутому» курсу лекций основ криптографии. Такое определение означает, что в нем содержатся разделы, выходящие за рамки подобных стандартных курсов, читаемых для общих специальностей по телекоммуникации и родственных специальностей в технических университетах. Это расширение является особенно необходимым для студентов, специализирующихся в области *информационной безопасности*, когда помимо аксиоматического знания о существовании стойких шифров необходимо некоторое понимание, почему они могут быть сделаны таковыми при определенном выборе их структуры и параметров.

Вместе с тем, целесообразно повторить, уже сказанное ранее во введении, что по настоящему пособию невозможно полностью подготовиться к такой профессиональной деятельности, как разработка новых шифров и протоколов. Тем не менее, студенты хорошо изучившие данный материал, могут успешно решать следующие профессиональные задачи:

- реализовывать совершенные (одноразовые) шифры и понимать ограниченность их применения,
- реализовывать доказуемо стойкие к линейному и дифференциальному криптоанализу алгоритмы блокового шифрования на основе использования ППШ,
- понимать свойства и особенности применения различных мод шифрования,
- понимать стойкость методов многократного шифрования,
- реализовывать доказуемо стойкие к некоторым методам криптоанализа потоковые шифры,
- знать основные побочные атаки на блоковые и потоковые шифры и уметь защититься от них,
- понимать работу основных современных стандартов шифрования,
- реализовывать основные алгоритмы шифрования с открытым ключом при выборе параметров, обеспечивающим стойкость этих шифров к основным и побочным атакам,
- иметь представление о реализации алгоритмов шифрования с открытым ключом над ЭК,
- реализовывать некоторые стойкие алгоритмы цифровой подписи,
- понимать проблематику криптографических протоколов и алгоритмы выполнения некоторых из них,
- понимать задачи управления ключами в криптографических системах,

- понимать основные критерии эффективности распределения ключей с использованием специального центра или взаимного обмена данными между пользователями,

- уметь анализировать стандартные сетевые протоколы в части решения ими задач управления ключами.

Трудно перечислить все направления прикладной криптографии, которые остались за пределами данного пособия, но основными из них являются следующие:

- практические (отличающиеся от «учебных») аспекты информационной безопасности, которые наиболее полно изложены в [19],

- полное описание криптографических протоколов [14, 15, 18],

- строгие доказательства некоторых утверждений из теории чисел и основанные на них свойства криптосистем с открытым ключом, которые можно найти в специализированных монографиях [1, 2, 5, 8, 13, 15, 17, 19, 20].

Трудно перечислить все направления прикладной криптографии, которые остались за пределами данного пособия, но основными из них являются следующие:

- практические (отличающиеся от «учебных») аспекты информационной безопасности, которые наиболее полно изложены в [19],

-полное описание криптографических протоколов [14,15,18],

-строгие доказательства некоторых утверждений из теории чисел и основанные на них свойства криптосистем с открытым ключом, которые можно найти в специализированных монографиях [1,2,5,8,13,15,17,19,20].

Коржик Валерий Иванович
Яковлев Виктор Алексеевич

ОСНОВЫ КРИПТОГРАФИИ

Учебное пособие

Редакторы:
И. И. Щенсняк, С. Д. Щербакова, Л. К. Паршина
Компьютерная верстка ***Е. А. Головинской***

Подписано в печать 11.04.2016. Формат 60 × 88 1 / 16. Усл. печ. л. 12.
Тираж 1000 экз. Первый завод 100 экз. Заказ № __
ООО «Издательский центр “Интермедия”». Адрес: 198334, Санкт-Петербург,
ул. Партизана Германа, 41-218.
Отпечатано с готового оригинал-макета
в ООО «Арт-экспресс».
Адрес: 199155, СПб, В.О., ул. Уральская, д. 17.