ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАНИЕ

**М. Е. ИЛЬИН, Т. И. КАЛИНКИНА,
В. Н. ПРЖЕГОРЛИНСКИЙ**

КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА ИНФОРМАЦИИ В ОБЪЕКТАХ ИНФОРМАЦИОННОЙ ИНФРАСТРУКТУРЫ

Учебник

Под редакцией В. Н. ПРЖЕГОРЛИНСКОГО

Учебник

ACADEMIA

Москва
Издательский центр «Академия»
2019

УДК 004.056.5(075.32)
ББК 32.973.26-018.2я723
И46

Рецензент — ???

Ильин М.Е.

И46 Криптографическая защита информации в объектах информационной инфраструктуры : учеб. для студ. учреждений сред. проф. образования / М.Е.Ильин, Т.И.Калинкина, В.Н.Пржегорлинский. — М. : Издательский центр «Академия», 2019. — с. ???

ISBN 978-5-4468-8717-0

Для обучающихся по специальностям укрупненной группы специальностей среднего профессионального образования 10.00.00 «Информационная безопасность». Может быть использован в дополнительном профессиональном образовании по программам повышения квалификации и переподготовки специалистов, работающих в сфере информационной безопасности, преподавателей, осуществляющих образовательную деятельность по специальностям укрупненной группы специальностей среднего профессионального образования 10.00.00 «Информационная безопасность».

Рассмотрены математические основы криптографии, методы и средства криптографической защиты информации, современные стандарты шифрования, основы криптоанализа. Описаны поточные шифры и генераторы псевдослучайных чисел, кодирование информации, симметричные и асимметричные системы шифрования. Изложены методы аутентификации, методы и средства электронной подписи, алгоритмы и протоколы аутентификации и обмена ключевой информации, криптографические протоколы. Рассмотрены вопросы криптографической защиты информации в вычислительных системах и сетях передачи данных. Материал учебного издания предназначен как для аудиторных занятий, так и для индивидуальной и самостоятельной работы обучающихся.

Для студентов учреждений среднего профессионального образования.

УДК 004.056.5(075.32)
ББК 32.973.26-018.2я723

ISBN 978-5-4468-0000-0

Уважаемый читатель!

Предисловие

Федеральные государственные образовательные стандарты среднего профессионального образования по специальностям 10.02.04 «Обеспечение информационной безопасности телекоммуникационных систем» и 10.02.05 «Обеспечение информационной безопасности автоматизированных систем» включают в структуру обязательной части профессионального цикла профессиональные модули 02, содержащие междисциплинарные курсы 02.02, посвященные изучению криптографической защиты информации в объектах информационной инфраструктуры, информационно-телекоммуникационных системах и автоматизированных системах в защищенном исполнении. В примерных основных образовательных программах среднего профессионального образования по специальностям 10.02.04 и 10.02.05 для изучения названых междисциплинарных курсов отводится 144 академических часа аудиторных занятий.

В соответствии с темами, включенными в примерные рабочие программы по этим междисциплинарным курсам, учебное издание состоит из четырех разделов, включающих в себя 12 глав.

В разделе I учебного издания рассматриваются основы криптографии.

В главе 1 изучаются математические основы криптографии.

В главе 2 даются понятия криптографии и криптографической защиты информации, определяются предметы и задачи криптографической защиты информации, приводятся основные термины в области криптографической защиты информации и их определения.

Рассматриваются открытые сообщения и их характеристики, модель открытого текста, основные задачи криптографии, виды шифрования, классификация и модели шифров, требования к шифрам и криптографические протоколы.

В главе 3 речь идет о кодировании и сжатии информации, различных разновидностях шифров перестановки, получении и применении псевдослучайных чисел.

В разделе II учебного издания подробно разбираются системы шифрования.

В главе 4 рассматриваются симметричные системы шифрования и построенные на их основе отечественные и зарубежные криптографические алгоритмы, проблема распределения ключей и управление ключами.

В главе 5 изучаются асимметричные системы шифрования, отечественные и зарубежные алгоритмы асимметричного шифрования, безопасность асимметричных систем шифрования.

В главе 6 рассматриваются вопросы криптоанализа и определения криптографической стойкости шифров.

Раздел III учебного издания рассматривает применение криптографии в различных областях деятельности.

В главе 7 речь идет об электронной подписи и ее использовании: алгоритмах формирования, свойствах и схемах электронной подписи, свойствах, обеспечиваемых электронной подписью.

В главе 8 рассмотрены вопросы применения криптографии в системах и средствах аутентификации и установления подлинности объектов.

В главе 9 освещаются вопросы криптографической защиты информации, передаваемой в сетях передачи данных, организации VPN-сетей.

В разделе IV учебного издания рассматриваются проблемы и направления развития криптографии.

В конце каждой главы всех четырех разделов учебного издания приведены вопросы и задания для текущего контроля знания обучающихся.

Раздел I

ОСНОВЫ КРИПТОГРАФИИ

Глава 1

МАТЕМАТИЧЕСКИЕ ОСНОВЫ КРИПТОГРАФИИ

1.1. ДЕЛИМОСТЬ ЧИСЕЛ. ПРИЗНАКИ ДЕЛИМОСТИ. ПРОСТЫЕ И СОСТАВНЫЕ ЧИСЛА

Для криптографии алгебра и теория чисел являются одним из основных инструментов в теоретических исследованиях и практических реализациях криптографических преобразований, поэтому в этой главе мы приводим основные понятия и определения, необходимые для понимания математического аппарата. В этой главе речь пойдет о целых числах и их свойствах.

Определение 1.1. Целое число a *делится* на b, если существует такое целое число q, для которого выполняется равенство $a = qb$.

Используются следующие обозначения: $(a \vdots b)$ — a делится на число b либо $(b \mid a)$ — b делит a. Для отношения делимости справедлив ряд свойств.

Свойство 1. Если a делится на b и b делится на c, то a делится на c.

Свойство 2. Если $a_1, \ldots, a_n$ делятся на b, то и $a_1 + \ldots + a_n$ делится на b.

Свойство 3. Если a_1 делится на b_1, ... , a_n делится на b_n, то и $a_1 a_2 \ldots a_n$ делится на $b_1 b_2 \ldots b_n$.

Имеет место следующая теорема о делении с остатком.

Теорема 1.1. Для любого целого числа a и натурального числа b существует единственная пара чисел q и r таких, что

$$a = bq + r,\ 0 \le r < b \qquad (1.1)$$

Число q называется ***неполным частным***, а число r — ***остатком***.

Доказательство. Пусть сначала $a \geqslant 0$. Будем выписывать одно за другим числа a, $a - b$, $a - 2b$, ... до тех пор, пока не появится отрицательное число. Пусть последним из неотрицательных

членов этой последовательности будет число $a - qb$. Обозначая его через r, мы имеем $a = qb + r$. Очевидно, что $r < b$, иначе число $r - b = a - (q + 1)b$ было бы неотрицательным. Пусть теперь $a < 0$. Рассуждая аналогично предыдущему, будем выписывать последовательность чисел a, $a + b$, $a + 2b$, ... до тех пор, пока не появится первое неотрицательное число r (легко проверить, что $r < b$). Пусть $r = a + q'b$. Тогда, обозначая $-q'$ через q, получаем $a = qb + r$, что и требовалось доказать.

Докажем единственность (1.1), т. е. что из $a = qb + r$ и $a = q'b + r'$ следует $q = q'$ и $r = r'$. В этом случае имеем равенство $qb + r = q'b + r'$, откуда $r - r' = (q' - q)b$, т. е. $(r - r')$ делится на b. Но $|r - r'| < b$, и равенство $r - r' = (q' - q)b$ возможно только в случае $r - r' = 0$. Но тогда $(q' - q)b = 0$ и, следовательно, $q' - q = 0$.

Заметим, что если остаток r равен нулю, то число a делится на число b.

Признаки делимости. Сформулируем простейшие признаки делимости.

1. *Признак делимости на 2.* Число делится на 2, если число, образованное его последней цифрой в десятичной записи, делится на 2.

2. *Признак делимости на 3.* Число делится на 3, если сумма чисел, образованных его цифрами в десятичной записи, делится на 3.

Доказательство. Число 10 равноостаточно с 1. Поэтому 100 равноостаточно с 1, 1 000 равноостаточно с 1 и т. д. Таким образом, число $\overline{a_n \ldots a_1 a_0} = a_0 + a_1 \cdot 10 + \ldots + a_n \cdot 10^n$ равноостаточно с $a_0 + a_1 + \ldots + a_n$. Фактически мы доказали, что число дает при делении на 3 такой же остаток, что и сумма чисел, образованных цифрами этого числа в десятичной записи.

3. *Признак делимости на 4.* Число делится на 4, если число, образованное двумя последними цифрами в его десятичной записи, делится на 4. Доказательство вытекает из того, что число 100 и его кратные делятся на 4.

4. *Признак делимости на 5.* Число делится на 5, если его последняя цифра в десятичной записи 0 или 5.

5. *Признак делимости на 9.* Число делится на 9, если сумма чисел, образованных его цифрами в десятичной записи, делится на 9. Доказательство аналогично предыдущему.

6. *Признак делимости на 8.* Число делится на 8, если число, образованное последними тремя цифрами в десятичной записи, делится на 8. Доказательство вытекает из того, что число 1 000 и его кратные делятся на 8.

7. *Признак делимости на 11.* Число делится на 11, если алгебраическая сумма чисел, образованных его цифрами в десятичной записи с чередующимися знаками, делится на 11.

Доказательство. Число 10 равноостаточно с -1. Поэтому 100 равноостаточно с $(-1) \cdot (-1) = 1$, 1 000 равноостаточно с -1 и т. д. Таким образом, число $\overline{a_n \ldots a_1 a_0} = a_0 + a_1 \cdot 10 + \ldots + a_n \cdot 10^n$ равноостаточно с $a_0 - a_1 + \ldots + (-1)^n \cdot a_n$.

Пример 1.1. Рассмотрим число 3 516 282. Алгебраическая сумма его цифр равна $2 - 8 + 2 - 6 + 1 - 5 + 3 = -11$. Следовательно, данное число делится на 11.

8. *Объединенный признак делимости на 7, 11 и 13.* Число делится на 7, 11 или 13, если алгебраическая сумма чисел, образованных тройками цифр данного числа в десятичной записи с чередующимися знаками, делится соответственно на 7, 11 или 13.

Доказательство. Заметим, что произведение чисел 7, 11 и 13 равно 1 001. Поэтому число 1000 при делении на 7, 11 или 13 равноостаточно с -1. Далее поступаем, как и в признаке делимости на 11.

Пример 1.2. Рассмотрим число 42 623 295. Число $295 - 623 + 42 = -286$ делится на 11 и 13, но не делится на 7. Следовательно, число делится на 11 и 13, но не делится на 7.

9. *Признак делимости на 37.* Число делится на 37, если сумма чисел, образованных тройками цифр данного числа в десятичной записи, делится соответственно на 37. Доказательство вытекает из того, что число 1 000 при делении на 37 равноостаточно с 1. Заметим также, что трехзначные числа 111, 222, ..., 999 делятся на 37. Легко видеть, что числа 356 643, 123 432, 215 673 делятся на 37.

Определение 1.2. Целое число p называется *простым*, если $p \neq \pm 1$ и единственными его делителями являются числа ± 1 и $\pm p$.

Так, числа 2, 3, 5 и -7 простые, а число $45 = 5 \cdot 9$ — нет. Почти всюду мы будем называть простым положительное простое число.

Целое число, отличное от ± 1 и не простое, называется *составным*, или *разложимым*. Для составного числа n существуют такие целые числа a и b, что $1 < a, b < n$ и $n = a\,b$. Значит, число $45 = 5 \cdot 9$ составное. Заметим, что числа ± 1 не являются ни составными, ни простыми. Они относятся к третьей группе — это единственные целые числа, у которых есть целые обратные.

Определение 1.3. Два числа называются *взаимно простыми*, если они не имеют общих натуральных делителей, кроме единицы.

1.2. ОСНОВНАЯ ТЕОРЕМА АРИФМЕТИКИ. НАИБОЛЬШИЙ ОБЩИЙ ДЕЛИТЕЛЬ

Теорема 1.2 (разложение на множители). Всякое целое число $n \geq 2$ единственным образом записывается в виде

$$n = p_1^{e_1} p_2^{e_2} \ldots p_k^{e_k}, \qquad (1.2)$$

где $p_1, p_2, \ldots, p_k$ — простые числа, $1 < p_1 < p_2 < \ldots < p_k$ и $e_1, e_2, \ldots, e_k$ — натуральные числа.

Правая часть (1.) называется ***каноническим разложением***, а саму теорему 1.2 — ***основной теоремой арифметики***. Показатели $e_1, e_2, \ldots, e_k$ — кратности простых множителей в разложении числа n. Другими словами, кратность множителя p_i в разложении числа n — наибольшее число e_i такое, что n делится на $p_i^{e_i}$. Отметим также, что у числа n есть k различных простых делителей, однако общее число его простых делителей равно $e_1 + e_2 + \ldots + e_k$.

После того как теорема о разложении сформулирована, понятно почему ±1 не следует считать простыми. Если включить их в число простых, то разложение на простые сомножители потеряет свойство единственности: $2 = 1 \cdot 2 = 1^1 \cdot 2 = \ldots$. С целью избежать этих псевдоразложений (множество которых бесконечно и бессмысленно), исключили ±1 из определения простого числа.

Определение 1.4. ***Наибольшим общим делителем*** **(*НОД*)** двух натуральных чисел a и b называется наибольшее из чисел, являющихся делителями a и b одновременно.

Докажем, что такое число существует. Действительно, если c является делителем натурального числа a, то $|c| \leq a$. Следовательно, у каждого натурального числа имеется конечное число делителей. Таким образом, число общих делителей двух натуральных чисел конечно и, значит, среди них есть наибольший элемент — наибольший общий делитель.

1.3. ВЗАИМНО ПРОСТЫЕ ЧИСЛА. АЛГОРИТМ ЕВКЛИДА НАХОЖДЕНИЯ НОД

Определить, является ли b делителем числа a, можно, подсчитав остаток от деления a на b и проверив, равен ли он нулю. Алгоритм Евклида предназначен для вычисления наибольшего общего делителя двух натуральных чисел.

Определение 1.5. Пусть a и b — натуральные числа. Наибольший общий делитель чисел a и b — это наибольшее целое число d, на которое и a, и b делятся; тогда имеем

$$\text{НОД}(a, b) = d. \qquad (1.3)$$

Определение (1.) подсказывает ***алгоритм вычисления НОД***. Если числа a и b заданы, то найдем все положительные делители числа a и все положительные делители числа b. Выберем все числа, входящие в оба множества, и возьмем наибольшее из них. Оно и будет наибольшим общим делителем. Эта процедура проста, однако, она неэффективна при больших a и b: *неизвестно ни одного простого алгоритма разложения целых чисел на множители.*

Алгоритм Евклида. Наибольший общий делитель можно найти с помощью алгоритма Евклида. Для этого разделим a на b с остатком; назовем этот остаток r_1. Если $r_1 \neq 0$, то разделим b на r_1 с остатком; пусть r_2 — остаток второго деления. Аналогично: если $r_2 \neq 0$, то разделим r_1 на r_2 и получим новый остаток r_3.

Таким образом, i-й цикл алгоритма состоит из одного деления с остатком, причем делимое равно остатку, полученному в $(i-2)$-м цикле, а делитель — остатку, полученному в $(i-1)$-м цикле. Цикл повторяется до тех пор, пока мы не получим нулевого остатка; наименьший ненулевой остаток является наибольшим общим делителем чисел a и b.

Пример 1.3. Вычислить наибольший общей делитель чисел 1 234 и 54.

Решение. Применим алгоритм Евклида:

$$1\,234 = 22 \cdot 54 + 46;$$

$$54 = 1 \cdot 46 + 8;$$

$$46 = 5 \cdot 8 + 6;$$

$$8 = 1 \cdot 6 + 2;$$

$$6 = 3 \cdot 2 + 0.$$

Последний ненулевой остаток равен 2, поэтому НОД (1 234,54) = = 2.

Отметим, что неполные частные не принимают непосредственного участия в подсчете наибольшего общего делителя.

Опишем теперь алгоритм Евклида.

Ввод: натуральные числа a и b, $a > b$.

Вывод: наибольший общий делитель чисел a и b.

Шаг 1. Положить $A = a$ и $R = B = b$.

Шаг 2. Заменить значение R остатком от деления A на B и перейти к шагу 3.

Шаг 3. Если $R = 0$, то сообщить: «наибольший общий делитель чисел a и b равен B», и остановиться; в противном случае перейти к шагу 4.

Шаг 4. Заменить значение A на значение B, значение B на значение R и возвратиться к шагу 2.

Конец алгоритма.

Следует доказать, что в последовательности остатков всегда появится нуль и наибольший общий делитель совпадает с последним ненулевым остатком в последовательности делений.

Обоснуем первое и тем самым мы докажем, что алгоритм всегда завершает работу. Предположим, что для того чтобы найти наибольший общий делитель чисел a и b, мы проделали следующие деления с остатком (1.1):

$$a = bq_1 + r_1,\ 0 \le r_1 < b;$$

$$b = r_1q_2 + r_2,\ 0 \le r_2 < r_1;$$

$$r_1 = r_2q_3 + r_3,\ 0 \le r_3 < r_2;$$

$$r_2 = r_3q_4 + r_4,\ 0 \le r_4 < r_3.$$

В правом столбце стоит последовательность остатков, в которой всякий остаток меньше предыдущего, а также, что все остатки неотрицательны. Переписав неравенства последовательно, получаем цепочку

$$0 \le \ldots r_3 < r_2 < r_1 < b. \qquad (1.4)$$

Поскольку между b и нулем есть лишь конечное число целых чисел, последовательность остатков не может продолжаться бесконечно: в конце ее может стоять только нуль, а значит, алгоритм заведомо остановится.

Можно получить верхнюю оценку на число делений, необходимое для вычисления наибольшего общего делителя. Из неравенств (1.4) следует, что каждое число в последовательности строго меньше предыдущего. Поэтому наибольшее возможное значение остатка в каждом делении на единицу меньше значения остатка на предыдущем делении. Если бы в каждом цикле это наибольшее возможное значение достигалось, то для получения нулевого остатка нам потребовалось бы b делений. Ясно, что это и есть наихудший возможный случай. Поэтому при применении алгоритма Евклида к паре чисел a и b число делений не превосходит b. Зафиксируем число n. Тогда задача формулируется так: для каких наименьших взаимно простых a и b вычисление НОД (a, b) требует n делений?

Заметим, что для того чтобы числа a и b были минимально возможными, частные на каждом шаге тоже должны быть минимально возможными. Если теперь предположить, что делитель меньше делимого, то ясно, что наименьшее возможное частное двух целых чисел равно 1. Предположим, что мы выполнили n делений до получения нулевого остатка. Тогда последовательность остатков имеет вид (1.4).

В наихудшем возможном случае все неполные частные равны 1. Запишем теперь все деления, начиная с последнего. В силу взаимной простоты чисел мы получаем

$$r_{n-1} = 1;$$

$$r_{n-3} = r_{n-2} \cdot 1 + 1;$$

$$r_{n-4} = r_{n-3} \cdot 1 + 1;$$

$$\ldots\ \ldots\ \ldots\ \ldots\ \ldots$$

$$a = b \cdot 1 + r_1$$

или следующую последовательность остатков при $n = 10$: 34, 21, 13, 8, 5, 3, 2, 1, 1, 0. Значит, наименьшая пара взаимно простых чисел a и b, для подсчета наибольшего общего делителя которых необходимо 10 делений с остатком, это $a = 34$ и $b = 21$. Заметьте, что, хотя число $b = 21$ и наименьшее, все равно оно больше, чем $n = 10$.

1.4. РАСШИРЕННЫЙ АЛГОРИТМ ЕВКЛИДА

У алгоритма Евклида, есть еще один вариант, более мощный. Достоинство нового варианта в том, что наибольший общий делитель — лишь часть выходных данных. Пусть a и b — натуральные числа, а d — их наибольший общий делитель. Расширенный алгоритм Евклида подсчитывает не только d, но и два целых числа α и β таких, что

$$\alpha a + \beta b = d. \qquad (1.5)$$

Отметим, что (за исключением нескольких тривиальных случаев) если a оказывается положительным, то β — отрицательное, и наоборот. Вычислять эти числа следует, добавив инструкции в обычный алгоритм Евклида так, чтобы d, α и β подсчитывались одновременно.

Напомним, что алгоритм Евклида состоит из последовательности делений с остатком. Наибольший общий множитель представляет собой последний ненулевой остаток в этой последовательности. Значит, нам надо найти способ записывать последний ненулевой остаток в виде (1.5).

Предположим, что для вычисления наибольшего общего делителя чисел a и b мы выполнили последовательность делений (1.1). Перепишем ее, сопровождая каждую операцию записью предполагаемого представления остатка:

$$a = bq_1 + r_1,\ r_1 = ax_1 + by_1;$$

$$b = r_1q_2 + r_2,\ r_2 = ax_2 + by_2;$$

$$r_1 = r_2q_3 + r_3,\ r_3 = ax_3 + by_3;$$

$$r_2 = r_3q_4 + r_4,\ r_4 = ax_4 + by_4;$$

$$\dots\ \dots\ \dots\ \dots\ \dots\ \dots\ \dots\ \dots\ \dots\ ;$$

$$r_{n-3} = r_{n-2}\ q_{n-1} + r_{n-1},\ r_{n-1} = ax_{n-1} + by_{n-1};$$

$$r_{n-2} = r_{n-1}\ q_n,\ r_n = 0. \qquad (1.6)$$

Числа $x_1, x_2, \dots, x_{n-1}$ и $y_1, y_2, \dots, y_{n-1}$ следует определить. Сведем их в табл. 1.1.

Отметим прежде всего, что таблица начинается с двух строчек, которым в ней не следовало бы быть. Действительно, стоящие в первом столбце этих строк числа не являются остатками в каких-либо операциях деления. Мы даем этим строчкам номера –1 и 0. Далее мы обоснуем их необходимость.

Таблица 1.1. Выполнение деления с помощью алгоритма Евклида

Номер шага	Остаток	Частное	x	y
1	a	—	x_{-1}	y_{-1}
2	b	—	x_0	y_0
3	r_1	r_1	x_1	y_1
4	r_2	r_2	x_2	y_2
5	r_3	q_3	x_3	y_3
⋮	⋮	⋮	⋮	⋮
n–2	r_{n-2}	q_{n-2}	x_{n-2}	y_{n-2}
n–1	r_{n-1}	q_{n-1}	x_{n-1}	y_{n-1}

Допустим, что мы получили таблицу заполненной до некоторой, скажем, $(j-1)$-й строки. Для заполнения j-й строки следует разделить r_{j-2} на r_{j-1}. В результате получатся r_j и q_j — первые два элемента j-й строки. Не будем забывать, что $r_{j-1} = r_{j-1}q_j + r_j$ и $0 \le r_j < r_{j-1}$. Таким образом,

$$r_j = r_{j-2} - r_{j-1}q_j. \qquad (1.7)$$

Из $(j–1)$-й и $(j–2)$-й строк найдем величины $x_{j-2}, x_{j-1}, y_{j-2}, y_{j-1}$, и запишем

$$r_{j-2} = ax_{j-2} + by_{j-2},\ r_{j-1} = ax_{j-1} + by_{j-1}.$$

Подставив эти значения в (1.), получим:

$$r_j = (ax_{j-2} + by_{j-2}) - (ax_{j-1} + by_{j-1})\ q_j = a\ (x_{j-2} - q_j\ x_{j-1}) + b\ (y_{j-2} - q_j\ y_{j-1}).$$

Поэтому $x_j = x_{j-2} - q_j\ x_{j-1}$, $y_j = y_{j-2} - q_j\ y_{j-1}$.

Заметим, что для вычисления x_j и y_j нам понадобились только частное q_j и данные из двух строк таблицы, непосредственно предшествующих j-й. Итак, мы получили рекуррентную процедуру. Все, что необходимо — запустить ее. Для этого и были добавлены в таблицу две дополнительные строки. Найти в них значения x и y очень просто. Придавая им тот же смысл, что и в остальных строках, мы должны получить

$$a = ax_{-1} + by_{-1},\ b = ax_0 + by_0.$$

Таким образом, можно просто положить

$$x_{-1} = 1,\ y_{-1} = 0,\ x_0 = 0,\ y_0 = 1,$$

и процедуру можно запускать.

В результате цепочки делений с остатком получим равенство НОД $(a, b) = r_{n-1}$ и вычислим такие целые числа x_{n-1} и y_{n-1}, что

$$d = r_{n-1} = ax_{n-1} + by_{n-1}.$$

Значит, $\alpha = x_{n-1}$ и $\beta = y_{n-1}$. Заметим, что, зная α и $d = r_{n-1}$, мы можем найти β по формуле

$$\beta = \frac{d - a\alpha}{b}.$$

Поэтому достаточно вычислять только первые три столбца таблицы.

Пример 1.4. С помощью расширенного алгоритма Евклида вычислить НОД (1 234, 54).

Решение. Для нахождения НОД составим табл. 1.2.

Таблица 1.2. Нахождения НОД с помощью расширенного алгоритма Евклида

Номер шага	Остаток	Частное	x	y
1	1 234	—	1	0
2	45	—	0	1
3	46	22	$1 - 22 \cdot 0 = 1$	$0 - 22 \cdot 1 = -22$
4	8	1	$0 - 1 \cdot 1 = -1$	$1 - 1 \cdot (-22) = 23$
5	6	5	$1 - 5 \cdot (-1) = 6$	$-22 - 5 \cdot 23 = -137$
6	2	1	$-1 - 1 \cdot 6 = -7$	$23 - 1 \cdot (-137) = 160$
7	0	3		

Поэтому $\alpha = -7$, $\beta = 160$ и $(-7) \cdot 1234 + 160 \cdot 54 = 2$.

Расширенный алгоритм Евклида приводит к следующей теореме.

Теорема 1.3. Пусть d — наибольший общий делитель натуральных чисел a и b. Тогда существуют такие целые числа α и β, что

$$\alpha a + \beta b = d. \tag{1.8}$$

Заметим, что пара чисел α, β в (1.8) не единственная. на самом деле, таких пар бесконечно много. Например, возьмем α и β, для которых $\alpha a + \beta b = d$, и рассмотрим какое-нибудь целое k. Тогда, как несложно проверить,

$$(\alpha + kb)\ \alpha + (\beta - ka)\ b = d.$$

1.5. ФУНКЦИЯ ЭЙЛЕРА

Определение 1.6. Числовая функция $\xi(m)$ натурального аргумента m называется *мультипликативной*, если $\xi(m_1 m_2) = \xi(m_1)\xi(m_2)$ для любых взаимно простых натуральных чисел m_1, m_2.

Пример 1.5. Функция $\xi(m) = m^s$ является мультипликативной при любом вещественном (или даже комплексном) значении s.

Перечислим простейшие свойства мультипликативных функций.

1. $\xi(1) = 1$.

2. Для любых попарно взаимно простых чисел $m_1, \ldots, m_k$ имеем

$$\xi(m_1 \ldots m_k) = \xi(m_1) \ldots \xi(m_k).$$

В частности, если $m = p_1^{\alpha_1} p_2^{\alpha_2} \ldots p_t^{\alpha t}$ — каноническое разложение числа m, то

$$\xi(m) = \xi\left(p_1^{\alpha_1}\right)\xi\left(p_2^{\alpha_2}\right)\ldots\xi\left(p_t^{\alpha_t}\right). \tag{1.9}$$

3. Мультипликативная функция $\xi(m)$ может быть задана следующим способом: произвольно задаем значения вида $\xi(p^\alpha)$, где p — простое, α — натуральное; остальные значения определяем формулой (1.9).

4. Если $\xi_1(m)$ и $\xi_2(m)$ — мультипликативные функции, то их произведение $\xi(m) = \xi_1(m)\ \xi_2(m)$ также будет мультипликативной функцией.

Доказательство этих свойств непосредственно вытекает из определения 1.6 и предоставляется читателю как упражнение.

В следующей теореме указан еще один способ конструирования мультипликативных функций.

Теорема 1.4. Пусть $\xi(m)$ — мультипликативная функция. Положим

$$\eta(m) = \sum_{d|m} \xi(d).$$

Тогда $\eta(m)$ — также мультипликативная функция. Здесь и далее символ $\sum_{d|m}$ означает суммирование по всем натуральным делителям d числа m.

Определение 1.7. Функция Эйлера $\varphi(m)$ определяется как количество чисел в ряду 0, 1, 2, ..., m–1, взаимно простых с m.

Пример 1.6. $\varphi(1) = \varphi(2) = 1$, $\varphi(3) = \varphi(4) = 2$, $\varphi(5) = 4$, $\varphi(6) = 2$.

Теорема 1.5. Функция Эйлера является мультипликативной.

1.6. ЭЛЕМЕНТЫ ТЕОРИИ МНОЖЕСТВ

Множество — начальное, неопределяемое понятие, как точка или прямая в геометрии, натуральное число в арифметике. Представления о натуральном числе или прямой получаются из неформальных разъяснений и примеров: 3 яблока, 3 карандаша и т. д. С помощью абстракции дают представление о числе 3. Луч света или натянутая проволока дают первоначальное представление о прямой и т. п. Далее первоначальные представления уточняются в процессе изучения свойств этих объектов. Аналогично, представление о множестве дает любое семейство или совокупность

объектов: слова в словаре, все положительные действительные числа, дни недели и т. п. Объекты, из которых состоит множество, называются *элементами множества*.

Множества обозначаются символами (обычно из прописных букв): A, B, C и т. д. Элементы множеств обозначаются строчными буквами или буквами с индексами, например a, x, c_{ij} и т. д. Для основных числовых множеств приняты стандартные обозначения: N — множество натуральных чисел, Z — множество целых чисел, Q — множество рациональных чисел, R — множество действительных чисел, C — множество комплексных чисел.

Для обозначения того, что объект x является элементом множества A, применяется формальная запись $x \in A$ или $A \ni x$, которая читается так: x принадлежит A или A содержит x соответственно. Если x не принадлежит A, применяется запись $x \notin A$. Например, $0{,}3 \notin N$, однако $0{,}3 \in Q$. Для задания определенного множества A надо указать те элементы, которые ему принадлежат. Это можно сделать несколькими способами, например, перечислением элементов: $A = \{a_1, a_2, \ldots, a_n\}$; заданием характеристического свойства: $B = \{x | x \in Z$, четное$\}$; порождающей процедурой:

$$F = \{x_k \mid x_0 = 0,\ x_1 = 1,\ x_k = x_{k-2} + x_{k-1}\}.$$

Отметим, что знак $|$, встречающийся внутри фигурных скобок в определении множеств B и F, читается как «... такие, что ...». Например, определение B «дословно» читается так: B — это множество всех x таких, что x — целое и четное. Конечно, это можно сказать более естественно: B — множество целых четных чисел. Главное, чтобы при этом не искажался смысл. Ясно, что перечислением можно задавать только конечные множества. Любые множества можно задавать с помощью порождающих процедур или характеристических свойств элементов. Можно строить новые множества из имеющихся с помощью некоторых операций, подобно тому, как, например, из чисел a и b строится число $a + b$ в арифметике. Для определения основных операций называемых алгеброй множеств будем использовать специальные знаки:

$\Rightarrow$ — если A, то B, или из A следует B;
$\Leftrightarrow$ — A истинно тогда и только тогда, когда истинно B;
$\neg$ — не A, неверно, что A;
$\vee$ — или A или B (или оба);
$\wedge$ — и A и B;
$\forall$ — для всякого ..., любой, всякий;
$\exists$ — существует, найдется такой ..., что ...;
$\exists!$ — существует и при том только один, такой ... .

Все знаки, кроме трех последних, связывают какие-то утверждения, обозначенные через A и B, и называются логическими связками. Три последних знака обращаются к элементам множества и указывают «количество» элементов, имеющих некоторое свойство: все или хотя бы один — и называются кванторами (*quantum* — количество, сумма). Конечно, ценность этих знаков не только в сокращении письма. В дальнейшем будет показано, что они являются полной основой языка, достаточного для записи любого математического утверждения.

Для чисел имеются такие основные операции, как сумма и произведение, а также отношения неравенства и равенства. При построении теории множеств — аналогичная картина.

Определение 1.8. Множество A называется подмножеством (частью) множества B, и это обозначается так: $A \subseteq B$, если все элементы из A являются также и элементами множества B.

С использованием введенных связок это можно записать так:

$$A \subseteq B \Leftrightarrow \forall x\ (x \in A \Rightarrow x \in B).$$

Последняя запись может быть прочитана так: A является подмножеством B тогда и только тогда, когда для любого x из A, следует, что он содержится в B. Для множеств это отношение включения напоминает отношение неравенства для чисел; во всяком случае, если конечное множество A является подмножеством конечного множества B, то количество элементов в A не больше числа элементов в B. Количество элементов в конечном множестве A будем обозначать так: $|A|$.

Назовем два множества равными и будем это обозначать так: $A = B$, если $(A \subseteq B) \wedge (B \subseteq A)$. Определим теперь некоторые операции над множествами:

1) объединение $A \cup B = \{x | (x \in A) \vee (x \in B)\}$;
2) пересечение $A \cap B = \{x | (x \in A) \wedge (x \in B)\}$;
3) разность $A \backslash B = \{x | (x \in A) \wedge (x \notin B)\}$;
4) симметрическая разность $A \Delta B = (A \cup B) \setminus (A \cap B) = (A \backslash B) \cup (B \backslash A)$.

Следует привести словесные прочтения данных определений. Например, объединением множеств A и B называется множество всех элементов, принадлежащих A или B (возможно, обоим) и аналогично для оставшихся определений. В дальнейшем будет использоваться смесь естественного и формального языков. Операции объединения и пересечения обобщаются на семейства (наборы) множеств. Пусть I — некоторое множество, каждому $i \in I$ соответствует множество A_i, тогда множество

$$\bigcup_{i\in I} A_i = \{x | \exists i \in I : x \in A_i\}$$

называется объединением семейства множеств A_i (по множеству индексов I). Аналогично определяется пересечение семейства множеств:

$$\bigcup_{i\in I} A_i = \{x | \forall i \in I : x \in A_i\}.$$

Отметим, что симметрическая разность выражается через ранее определенные операции и потому может считаться неосновной. Однако эта операция очень естественна (она дает величину «несовпадения» множеств), и к тому же через нее и одну из предыдущих операций остальные тоже выражаются, как можно проверить, так что ее можно брать за начальную. Для графической иллюстрации операций с множествами используются диаграммы Эйлера (рис. 1.1), в которых множества условно изображаются кругами или частями кругов, а результат операции выделяется штриховкой или цветом.

На рис. 1.1 более темным цветом выделены результаты применения операций объединения — $A \cup B$, пересечения — $A \cap B$, разности $A\backslash B$ и симметрической разности $A \Delta B$ к множествам A и B. Диаграммы Эйлера в ряде случаев наглядно представляют результат применения к множествам и нескольких операций. Отметим, что диаграмма является лишь иллюстрацией, но не средством доказательства, как и чертеж в геометрии.

Аналогично свойствам операций над числами, таким как коммутативность и ассоциативность сложения и умножения и т. п., имеется ряд основных свойств операций над множествами.

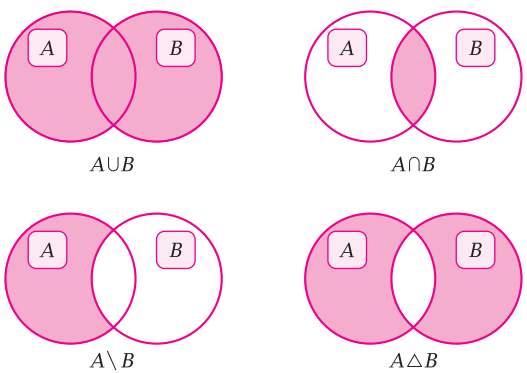

Рис. 1.1. Диаграммы Эйлера

Теорема 1.6. Справедливы следующие операции над множествами:

1) ассоциативность объединения и пересечения

$$(A \cup B) \cup C = A \cup (B \cup C);\ (A \cap B) \cap C = A \cap (B \cap C);$$

2) коммутативность объединения и пересечения

$$A \cup B = B \cup A;\ A \cap B = B \cap A;$$

3) дистрибутивность пересечения относительно объединения, дистрибутивность объединения относительно пересечения

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C);\ A \cup (B \cap C) = (A \cup B) \cap (A \cup C);$$

4) поглощение

$$(A \cap B) \cup A = A;\ (A \cup B) \cap A = A.$$

Приведем для примера формальное доказательство дистрибутивности пересечения относительно объединения. Отметим, что для доказательства равенства двух множеств необходимо, согласно определению, показать, что всякий элемент из одного множества содержится во втором, и наоборот, любой элемент второго содержится в первом множестве:

$$x \in A \cap (B \cup C) \Rightarrow x \in A \wedge x \in (B \cup C) \Rightarrow x \in A \wedge (x \in B \vee x \in C)$$

$$\Rightarrow (x \in A \wedge x \in B) \vee (x \in A \wedge x \in C) \Rightarrow x \in (A \cap B) \vee x \in (A \cap C)$$

$$\Rightarrow x \in (A \cap B) \cup (A \cap C).$$

Показано, что $A \cap (B \cup C) \subseteq (A \cap B) \cup (A \cap C)$. Обратное включение доказывается аналогично — все логические стрелки следует обратить. Отметим, что приведенные выкладки никак не являются полным доказательством, не требующим ни одного слова пояснения. Это только некоторая форма, которая должна быть дополнена пояснениями.

В любой математической теории рассматривается некоторое основное множество объектов, а другие множества являются его подмножествами. В связи с этим при рассмотрении систем подмножеств некоторого множества приняты такие обозначения: основное множество называется универсумом и обозначается через U, а множество всех его подмножеств называется булеан U и обозначатся через $B(U)$. Одним из его элементов является само множество U: $U \in B(U)$, другой крайний случай — пустое множество $\varnothing \in B(U)$. Пустое множество $\varnothing$ — аналог нуля в арифметике, это множество, не содержащее ни одного элемента. Можно ска-

зать, что пустое множество является подмножеством любого множества. Для элементов булеана введем еще одну операцию — дополнение множества A внутри универсума:

$$\bar{A} = U \backslash A.$$

Конечно, и A и $\bar{A}$ являются подмножествами U и элементами булеана. Для подмножеств универсума выполненыдополнительные свойства.

Теорема 1.7. Справедливы следующие:

1) свойства нуля и единицы

$$A \cup \varnothing = A;\ A \cap \varnothing = \varnothing;\ A \cup U = U;\ A \cap U = A;$$

2) законы де Моргана

$$A \cup B = A \cap B;\ A \cap B = A \cup B;\ A = A;$$

3) свойства дополнения и разности

$$A \cup A = U;\ A \cap A = \varnothing;\ A \backslash B = A \cap B.$$

1.7. БИНАРНЫЕ ОПЕРАЦИИ

Пусть задано два множества A и B. Выделим некоторое подмножество $\Re$ декартова произведения $A \times B$ (всевозможных упорядоченных пар элементов из этих множеств) и будем трактовать его элементы (a_i, b_j) как выражение того факта, что a_i и b_j находятся в некотором соответствии. Нас не интересует характер этого соответствия, а только сам факт его существования. Множество $\Re$ назовем ***отображением множества A на множество B***.

Если $(a_i, b_j) \in \Re$, то a_i называется прообразом b_j, а b_j — образом a_i при отображении $\Re$. Множество $\tilde{A}$ всех прообразов в A есть область определения отображения $\Re$, а множество $\tilde{B}$ всех образов в B — область значений $\Re$. Если $\tilde{B}$ равно B, то говорят об отображении на B, если $\tilde{B}$ — только часть B, то говорят об отображении в B.

Отображение будем обозначать как $A\ \Re\ B$, или $\Re(a) = b$. Если для каждого a_i образ b_j единственен, то отображение называют ***функциональным***. Если в $\Re$ все пары (a_i, b_j) переписать наоборот, как (b_j, a_i), получим отображение B в A, которое является ***обратным*** к $\Re$ и обозначается $\Re^{-1}$. Пусть множества A и B совпадают, $\Re \subseteq A \times A$. В этом случае $\Re$ называют ***бинарным отношением***, а множество A — ***базовым множеством*** отношения $\Re$.

Если $(a_i, a_j) \in \Re$, то говорят, что элемент a_i находится в отношении с элементом a_j. В общем случае можно определить, что в отношении находится не пара, а k элементов и считать, что $\Re \subseteq A^k$. Величина k определяет арность отношения $\Re$. Говорят, что $\Re \subseteq A^k$ — k-арное отношение. Термин «отношение» используют также, если арность >2 и множества в декартовом произведении различны. Далее будем рассматривать бинарные отношения на множестве A.

Бинарное отношение $\Re$ как любое подмножество может быть представлено в виде перечисления, через указания свойства или через порождающую процедуру. Но наиболее часто используется представление матрицей, в котором учитывается специфика множества. Столбцам и строкам матрицы сопоставлены элементы базового множества A, значение элемента матрицы (a_i, a_j) равно 1, если $(a_i, a_j) \in \Re$, в противном случае значение этого элемента равно 0.

Виды бинарных отношений. Бинарное отношение называется ***рефлексивным***, если $\forall\ (a_i) \in A\ (a_i, a_i) \in \Re$. Если отношение рефлексивное, то в каждой клетке главной диагонали стоят единицы. Бинарное отношение ***антирефлексивно***, если $\forall\ (a_i) \in A\ (a_i, a_i) \notin \Re$. В антирефлексивном отношении главная диагональ не содержит ни одной единицы.

Бинарное отношение называется ***симметричным***, если из того, что $(a_i, a_j) \in \Re$, следует $(a_j, a_{-i}) \in \Re$. Для симметричного отношения таблица симметрична относительно главной диагонали. Бинарное отношение ***антисимметрично***, если из того, что $(a_i, a_j) \in \Re$, следует, что $(a_j, a_i) \notin \Re$.

Бинарное отношение называется ***транзитивным***, если из того, что $(a_i, a_j) \in \Re$ и $(a_{-j}, a_{-k}) \in \Re$, следует $(a_i, a_k) \in \Re$.

Определение 1.9. Бинарное отношение называется *отношением эквивалентности*, если оно одновременно рефлексивно, симметрично и транзитивно.

Два элемента связаны отношением эквивалентности, если они имеют одинаковое свойство из множества альтернативных свойств. Примерами таких отношений являются принадлежность студентов к одной учебной студенческой группе, отношение родства. Альтернативность предполагает, что случаи, когда один студент принадлежит к нескольким группам или один человек имеет разноцветные волосы, из рассмотрения исключаются (иначе не выполнялась бы транзитивность). Тогда множество разбивается на непересекающиеся подмножества элементов, удовлетворяющие свойству, которые при объединении покрывают все множе-

ство. Последнее обеспечивается свойством рефлексивности, когда для каждого элемента находится элемент, с которым он состоит в отношении (по крайней мере, с самим собой). Эти подмножества называются классами эквивалентности.

Справедливо утверждение: любому отношению эквивалентности однозначно сопоставляется разбиение множества и, обратно, любому разбиению множества однозначно сопоставляется отношение эквивалентности.

Говорят, что отношение $\Re$ отвечает ***свойству дихотомии (порядка)***, если из того, что $(a_i, a_j) \notin \Re$, следует, что $(a_j, a_i) \in \Re$. Значит, если выполняется свойство дихотомии, то в множестве любые два элемента находятся в данном отношении. Примером дихотомии может служить отношение «быть по возрасту не старше» между людьми. Здесь: если Иван старше Петра, то, значит, Петр не старше Ивана.

1.8. ГРУППЫ, КОЛЬЦА, ПОЛЯ

Определение 1.10. *Группой* называется множество G, на котором определена ассоциативная бинарная операция «$\circ$», содержащее элемент e такой, что для любого элемента $a \in G$ выполняется $e \circ a = a \circ e = a$, и существует элемент a^{-1} такой, что $a \circ a^{-1} = a^{-1} \circ a = e$. Указанный элемент e называется *единицей группы*. Элемент a^{-1} называется обратным к элементу a.

Легко показать, что единица группы единственна и что элемент, обратный к данному элементу, также определяется однозначно. Если операция «$\circ$» коммутативна, то ***группа называется коммутативной,*** или ***абелевой***. Заметим, что не все коммутативные операции ассоциативны. Например, если в качестве операции «$\circ$» использовать среднее арифметическое двух вещественных чисел, то эта коммутативная операция не ассоциативна. Действительно:

$$(a \circ a) \circ b = a \circ b = \frac{a+b}{2}, \quad a \circ \left(\frac{a+b}{2}\right) = \frac{3a+b}{4}.$$

Примерами абелевых групп являются: множества Z целых, Q рациональных, R действительных и C комплексных чисел с соответствующими операциями сложения; множества $Q\backslash\{0\}$, $R\backslash\{0\}$ и $C\backslash\{0\}$ отличных от нуля рациональных, действительных и комплексных чисел с соответствующими операциями умножения.

В теории групп используется две равноправных и эквивалентных друг другу терминологических системы: ***аддитивная и мультипликативная***. В аддитивной системе групповую операцию называют операцией сложения, а группы в аддитивной записи для краткости иногда будем назвать аддитивными. Группы, операцию в которых называют умножением, иногда именуются мультипликативными группами. Операцию аддитивной группы принято обозначать знаком «+», операцию мультипликативной группы обозначают знаком умножения «$\times$» или «$\cdot$» (или ее обозначение, по умолчанию, опускают). Единичный элемент аддитивной группы обозначается 0 и называется нулем. Элемент аддитивной группы, обратный к элементу a, обозначается $-a$ и называется противоположным к этому элементу. Единичный элемент мультипликативной группы обозначается 1 и называется единицей. А элемент обратный к a обозначается через a^{-1}.

Ассоциативность операции «$\circ$» позволяет записывать кратное произведение $(\ldots((a \circ a) \circ a) \circ \ldots \circ a) \circ a$, опуская скобки $a \circ a \circ a \circ \ldots \circ a$. Такая формула называется k-й степенью элемента a. В аддитивных группах k-я степень элемента a обозначается ka, в мультипликативных группах используется обозначение a^k. По определению $0 \cdot a = 0$ и $a^0 = 1$. В аддитивной группе определяется операция вычитания $a - b$ по следующей формуле $a - b = a + (-b)$. Результат $a - b$ называется разностью элементов a и b. В мультипликативной группе определяется операция деления $\frac{a}{b}$ по следующей формуле: $\frac{a}{b} = ab^{-1}$. Результат $\frac{a}{b}$ называется частным от деления элемента a на элемент b. Рассмотренные выше примеры аддитивных и мультипликативных групп представляют ***бесконечные группы***.

Группа, определенная на конечном множестве G, называется ***конечной***. Тривиальная (единичная) группа определена на одноэлементном множестве {e} и содержит только единицу. Число элементов конечной группы называется ***порядком группы***. Порядок тривиальной группы равен 1, простейшая нетривиальная группа имеет порядок 2. Порядком элемента g группы G называется наименьшее число n такое, что $g^n = e$. Порядок элемента g обозначается ord g. Элемент, порядок которого равен порядку группы, если он существует, называется образующим (примитивным) элементом группы.

Все ненулевые элементы группы могут быть представлены в виде степени образующего (примитивного) элемента. Группа, имеющая образующий элемент, называется ***циклической***.

Множество, на котором задано отношение эквивалентности, разбивается на классы эквивалентности $[a]$, где a — представитель класса. Совокупность классов эквивалентности есть фактор-множество множества, на котором определено это бинарное отношение эквивалентности.

Так, на множестве Z целых чисел относительно натурального числа m можно определить отношение $\{(x, y) \mid x \equiv y \bmod m\}$, где $x \equiv y \bmod m$ означает, что число m делит разность $x - y$. Это отношение называется ***отношением сравнения (эквивалентности) по модулю m***, а классы эквивалентности по этому отношению — классами, или ***классами вычетов по модулю m***. Фактор-множество по этому отношению сокращенно обозначают Z_m, аналогично, классы конгруэнтности обозначаются просто $[a]$. Легко видеть, что $x \equiv y \bmod m$ тогда и только тогда, когда $x \bmod m \equiv y \bmod m$, где $x \bmod m$ и $y \bmod m$ — остаток от деления числа x или числа y на m. на фактор-множестве Z_m можно определить арифметические операции. Сумму классов эквивалентности определяют следующим образом: $[x] + [y] = [x + y]$. Удобно в качестве представителей классов $[x]$ использовать наименьшие неотрицательные элементы $x \bmod m$ классов. Тогда операцию сложения можно описать в обозначениях этих представителей:

$$[x] + [y] = [x + y] \bmod m.$$

Легко показать, что фактор множество Z_m с только что описанной операцией сложения есть аддитивная группа с нулевым элементом $[0]$ и что противоположный к элементу $[a]$ группы есть элемент $-[a] = [m - a]$. Аналогично, на фактор-множестве Z_m вводится операция умножения по модулю m, т. е.

$$[x] \cdot [y] = [x \cdot y] = [x \cdot y] \bmod m.$$

При этом множество ненулевых классов конгруэнтности $[a]$, $a \neq 0$, имеющих обратный класс $[a^{-1}]$, где $a\, a^{-1} \bmod m \equiv 1$, образует мультипликативную группу, которая обозначается Z_m^*. Мультипликативной единицей является класс $[1]$. Множество классов, взаимно простых с модулем m, как раз и представляют группу Z_m^*. Порядок группы Z_m^* обозначается символом $\varphi(m)$ — функция Эйлера.

Группы Z_m^* и Z_m совпадают тогда и только тогда, когда m — простое число. Рассмотренные аддитивная и мультипликативная группы Z_m и Z_m^* изоморфны аддитивной и мультипликативной группам, заданным на множестве наименьших неотрицательных представителей этих классов, в соответствии с операциями сложения и умножения по модулю m на множестве этих представителей. Поэтому часто вместо группы на фактор-множестве рассматривают группы на множестве представителей классов, при этом представителей этих множества $Z_m = \{0,1, \ldots, m-1\}$ и $Z_m^* = \{[a] | a \text{ и } m \text{ взаимно просты}\}$ обозначают так же, как множества классов.

Подмножество H группы G, замкнутое относительно операций группы и являющееся группой с этими же операциями, называется *подгруппой* данной группы. Например, тривиальная группа есть подгруппа любой группы.

Теорема 1.8 (Лагранж). Если m — порядок группы G, а n — порядок элемента $g \in G$, то $g^m = e$ *и* $m \equiv 0 \bmod n$ (порядок элемента делит порядок группы).

Теорема 1.9. Для любой группы G при любом натуральном k порядок t элемента a^k определяется равенством $t = \dfrac{\delta}{\text{НОД}(k,\delta)}$,, где $\delta = \text{ord}\, a$. В частности, $\text{ord}\, a^k = \delta$ тогда и только тогда, когда НОД $(k, \delta) = 1$ (k и d — взаимно простые).

Определение 1.11. Кольцо R — множество с операциями сложения и умножения такими, что R является абелевой группой относительно сложения и умножения, ассоциативна и дистрибутивна относительно операции сложения:

$$(ab)c = a\ (bc),$$

$$a(b + c) = ab + ac,$$

$$(b + c)\ a = ba + ca.$$

Следствием определения кольца является следующее свойство: для любого a

$$a \cdot 0 = 0 \cdot a = 0.$$

Примерами колец служат множества Z целых, Q рациональных и R действительных чисел с операциями сложения и умножения. Кольцо, в котором из уравнения $a\, b = 0$ следует, что $a = 0$ или $b = 0$, называется областью целостности. Если в кольце имеется мультипликативная единица, то кольцо называется кольцом с единицей. Ниже рассматриваются только кольца с единицей.

Определение 1.12. Элемент a' кольца с единицей такой, что $a \cdot a' = 1$, называется *обратным* к элементу a. Элемент, обратный к элементу a кольца, обозначается a^{-1}. Каждый элемент кольца имеет не более одного обратного к нему элемента. Элемента, обратного к нулевому элементу кольца, не существует. Множество

элементов кольца, имеющих обратный элемент, составляет *мультипликативную группу кольца R*, которая обозначается R^*.

Определение 1.13. *Полем* называется кольцо F с единицей, множество ненулевых элементов которого с операцией умножения, является абелевой группой. Эта группа называется *мультипликативной группой поля.*

Примерами ***бесконечных полей*** являются поля Q рациональных, R действительных и C комплексных чисел. Подмножество F поля Q, замкнутое относительно обеих операций и являющееся полем, называется подполем, что обозначается $F^* \subseteq Q$. Поле, которое не имеет подполя, не совпадающего с самим полем, называется простым полем. Существует единственное простое бесконечное поле — это поле Q рациональных чисел.

Конечные поля называются ***полями Галуа*** — по имени французского математика Эвариста Галуа. Далее рассматриваются и используются, как правило, конечные поля. ***Порядком поля*** называется число элементов. Конечное поле порядка q обозначается $GF(q)$, или F_q.

Пример 1.7. Простейшим полем является поле из двух элементов — поле $GF(2)$. Операции этого поля определяются таблицами, из которых следует, что сложения соответствует булевой функции сложения по модулю 2, а умножение — конъюнкции (табл. 1.3).

Пример 1.8. Если p — простое число, то целые числа {0, 1, 2, ..., p–1} образуют поле $GF(p)$. При этом все операции (сложения, вычитания, умножения, деления) выполняются по модулю p.

Пример 1.9. Если p — простое число, а n — натуральное, то поле, которое содержит $N = p^n$ элементов, не может быть образовано из совокупности целых чисел по модулю N. Действительно, для $p = 2$ и $n = 2$ число элементов $N = p^n = 2^2 = 4$. В множестве классов вычетов по модулю 4 элемент 2 не имеет обратного, так как $2 \cdot 2 = 0 \bmod 4$, т.е. множество, состоящее из четырех элементов, совсем не похоже на поле $GF(N)$, которое состоит из $p^n = N$ элементов. Элементами поля, которое состоит из p^n элементов, являются все многочлены степени не более $(n - 1)$ с коэффициентами из поля $G(p)$. Чтобы подчеркнуть эту разницу между представленными полями, поля из p^n элементов обозначают через $GF(p^n)$ (вместо $GF(N)$). Поля $GF(p^n)$ будут рассматриваться ниже. Мультипликативная группа конечного поля порядка q обозначается $GF(q)^*$ и имеет порядок на единицу меньше порядка поля.

Таблица 1.3. Результаты операций между элементами поля *GF*(2).

Сложение (+)	A	
B	0	1
0	0	1
1	1	0

Умножение (x)	A	
B	0	1
0	0	0
1	0	1

Два поля $F^{(1)}$ и $F^{(2)}$ называются ***изоморфными***, если существует биекция φ: $F^{(1)} \leftarrow F^{(2)}$, сохраняющая операции. Эта биекция и обратная к ней функция φ–1 называются ***изоморфизмами***.

Заметим, что фактор-множество Z_p кольца Z целых чисел по модулю простого числа p является полем порядка p и что все конечные поля простого порядка p являются простыми и изоморфны друг другу, т.е. такие поля составляют класс всех простых конечных полей.

1.9. ОТНОШЕНИЯ СРАВНИМОСТИ. СВОЙСТВА СРАВНЕНИЙ

Возьмем натуральное целое число m, которое будем называть модулем.

Определение 1.14. Целые числа a и b называются сравнимыми по модулю m, если разность $(a - b)$ делится на m $(m \mid a - b)$.

Это соотношение между a, b и m для краткости записывается как $a \equiv b \bmod m$, где a и b будем называть соответственно левой и правой частями сравнения. Из определения следует, что если a делится на m, то $a \equiv 0 \bmod m$.

Теорема 1.10. Число a сравнимо с b тогда и только тогда, когда a и b имеют одинаковые остатки при делении на модуль m.

Доказательство. Пусть $a \equiv b \bmod m$. Представим a и b в виде $a = mq_1 + r_1$, $b = mq_2 + r_2$, где $0 \le r_1 < m$, $0 \le r_2 < m$. Из этих представлений получаем $a - b = m\,(q_1 - q_2) + r_1 - r_2$. Так как $(m \mid a - b)$, то m должно делить $(r_1 - r_2)$. Однако имеет место $-m < r_1 - r_2 < m$. Отсюда следует, что $r_1 - r_2 = 0$, т.е. $r_1 = r_2$. Обратно. Пусть остатки от деления a и b на m равны, т.е. имеет место $a = mq_1 + r$, $b = mq_2 + r$. Тогда $a - b = m\,(q_1 - q_2)$, т.е. $m \mid a - b$.

Свойства сравнений. Из теоремы следует, что целые числа a и b называются сравнимыми по модулю m, если остатки от деления этих чисел на m равны.

Теорема 1.11. Отношения сравнимости рефлексивно, т.е. $a \equiv a \bmod m$.

Доказательство следует из того, что левая и правая часть сравнения имеют равные остатки.

Теорема 1.12. Отношения сравнимости симметрично, т. е. если $a \equiv b \bmod m$, то $b \equiv a \bmod m$.

Доказательство. Если a и b имеют одинаковые остатки при делении на m, то остатки от деления b и a на m также равны.

Теорема 1.13. Отношения сравнимости транзитивно, т. е. если $a \equiv b \bmod m$, $b \equiv c \bmod m$, то $a \equiv c \bmod m$.

Доказательство. Если у чисел a и b, а также у чисел b и c остатки от деления на m равны, то a и c имеют одинаковые остатки при делении на m.

Теорема 1.14. Если $a \equiv b \bmod m$ и k произвольное целое число, то $ka \equiv kb \bmod m$.

Доказательство. Если $a \equiv b \bmod m$, то $m \mid a - b$. Тогда $m \mid k(a - b)$ или $m \mid ka - kb$. Следовательно, $ka \equiv kb \bmod m$.

Теорема 1.15. Если $ka \equiv kb \bmod m$ и НОД $(k, m) = 1$, то $a \equiv b \bmod m$.

Доказательство. Если $ka \equiv kb \bmod m$, то $m \mid ka - kb$, т. е. $m \mid k(a - b)$. Так как по условию выше приведенных теорем наибольший общий делитель равен 1, то из условия $m \mid k(a - b)$ следует, что $m \mid a - b$.

Теорема 1.16. Если $a \equiv b \bmod m$ и k — произвольное целое натуральное число, то $ka \equiv kb \bmod km$.

Доказательство. Если $m \mid a - b$, то $km \mid ka - kb$. Отсюда следует, что $ka \equiv kb \bmod km$.

Теорема 1.17. Если $a \equiv b \bmod m$, $c \equiv d \bmod m$, то $a + c \equiv (b + d) \bmod m$, $a - c \equiv (b - d) \bmod m$.

Доказательство. Если $m \mid a - b$ и $m \mid c - d$, то $m \mid (a - b) \pm (c - d)$. Отсюда получаем $m \mid (a \pm c) - (b \pm d)$.

Теорема 1.18. Если $a \equiv b \bmod m$, $c \equiv d \bmod m$, то $ac \equiv bd \bmod m$.

Доказательство. Если $a \equiv b \bmod m$ и $c \equiv d \bmod m$, то имеем $ac \equiv bd \bmod m$ и $cb \equiv db \bmod m$. Тогда из свойства транзитивности сравнений получим, что $ac \equiv bd \bmod m$.

Теорема 1.19. Если $a \equiv b \bmod m$, то при любом целом $n \neq 0$ имеет место $an \equiv bn \bmod m$.

Доказательство. Применяя n раз теорему 1.19 для сравнений $a \equiv b \bmod m$ и $a \equiv b \bmod m$, получим доказательство.

Пусть n — любое положительное целое число, $p(x)$ — многочлен степени n с целыми коэффициентами. Тогда из сформулированных выше теорем вытекает следующая

Теорема 1.20. Если $a \equiv b \bmod m$, то $p(a) \equiv p(b) \bmod m$.

++++++++++++++++++++++++++

1.10. МОДУЛЯРНАЯ АРИФМЕТИКА

Модулярная арифметика, или арифметика остатков, является основой современной криптографии. Эта арифметика заключается в том, что у любого выражения, в котором используются только целые числа, требуется вычислить не значение этого выражения, а остаток от деления этого значения на некоторое целое число n. Арифметика остатков во многом совпадает с обычной арифметикой. Она коммутативна, ассоциативна и дистрибутивна. Кроме того, вычисление промежуточного результата по модулю n дает такой же результат, что и выполнение всего вычисления с последующим вычислением остатка, т. е. выполняются

$$(a + b) \bmod n \equiv ((a \bmod n) + (b \bmod n)) \bmod n;$$

$$(a - b) \bmod n \equiv ((a \bmod n) - (b \bmod n)) \bmod n;$$

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

$$(c\ (a + b)) \bmod n \equiv (((ca) \bmod n) + ((cb) \bmod n)) \bmod n.$$

Пока при составлении арифметических выражений будем использовать только операции сложения, умножения и вычитания. В дальнейшем мы научимся вычислять обратный элемент, логарифм числа и квадратный корень по модулю n.

Например, вычисление $a^x \bmod n$ в случае, когда показатель степени x является степенью 2, для вычисления $a^x \bmod n$ можно использовать формулу, которая для $x = 16$ имеет следующий вид:

$A^{16} \bmod n = (((a^2 \bmod n)^2 \bmod n)^2 \bmod n)^2 \bmod n$.

Если показатель x не является степенью 2, то, записав этот показатель в двоичной системе счисления, можно представить x в виде суммы степеней 2, что позволяет значительно уменьшить число операций умножений.

1.11. КЛАССЫ

Как уже было отмечено, отношение «быть сравнимыми по модулю m» является отношением эквивалентности на множестве целых чисел Z. Как следствие, последнее разбивается на классы эквивалентных элементов — классы целых чисел, которые принято называть классами вычетов по модулю m.

Определение 1.15. Классом вычетов по модулю m с представителем $a \in Z$ называется множество $\{x \mid x \in Z, x \equiv a \bmod m\}$. Обозна-

чение: $[a]_m$ или просто $[a]$ (при этом модуль m должен быть дополнительно указан или ясен из контекста). Очевидно, любой класс вычетов по модулю m состоит из чисел, попарно сравнимых между собой по модулю m (или, что то же самое, равноостаточных при делении на m). Равенство двух классов вычетов $[a_1] = [a_2]$ по модулю m означает сравнимость их представителей: $a_1 \equiv a_2 \bmod m$. Если r — остаток от деления a на модуль m, то $[a] = [r]$. Поскольку имеется в точности m различных остатков от деления на m (очевидно, попарно не сравнимых по модулю m), то все различные классы вычетов по модулю m таковы:

$$[r] = \{r + mt: t \in Z\}, r \in \{0, 1, ..., m - 1\}. \quad (1.10)$$

Множество всех классов вычетов по модулю m обозначим Z_m. Оно содержит, таким образом, m элементов вида (1.10).

Определение 1.16. Наименьшее неотрицательное число, содержащееся в данном классе вычетов по модулю m, называется наименьшим неотрицательным вычетом этого класса. Абсолютно наименьшим вычетом называется наименьшее по абсолютной величине число из данного класса вычетов.

Для класса вычетов $[a]$ по модулю m наименьшим неотрицательным вычетом будет r — остаток от деления a на m. Абсолютно наименьший вычет этого класса есть

$$\rho = \begin{cases} r, r < \dfrac{m}{2}, \\ -(m - r), r \geq \dfrac{m}{2} \end{cases}$$

(если m четно и $r = m/2$, то абсолютно наименьший вычет принимает два значения: $\rho = \pm\dfrac{m}{2}$).

1.12. ПОЛНАЯ И ПРИВЕДЕННАЯ СИСТЕМЫ ВЫЧЕТОВ

Определение 1.17. Если из каждого класса вычетов по модулю m взять по одному представителю, то возникнет полная система вычетов по модулю m.

Пример 1.10. Числа –14, 8, 16, –4, 18, –2, –1 образуют полную систему вычетов по модулю $m = 7$.

Легко понять, что любая система из m попарно не сравнимых по модулю m чисел будет полной системой вычетов по этому модулю

(действительно, эти числа принадлежат разным классам вычетов, а поскольку чисел имеется столько же, сколько и классов вычетов, то среди них найдется представитель любого класса вычетов по модулю m). Обычно используют полную систему наименьших неотрицательных вычетов по модулю m, т. е. систему возможных остатков от деления на m. Однако в вычислениях предпочтительнее полная система абсолютно наименьших вычетов по модулю m.

Пример 1.11. Найдем возможные остатки от деления квадратов целых чисел на $m = 7$. Очевидно, достаточно найти остатки от деления на 7 чисел вида p^2, где p пробегает полную систему абсолютно наименьших вычетов по модулю 7, т. е. $p \in \{0, \pm1, \pm2, \pm3\}$. Эти остатки таковы: 0, 1, 2, 4.

Часто бывает полезным следующее свойство полных систем вычетов по модулю m.

Теорема 1.21. Пусть $a, b \in Z$, при этом НОД $(a, m) = 1$. Если x пробегает полную систему вычетов по модулю m, то $y = ax + b$ также пробегает полную систему вычетов по модулю m.

Доказательство. Утверждение теоремы является непосредственным следствием такого факта: если $x_1 \not\equiv x_2 \bmod m$, то $y_1 = ax_1 + b \not\equiv ax_2 + b = y_2 \pmod m$. Этот факт можно обосновать рассуждением от противного: иначе мы получили бы сравнение $ax_1 \equiv ax_2 \bmod m$, которое после сокращения на a привело бы к $x_1 \equiv x_2 \bmod m$.

Упражнение 1.1. Пусть НОД $(m_1, m_2) = 1$ и $m = m_1m_2$. Докажите, что числа $x = m_2x^{(1)} + m_1x^{(2)}$, где $x^{(1)}$ пробегает полную систему вычетов по модулю m_1, а $x^{(2)}$ — полную систему вычетов по модулю m_2, образуют полную систему вычетов по модулю m. Указание. Эти числа попарно не сравнимы по модулю m. Из свойства сравнений (см. Отношения сравнимости. Свойства сравнений) следует, что все числа некоторого класса вычетов по модулю m либо все взаимно просты с m, либо все имеют с m общий делитель $d > 1$. Это замечание делает корректным следующее.

Определение 1.18. Класс вычетов $[a]$ по модулю m называется взаимно простым с модулем, если НОД $(a, m) = 1$.

Поскольку $[a] = [r]$, где $r \in \{0, 1, ..., m - 1\}$, количество всех взаимно простых с модулем классов вычетов равно $\varphi(m)$ — значению функции Эйлера от модуля m. Их множество будем обозначать Z_m^*.

Пример 1.12. Имеем $Z_{12}^* = \{[1],[5],[7],[11]\}$.

Определение 1.19. Если из каждого класса вычетов по модулю m, взаимно простого с m, взять по одному представителю, то возникнет приведенная система вычетов по модулю m.

Пример 1.13. Числа –11, 17, 19, 23 образуют приведенную систему вычетов по модулю $m = 12$.

Приведенную систему вычетов по модулю m образует любая система из $\varphi(m)$ попарно не сравнимых по модулю m и взаимно простых с ним чисел.

Теорема 1.22. Пусть НОД $(a, m) = 1$. Если x пробегает приведенную систему вычетов по модулю m, то $y = ax$ также пробегает приведенную систему вычетов по модулю m.

Доказательство. Единственное отличие от доказательства аналогичной теоремы 1.21 состоит в том, что нужно предварительно заметить следующее: если x взаимно просто с m, то $y = ax$ также взаимно просто с m (это свойство взаимно простых чисел).

1.13. ТЕОРЕМА ФЕРМА — ЭЙЛЕРА

Теорему, которую мы хотим доказать, иногда называют малой теоремой Ферма. Она утверждает, что если p простое число и a — любое целое, то p делит $a_p - a$. Частный вариант теоремы известен многие сотни лет, но Ферма, кажется, был первым, кто доказал теорему в полной общности.

Лемма 1.1. Пусть p — положительное простое число, a и b — целые; тогда

$$(a + b)^p \equiv a^p + b^p \bmod p.$$

Доказательство леммы. Воспользуемся формулой бинома Ньютона,

$$(a+b)^p = a^p + b^p + \sum_{i=1}^{p-1} C_p^i a^{p-i} b^i.$$

Имеем для биномиальных коэффициентов при $1 \le i \le p - 1$

$$C_p^i = \frac{p!}{i!(p-i)!} = \frac{p(p-1)(p-2)\cdots(p-i+1)}{i!}.$$

Эти коэффициенты целые p, число p не является делителем числа $i!$. Поэтому сомножитель p из числителя дроби не может сократиться ни с каким сомножителем знаменателя. Значит, знаменатель обязан делить произведение $(p - 1)\ (p - 2)\ \ldots\ (p - i + 1)$. Отсюда C_p^i — число кратное p, что и требовалось доказать; a и b заведомо делятся на p (остаток равен нулю). Это и доказывает лемму.

Теорема 1.23 (Ферма). Пусть p — положительное простое число и a — целое; тогда

$$a^p \equiv a \bmod p. \qquad (1.11)$$

Доказательство. Воспользуемся математической индукцией и предыдущей леммой. Пусть выполняется:

$$P(n) : n^p \equiv n \bmod p$$

для натурального n. Разумеется, $P(1) = 0$ истинное высказывание, поскольку $1^1 = 1$. Таким образом, база индукции выполнена. Для индуктивного перехода от $P(n)$ к $P(n + 1)$ нам нужно выявить связь между этими утверждениями, т.е. показать, что $(n + 1)^p \equiv n + 1 \ mod\ p$. Согласно лемме

$$(n + 1)^p \equiv n^p + 1^p \equiv n^p + 1 \bmod p.$$

По предположению индукции n^p можно заменить на n. Проделав это, мы получаем: $(n + 1)^p \equiv n^p + 1 \equiv n + 1 \bmod p$, что мы и хотели показать.

Воспользуемся доказанной теоремой для упрощения вычислений степеней по модулю p, проблемы, с которой мы уже сталкивались.

Согласно теореме, если p — простое число, а a — целое, то $a^p \equiv a \pmod p$. Предположим теперь, что a не делится на p. Тогда, ввиду простоты p, числа a и p взаимно просты, и из теоремы обратимости следует, что a обратимо по модулю p. Пусть a' — обратный к a элемент. Умножая на a' сравнение $a^p \equiv a \bmod p$, имеем

$$a' \cdot a \cdot a^{p-1} \equiv a' \cdot a \bmod p.$$

Но $a' \cdot a \equiv 1 \bmod p$, и окончательно $a^{p-1} \equiv 1 \bmod p$. Именно этот вариант уравнения теоремы Ферма мы будем использовать в дальнейшем чаще всего. Для будущих ссылок сформулируем его в виде утверждения.

Теорема 1.24 (Ферма). Пусть p — положительное простое число и a — целое, не делящееся на p; тогда

$$a^{p-1} \equiv 1 \bmod p. \qquad (1.12)$$

В 1736 г. Эйлер распространил малую теорему Ферма с простого модуля p на произвольный модуль m.

Теорема 1.25 (Эйлер). Если НОД $(a, m) = 1$, то

$$a^{\varphi(m)} \equiv 1 \bmod p. \qquad (1.13)$$

1.14. АЛГОРИТМ БЫСТРОГО ВОЗВЕДЕНИЯ В СТЕПЕНЬ ПО МОДУЛЮ

Задача, к которой мы хотели бы применить теорему Ферма, звучит следующим образом. Для трех данных натуральных чисел a, k и p таких, что $k > p - 1$, найти вычет a^k по модулю p.

Если p делит a, то вычет равен 0. Поэтому без ограничения общности можно предполагать, что p не делит a. Разделим k на $p - 1$ с остатком: $k = (p - 1)q + r$ где q и r — неотрицательные целые числа, причем $0 \le r \le p - 1$. Следовательно,

$$a^k \equiv a^{(p-1)q+r} \equiv (a^{p-1})^q a^r \bmod p.$$

По теореме Ферма $a^{p-1} \equiv 1 \bmod p$. Значит, $a^k \equiv a^r \bmod p$, и достаточно делать вычисления только для экспонент, показатель которых меньше $p - 1$.

Пример 1.14. Найти вычет числа $2^{5432675}$ по модулю 13.

Решение. Метод предыдущего раздела предписывал вычислить несколько степеней двойки по модулю 13, прежде чем что-нибудь получить. Посмотрим, что будет, если применить теорему Ферма. Сначала найдем остаток от деления 5432675 на 13 – 1 = 12. Он равен 11. Затем, рассуждая, как ранее, мы имеем

$$2^{5432675} \equiv 2^{11} \bmod 13$$

Непосредственный счет дает окончательный ответ: $2^{11} \equiv 7 \bmod 13$.

1.15. СРАВНЕНИЯ ПЕРВОЙ СТЕПЕНИ

В этом подразделе мы рассмотрим один тип сравнений с неизвестными, который допускает сравнительно простое исследование. Речь идет о так называемых линейных сравнениях, или сравнениях первой степени.

Определение 1.20. Сравнение называется линейным, или первой степени, если $\deg f(x_1, \ldots, x_n) = 1$, т. е.

$$f(x_1, \ldots, x_n) = a_1x_1 + \ldots + a_nx_n + b,$$

где все коэффициенты — целые числа, причем среди a_i найдется не кратный m. Мы рассмотрим случай одного неизвестного, а также покажем, как к нему можно свести случай многих неизвестных. Очевидно, сравнение первой степени с одним неизвестным можно записать в виде)

$$ax \equiv b \bmod m, \qquad (1.14)$$

где $a \not\equiv 0 (\bmod m)$.

Пусть сначала НОД $(a, m) = 1$. Тогда сравнение (1.14) имеет в точности одно решение. Это следует, например, из свойства полных систем вычетов по модулю m (1.8). Более того, для представителя r_0 этого единственного решения $[r_0]$ можно дать явную формулу:

$$r_0 = a^{\varphi(m)-1} b.$$

следующую из теоремы Эйлера: $ar_0 = a^{\varphi(m)}b \equiv 1 \cdot b = b \pmod m$.

Однако этой формулой почти не пользуются на практике. Главная причина в том, что показатель этой степени содержит значение функции Эйлера $\varphi(m)$ — величину, которую, вообще говоря, трудно вычислить.

Правильный с практической точки зрения способ найти r_0 основан на алгоритме Евклида и состоит в следующем. Сначала с помощью алгоритма Евклида находим линейное представление $1 = \text{НОД}(a, m)$ (1.8):

$$1 = ax_0 + my_0,$$

где $x_0, y_0 \in Z$, после чего полагаем $r_0 = bx_0$. Проверка: $ar_0 = abx_0 = b - mb_0y_0 \equiv b \bmod m$.

Пример 1.15. Решим сравнение $127x \equiv 13 \bmod 257$. Алгоритм Евклида дает НОД (127, 257) = 1, а также равенство $1 = 127 \cdot 85 + 257 \cdot (-42)$. Тогда $r = 13 \cdot 85 = 1\,105 \equiv 77 \bmod 257$. Таким образом, решением будет класс вычетов [77].

Рассмотрим теперь случай, когда НОД $(a, m) = d > 1$. В этом случае сравнение (1.14) уже может не иметь решений. Действительно, необходимым условием разрешимости является, как нетрудно видеть, делимость b на d, а она не всегда имеет место. Если b все-таки делится на d, то разделим все части сравнения (1.14), включая модуль, на d:

$$a_1x \equiv b_1 \bmod m_1, \qquad (1.15)$$

где $a_1 = \dfrac{a}{d}$, $b_1 = \dfrac{b}{d}$ и $m_1 = \dfrac{m}{d}$, при этом НОД $(a_1, m_1) = 1$. Полученное сравнение (1.15) имеет единственное решение — как класс вычетов $[r_0]_{m1}$ по модулю m_1. Но всякий класс вычетов по модулю m_1 можно представить как объединение d классов вычетов по модулю $m = dm_1$:

$$[r_0]_{m_1} = \bigcup_{i=0}^{d-1} [r_0 + im_1]_m.$$

Таким образом, в рассматриваемой ситуации сравнение (1.15) будет иметь в точности d решений: это классы вычетов

$$[r_0 + im_1]_m,\ i = 0,\ 1,\ \ldots,\ d - 1.$$

При этом если

$$d = ax_0 + my_0, \tag{1.16}$$

где $x_0,\ y_0 \in Z$, то можно взять $r_0 = \dfrac{bx_0}{d}$.

Пример 1.16. Решим сравнение $345x \equiv 39 \pmod{597}$.

Из алгоритма Евклида следует, что

НОД $(345,\ 597) = 3 = 345 \cdot 45 + 597 \cdot (-26)$.

Поскольку 39 кратно 3, сравнение разрешимо и имеет три решения. Сокращая на 3, получим сравнение

$$115x \equiv 13 \bmod 199,$$

единственное решение которого есть $[13 \cdot 45]_{199} = [-12]_{199}$. Следовательно, решениями исходного сравнения будут $[-12]_{597}$, $[187]_{597}$, $[386]_{597}$.

Итак, мы доказали следующую теорему.

Теорема 1.26. Пусть d = НОД $(a,\ m)$. Если b не делится на d, то сравнение (1.14) неразрешимо. В противном случае это сравнение имеет d решений

$$\left[\frac{bx_0 + im}{d}\right],\ i = 0,1,\ldots,d-1;$$

здесь x_0 — коэффициент при a в линейном представлении (1.10).

Покажем теперь, как, пользуясь теоремой 1.26, можно решать сравнения первой степени с несколькими неизвестными.

Пример 1.17. Решим сравнение $48x - 45y + 13 \equiv 0 \bmod 100$.

Запишем это сравнение в виде

$$48x \equiv 45y - 13 \bmod 100. \tag{1.17}$$

Так как НОД $(48,\ 100) = 4$, то должно выполняться сравнение $45y - 13 \equiv \bmod 4$. Решая это сравнение в целых числах, получим

$$y = 1 + 4t_1,$$

где $t_1 \in Z$. Подставив в (1.17) и после сокращения на 4, будем иметь

$$12x \equiv 45t_1 + 8 \bmod 25.$$

Поскольку НОД $(12,\ 25) = 1$, ограничений на t_1 нет. Решая опять в целых числах, найдем

$$x = 9 + 10t_1 + 25t_2,$$

где $t_2 \in Z$. Итак, все решения исходного сравнения в целых числах суть

$$(x,\ y) = (9 + 10t_1 + 25t_2,\ 1 + 4t_1),\ t_1,\ t_2 \in Z.$$

При желании это множество пар целых чисел можно записать в виде пар классов вычетов по модулю 100 (всего их окажется 100, так как каждый из 25 возможных классов для y приведет к 4 классам для x), и мы получим ответ в виде

$$([9 + 10j_1 + 25j_2\},\ [1 + 4j_1]),\ j_1 = 0,\ 1,\ \ldots,\ 24,\ j_2 = 0,\ 1,\ \ldots,\ 4.$$

1.16. ЛИНЕЙНЫЕ ДИОФАНТОВЫЕ УРАВНЕНИЯ

Теория сравнений первой степени оказывается полезной при решении неопределенных уравнений первой степени.

Определение 1.215. Уравнение вида

$$a_1x_1 + \ldots + a_nx_n = b,$$

где $n > 1$ и все коэффициенты — целые числа, называется неопределенным уравнением первой степени.

Предполагается, что неизвестные $x_1,\ \ldots,\ x_n$ могут принимать только целочисленные значения. Алгебраические уравнения

$$f\,(x_1,\ \ldots,\ x_n) = a_1x_1 + \ldots + a_nx_n = b = 0$$

с таким ограничением на неизвестные уравнения принято называть ***диофантовыми*** — по имени Диофанта Александрийского (III в.), автора знаменитого сочинения «Арифметика», в котором содержатся многочисленные примеры решения неопределенных уравнений.

В случае двух неизвестных решение неопределенного уравнения первой степени эквивалентно решению некоторого сравнения первой степени. Действительно, пусть дано неопределенное уравнение

$$Ax + By = C,$$

где коэффициенты A и B отличны от нуля. Рассмотрим сравнение

$$Ax \equiv C \bmod |B|.$$

Любое решение $x_0 \in Z$ этого сравнения приводит к решению $(x_0,\ y_0) \in Z^2$ неопределенного уравнения, при этом

$$y_0 = \frac{C - Ax_0}{B}.$$

Обратно: если (x_0, y_0) — некоторое решение неопределенного уравнения, то число x_0 будет удовлетворять сравнению. Таким образом, решив сравнение, мы найдем и все решения неопределенного уравнения.

Пример 1.18. Решить уравнение $50x - 42y = 34$.

Составим сравнение $50x \equiv 34 \bmod 42$ и, решая его в целых числах, найдем $x = 20 + 21t$, где $t \in Z$. Тогда

$$y = \frac{50x - 34}{42} = \frac{50(20 + 21t) - 34}{42} = 23 + 25t.$$

Итак, все решения данного уравнения — это пары целых чисел вида

$$(x, y) = (20 + 21t, 23 + 25t),\ t \in Z.$$

Последовательно решая подходящим образом составленные сравнения первой степени, можно решать неопределенные уравнения с любым числом неизвестных.

Пример 1.19. Решим уравнение $30x + 24y - 55z = 11$.

Это уравнение эквивалентно сравнению

$$30x \equiv -24y + 11 \pmod{55}. \qquad (1.18)$$

Поскольку НОД $(30, 55) = 5$, последнее разрешимо тогда и только тогда, когда $24y \equiv 11 \pmod 5$. Решив это сравнение, найдем $y = 4 + 5t_1$, где $t_1 \in Z$. Подставив в (1.18), после упрощений получим $6x \equiv -17 - 24t_1 \pmod{11}$. Имеем НОД $(6, 11) = 1$, поэтому никаких ограничений на t_1 не будет. Решая последнее сравнение, мы находим

$$x = -34 - 48t_1 + 11t_2,$$

где $t_2 \in Z$. Осталось найти неизвестное z:

$$z = \frac{30x + 24y - 11}{55} = \frac{30(-34 - 48t_1 + 11t_2) + 24(4 + 5t_1) - 11}{55} =$$
$$= -17 - 24t_1 + 6t_2.$$

Таким образом, все решения данного уравнения — это тройки целых чисел вида

$$(x, y, z) = (-34 - 48t_1 + 11t_2, 4 + 5t_1, -17 - 24t_1 + 6t_2),\ t_1, t_2 \in Z.$$

1.17. КИТАЙСКАЯ ТЕОРЕМА ОБ ОСТАТКАХ

Китайская теорема об остатках позволяет вычислить целое число, если известны его остатки по нескольким простым модулям. Впервые эта теорема была упомянута в трактате китайского математика Сунь Цзы, примерно в III в. до н.э.

Теорема 1.27. Если натуральные числа $m_1, m_2, \ldots, m_n$ попарно взаимно просты, то для любых целых $r_1, r_2, \ldots, r_n$ таких, что $0 \le r_i < m_i$ при всех i, найдется такое число x, которое при делении на m_i дает остаток r_i при всех $1 \le i \le n$. Более того, любые два таких числа x_1 и x_2 удовлетворяют уравнению

$$x_1 \equiv x_2 \bmod m, \text{ где } m = m_1 m_2 \ldots m_n.$$

В трактате китайского математика Джу Шао Квина (Jiushao Qin) (1247 г. н. э.) дается формула для вычисления числа x, удовлетворяющего теореме: $x = \sum_{i=1}^{n} r_i \cdot e_i,\ e_i = \frac{m}{m_i} \cdot \left(\left(\frac{m}{m_i} \right)^{-1} mod\, m_i \right),\ 1 \le i \le n.$

(1.19)

Заметим, что поскольку число m_i взаимно просто с $\frac{m}{m_i}$, то обратное число в формуле для e_i всегда существует для $1 \le i \le n$. Кроме того, имеют место равенства

$$\begin{cases} e_i \cdot e_i \equiv e_i \bmod m, \\ e_i \cdot e_j \equiv 0 \bmod m, i \ne j, \end{cases}$$

т. е. компоненты e_i взаимно ортогональны по модулю m.

Доказательство. Пусть $M_i = \frac{m}{m_i}$ и для каждого $i = 1, \ldots, k$ рассмотрим сравнение

$$M_i x \equiv 1 \bmod m_i.$$

Поскольку

$$\text{НОД } (M_i, m_i) = 1,\ i = 1, \ldots, k,$$

каждое такое сравнение будет иметь единственное решение, которое мы обозначим $\left[M_i^{\cdot} \right]_{m_i}$. Теперь в качестве искомого числа r мы можем взять наименьший неотрицательный вычет числа

$$R = M_1 M_1^{\cdot} r_1 + \ldots + M_k M_k^{\cdot} r_k. \qquad (1.20)$$

по модулю m. В самом деле, для любого $i = 1, \ldots, k$ имеем

$$R \equiv M_i M_i^{\cdot} r_i \equiv r_i \bmod m_i,$$

что и требовалось доказать.

Упражнение 1.2. Найдите все целые числа r, дающие при делении на 3 остаток 2, при делении на 5 — остаток 3, а при делении на 7 — снова остаток 2.

Ответ: $r = 23 + 105t$, $t \in Z$.

Отметим, что числа R вида (1.20) при r_i, пробегающих полные системы вычетов по модулям m_i, $i = 1, \ldots, k$, образуют полную систему вычетов по модулю m.

1.18. ПРОВЕРКА ЧИСЕЛ НА ПРОСТОТУ

Все тесты делятся на детерминированные и вероятностные. ***Детерминированные тесты*** дают определенный ответ, является ли данное число простым или составным. ***Случайные*** (вероятностные) ***тесты*** дают такой же ответ, но с некоторой вероятностью (обычно близкой к 1) того, что он 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. Его сложность оказывается равной $O(\log^{12} n)$, хотя для специальных чисел вида $2p + 1$ сложность будет значительно меньше, а именно: $O(\log^6 n)$. Ввиду значительной сложности этого алгоритма предпочтение, однако, 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, однако вероятность такого события может быть сделана сколь угодно малой. Рассмотрим далее два важнейших примера подобных алгоритмов тестирования чисел на простоту.

Согласно теореме Ферма 1.23, если p простое число и p не делит a, то $a^{p-1} \equiv 1 \bmod p$. Это и положим в основу **теста Ферма**.

Вход: n — натуральное нечетное число, или параметр секретности.

Выход: сообщение о простоте числа.

Шаг 1. Сгенерировать новое случайное число a: $2 \le a_1 \le n - 1$.

Шаг 2. Вычислить $r = a^{n-1} \bmod n$.

Шаг 3. Если $r \ne 1$, то завершить алгоритм, выдать сообщение, что n — составное.

Шаг 4. Если $r = 1$, то перейти к шагу 2 и повторить все то же самое с новым случайным числом a и так далее, вплоть до повторения t шагов. При получении t сравнений $a^{n-1} \equiv 1 \bmod n$, считать n простым числом.

Конец алгоритма

Данный тест приводит к ошибке, когда на всех шагах это условие выполняется, но число n тем не менее является составным.

Пример 1.20. Если $n = 341 = 11 \cdot 31$, имеем $2^{340} \bmod 341 \equiv 1$.

Случай, когда для любых чисел $a_1, a_2, \ldots, a_t$, составное число n проходит тест, является особым. Такие числа n называются ***числами Кармайкла*** при условии, что НОД $(a, n) = 1$. Наименьшее число Кармайкла — это число $n = 561 = 3 \cdot 11 \cdot 17$. Числа Кармайкла встречаются, однако, довольно редко.

При использовании теста Ферма, если число n не является числом Кармайкла, вероятность ошибки тестирования будет равна $\frac{1}{2^t}$, где t — число шагов. Таким образом, выбирая параметр «секретности» t достаточно большим, можно обеспечить высокую надежность тестирования простых чисел.

Тест Миллера — Рабина базируется на следующей теореме теории чисел.

Теорема 1.28. 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 НОД $(a, n) = 1$, тогда: или $a^r = 1 \bmod n$, или $a^{2^j} \cdot r = -1 \bmod n$, где $0 \le j \le s - 1$.

Вход: n — натуральное нечетное число, или параметр секретности.

Выход: сообщение о простоте числа.

Шаг 1. Представить $n - 1$ в виде $2^s \cdot r$, где r — нечетное число.

Шаг 2. Сгенерировать случайное число a, такое что $2 \le a \le n - 1$.

Шаг 3. Вычислить $y = a^r \bmod n$:

а) если $y = \pm 1$, то n прошло тест и возможно является простым. Переходим на шаг 2;

б) если $y \ne \pm 1$, то вычисляются $y^2 \bmod n$, $y^4 \bmod n$, ..., y^{2^j} для $j < s$ до тех пор, пока не получится -1 для некоторого j. Если такое событие происходит, перейти на шаг 2.

Шаг 4. Если ни при каких j не выполняется шаг 3б, то число n — составное и отбрасывается как не прошедшее тест.

Конец алгоритма

Вероятность ошибки при использовании теста Миллера — Рабина аппроксимируется величиной $\frac{1}{4^t}$, что значительно лучше, чем для теста Ферма.

1.19. АЛГОРИТМЫ ГЕНЕРАЦИИ ПРОСТЫХ ЧИСЕЛ. МЕТОД ПРОБНЫХ ДЕЛЕНИЙ. РЕШЕТО ЭРАТОСФЕНА

Метод пробных делений является наиболее простым методом проверки простоты входного составного числа n или нахождения его делителей. Для этого в цикле выполняется пробное деление n на все целые числа от 2 до $\sqrt{n}$.

Процесс поиска можно ускорить, если заранее вычислить произведения простых чисел $s = 1 \cdot 2 \cdot 3 \cdot ...$ (так, чтобы данное произведение было меньше $\sqrt{n}$) и, используя алгоритм Евклида, определить общий делитель d = НОД (n, s).

Решето Эратосфена — старейший из известных способов определения простых чисел. Цель решета — определить все положительные простые числа, меньшие некоторой верхней границы $n > 0$, которую мы предполагаем целой. Чтобы использовать метод решета, поступаем следующим образом. Сначала выписываем все нечетные целые между 3 и n. Причина, по которой мы не трогаем четные числа, заключается в том, что, кроме 2, среди них нет простых чисел.

Теперь мы начинаем просеивать список. Первое число в нем 3. Начиная со следующего числа в списке (это 5), мы вычеркиваем из него каждое третье число. Проделав это до конца, мы вычеркнем все числа из списка, кратные 3 и большие самой 3.

Теперь выберем наименьшее число из списка, превосходящее 3, которое еще не было вычеркнуто. Таким будет 5, а следующее за ним число — 7. Вычеркиваем каждое пятое число из нашего списка, начиная с 7. Таким образом, все числа, кратные 5, будут вычеркнуты. Продолжаем эту процедуру, пока не дойдем до n. Заметим, что если мы собираемся вычеркивать каждое p-е число, то всегда надо начинать отсчет с числа $p + 2$, даже когда это число было вычеркнуто на предыдущих шагах. Например, для $n = 41$ список нечетных чисел выглядит так:

3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41.

Вычеркнув каждое третье число, начиная с 5, мы получим:

3, 5, 7, , 11, 13, , 17, 19, , 23, 25, , 29, 31, , 35, 37, , 41.

Теперь мы вычеркиваем каждое пятое число, начиная с 7, что дает:

3, 5, 7, , 11, 13, , 17, 19, , 23, , , , 29, 31, , , , 37, , , 41.

Следует теперь вычеркнуть каждое седьмое число, начиная с 9. Но если мы это сделаем, то никакие новые числа не отсеются. Далее нам нужно бы вычеркнуть каждое одиннадцатое число, начиная с 13, но это опять не даст никакого эффекта. на самом деле, ни одно из чисел, оставшихся в списке после вычеркивания каждого пятого, не будет позже вычеркнуто ни на каком этапе просеивания. Итак, положительные нечетные простые числа, не превосходящие 41, это

3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41.

В разобранном примере есть пара важных обстоятельств, которые следует учитывать. Во-первых, хотя нам следовало повторять процесс вычеркивания вплоть до граничного числа n (41 в нашем примере), мы избавились от всех составных чисел к тому моменту, когда отсеяли кратные 5, и все остальные просеивания оказались излишними. Во-вторых, некоторые числа вычеркивались больше, чем один раз. Такое произошло, например, с 15. Первый раз оно было вычеркнуто, когда мы отсеивали кратные 3. Но 15 также делится на 5, поэтому оно было вычеркнуто снова при отсеивании кратных 5.

Подобно всем алгоритмам, решето Эратосфена имеет ограничения. Например, оно неэффективно при поиске очень больших простых чисел. Напомним, однако, что цель этого алгоритма — поиск всех простых меньших, чем определенная верхняя граница. К его достоинствам можно отнести простоту программирования и, кроме того, нам не нужно вычислять отдельные делители.

1.20. АЛГОРИТМЫ ФАКТОРИЗАЦИИ. ФАКТОРИЗАЦИЯ ФЕРМА. МЕТОД ПОЛЛАРДА

Алгоритм предыдущего раздела эффективен только в случае, если у числа n, разложение которого мы ищем, есть маленький простой делитель. В настоящем разделе мы изучаем алгоритм, который эффективен, когда у n есть делитель (не обязательно про-

стой), незначительно превосходящий $\sqrt{n}$. Идея алгоритма восходит к Ферма.

Пусть n — нечетное число. Если бы оно было четным, то 2 было бы его делителем. Идея алгоритма состоит в том, чтобы попробовать представить n в виде $n = x^2 - y^2$, где x, y — неотрицательные целые числа. Если такие числа найдены, то

$$n = x^2 - y^2 = (x - y)(x + y).$$

Значит, $x - y$ и $x + y$ являются делителями числа n.

Заметим, если $n = r^2$ для некоторого целого числа r, то r является делителем n. Тогда $x = r$ и $y = 0$. С другой стороны, если $y > 0$, то

$$x = \sqrt{n + y^2} > \sqrt{n}.$$

Факторизация Ферма. Рассмотрим алгоритм Ферма разложения на множители.

Ввод: нечетное натуральное число n.

Вывод: множитель числа n или сообщение о том, что n простое.

Шаг 1. Положить $x = \left[\sqrt{n}\right]$. Если $n = x^2$, то x является делителем числа n, и работа алгоритма останавливается; в противном случае увеличить x на 1 и перейти к шагу 2.

Шаг 2. Если $x = \dfrac{n+1}{2}$, то число n простое, и работа алгоритма останавливается; в противном случае вычислить $y = \sqrt{x^2 - n}$.

Шаг 3. Если число y целое (т. е., если $\sqrt{n}^2 = x^2 - n$), то n раскладывается в произведение $(x - y)(x + y)$, и работа алгоритма останавливается; в противном случае увеличить x на 1 и перейти к шагу 2.

Конец алгоритма.

Пример 1.21. Разложить на множители с помощью алгоритма Ферма число $n = 1\ 342\ 127$.

Решение. Присвоим переменной x целую часть числа $\sqrt{n}$. Она равна $x = 1158$. Выполняется

$$x^2 = (1\ 158)^2 = 1\ 340\ 964 < 1\ 342\ 127.$$

Поэтому мы должны увеличить x на 1. Мы продолжим этот процесс до тех пор, пока число $y = \sqrt{x^2 - n}$ не станет целым или не будет выполняться равенство $x = \dfrac{n+1}{2}$. Заметим, что в нашем

Таблица 1.4. Пример разложения на множители числа n с помощью алгоритма Ферма

Номер цикла	x	$y = \sqrt{x^2 - n}$
1	1 159	33,97...
2	1 160	58,93...
3	1 161	76,11...
4	1 162	90,09...
5	1 163	102,18...
6	1 164	113

примере $\dfrac{n+1}{2} = 671064$. Значения переменных x и y после завершения каждого цикла приведены в табл. 1.4.

Таким образом, на шестом цикле мы получили целое число. Значит, искомые числа $x = 1\ 164$ и $y = 113$. Соответствующие множители равны $x + y = 1\ 277$, $x - y = 1\ 051$.

Метод факторизации Полларда. Метод Полларда, или $(p-1)$-метод, был разработан английским математиком Джоном Полларом в 1974 г.

Пусть n — факторизуемое число, а $1 < p < n$ — его простой делитель. Согласно малой теореме Ферма, для любого a, $1 \le a < p$, выполняется условие $a^{p-1} \equiv 1 \pmod{p}$. Это же сравнение выполнится, если вместо степени $p - 1$ взять произвольное натуральное число M кратное $p - 1$, так как если $M = (p - 1)\,k$, то $a^M = (a^{p-1})^k \equiv \equiv 1^k \equiv 1 \pmod{p}$. Последнее условие эквивалентно $a^M - 1 = pr$ для некоторого целого r. Отсюда, если p является делителем числа n, тогда p является делителем наибольшего общего делителя НОД $(n, a^M - 1)$ и совпадет с НОД $(n, a^M - 1)$, если $a^M - 1 < n$. Пусть

$$p - 1 = p_1^{r_1} p_2^{r_2} \ldots p_n^{r_n}. \qquad (1.21)$$

Идея $(p-1)$-метода состоит в том, чтобы выбрать M в виде произведения как можно большего числа простых сомножителей или их степеней так, чтобы M делилось на каждый сомножитель, $p_i^{r_i}$ входящий в разложение (1.21). Тогда НОД $(n, a^M - 1)$ даст искомый делитель.

1.21. АЛГОРИТМЫ ДИСКРЕТНОГО ЛОГАРИФМИРОВАНИЯ. МЕТОД ПОЛЛАРДА

Пусть G — мультипликативная абелева группа, $a, b \in G$. Задача нахождения решения уравнения

$$a^x = b$$

называется ***задачей дискретного логарифмирования в группе G***. Ее решение x называется дискретным логарифмом элемента b по основанию a и обозначается $\log_a b$, если это решение существует.

Задача дискретного логарифмирования имеет важные приложения в криптографии. Особенно важен случай $G = GF(q)^*$, где $q = p^l$, p — простое число, $l \in N$, а также случай, когда G является группой точек эллиптической кривой над конечным полем. Рассмотрим уравнение

$$a^x \equiv b \pmod{p}. \qquad (1.22)$$

в группе Z_p^*, где p — простое число. Мы будем предполагать, что порядок $a \pmod{p}$ равен $p - 1$. Тогда уравнение разрешимо, и решение x является элементом $Z_{(p-1)}^*$. Мы опишем детерминированные методы решения (1.22). С помощью перебора можно решить уравнение (1.22) за $O(p)$ арифметических операций. Решение $\log_a b$ уравнения (1.22) можно находить по формуле

$$\log_a b \equiv \sum_{j=1}^{p-2} \left(1 - a^j\right)^{-1} b^j \pmod{p-1}.$$

Алгоритм согласования. Сложность вычисления по этой формуле хуже, чем для перебора. Следующий алгоритм решения имеет сложность $O\left(p^{\frac{1}{2}} \log p\right)$ арифметических операций.

Шаг 1. Присвоить $H = \left[p^{\frac{1}{2}}\right] + 1$.

Шаг 2. Найти $c \equiv a^H \pmod{p}$.

Шаг 3. Составить таблицу значений $c^u \pmod{p}$, $1 \le u \le H$, и упорядочить ее.

Шаг 4. Составить таблицу значений $b \cdot a^v \pmod{p}$, $0 \le v \le H$, и упорядочить ее.

Шаг 5. Найти совпавшие элементы из первой и второй таблиц. Для них

$$c^u \equiv b \cdot a^v \pmod{p},$$

откуда $a^{Hu-v} \equiv b \pmod{p}$.

Шаг 6. Выдать $x \equiv Hu - v \pmod{p-1}$.

Конец алгоритма.

В подразд. 1.20 мы описали (p–1)-метод, или метод Полларда для факторизации целых чисел. Аналогичный метод был предложен для дискретного логарифмирования по простому модулю p. Мы хотим решить уравнение $a^x \equiv b \pmod{p}$. Для этого рассмотрим три числовые последовательности $\{u_i\}$, $\{v_i\}$,$\{z_i\}$, $i = 0, 1, 2, \ldots$, определенные следующим образом: $u_0 = v_0 = 0$, $z_0 = 1$;

$$u_{i+1} \equiv \begin{cases} u_i + 1, 0 < z_i < \dfrac{p}{3}; \\ 2u_i, \dfrac{p}{3} < z_i < \dfrac{2}{3}p; \\ u_i, \dfrac{2}{3}p < z_i < p; \end{cases} \bmod(p-1),$$

$$v_{i+1} \equiv \begin{cases} v_i, 0 < z_i < \dfrac{p}{3}; \\ 2v_i, \dfrac{p}{3} < z_i < \dfrac{2}{3}p; \\ v_i + 1, \dfrac{2}{3}p < z_i < p; \end{cases} \bmod(p-1),$$

$$z_{i+1} \equiv b^{u_{i+1}} a^{v_{i+1}} \bmod(p-1).$$

Здесь под $c \bmod p$ мы понимаем наименьший неотрицательный вычет в данном классе вычетов. Далее рассматриваем наборы (z_i, u_i, v_i, z_{2i}, u_{2i}, v_{2i},), $i = 1, 2, 3, \ldots$, и ищем номер i, для которого $z_i = z_{2i}$. Из последнего следует, что

$$b^{u_{2i}-u_i} \equiv a^{v_i - v_{2i}} \pmod{p}.$$

Если $(u_{2i} - u_i, p - 1) = 1$, то при $l \in Z$, $l\,(u_{2i} - u_i) \equiv 1 \pmod{p-1}$ получаем

$$b \equiv a^l (v^i - v^{2i}) \bmod p,$$

откуда искомый x равен $\log_a b \equiv l(v^i - v^{2i}) \pmod{p-1}$.

1.22. АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ НАД БОЛЬШИМИ ЧИСЛАМИ

В данном подразделе мы описываем основные алгоритмы выполнения арифметических операций с большими целыми числами. Числа записаны в двоичной системе счисления, где b — фиксированное натуральное число, $b \geq 2$. При этом натуральное число, записываемое не более чем n цифрами двоичной системе счисления, мы обозначаем $u_1 \ldots u_n$.

Алгоритм сложения неотрицательных целых чисел. Для двух неотрицательных чисел $u_1 \ldots u_n$ и $v_1 \ldots v_n$ вычисляется их сумма $w_0 \ldots w_n$; при этом w_0 — цифра переноса — всегда равна 0 или 1.

Шаг 1. Присвоить $j = n$, $k = 0$ (здесь j идет по разрядам, k следит за переносом).

Шаг 2. Присвоить $w_j = u_j + v_j + k \pmod b$, w_j — наименьший неотрицательный вычет в данном классе вычетов;

$$k = \left[\frac{u_j + v_j + k}{b}\right].$$

(Заметим, что w_j — очередная цифра, k — перенос; всегда $k = 0$ или $k = 1$. При этом, если $b = 2$ или b — размер машинного слова, то для вычисления w_j и k не нужно использовать деления, достаточно взять соответствующий разряд (или разряды) в записи $u_j + v_j + k$.)

Шаг 3. Присвоить $j = j - 1$. Если $j > 0$, то идти на шаг 2; если $j = 0$, то присвоить $w_0 = k$ и закончить работу.

Конец алгоритма.

Обоснование корректности алгоритма очевидно.

Алгоритм вычитания неотрицательных целых чисел. По двуразрядным неотрицательным целым числам $u = u_1 \ldots u_n \geq v = v_1 \ldots v_n \geq 0$ вычисляется их разность $w = w_1 \ldots w_n = u - v$.

Замечание. Для того чтобы в общем случае установить, что $u_1 \ldots u_n \geq v_1 \ldots v_n$, надо пройти по цифрам, вычисляя $u_j - v_j$. Это простая проверка; с ее помощью находится знак разности $u - v$ в общем случае.

Шаг 1. Присвоить $j = n$, $k = 0$ (переменная k — это заем из старшего разряда).

Шаг 2. Присвоить $w_j = u_j - v_j + k \pmod b$ — наименьший неотрицательный вычет в данном классе вычетов;

$$k = \left[\frac{u_j - v_j + k}{b}\right].$$

Шаг 3. $j = j - 1$. Если $j > 0$, то идти на шаг 2; при $j = 0$ закончить работу.

Конец алгоритма.

Обоснование алгоритма достаточно очевидно. При $j = n$ мы находим $w_n = u_n - v_n$, если $u_n \geq v_n$, и $w_n = b + u_n - v_n$, если $u_n < v_n$. Соответственно $k = 0$ и $k = -1$ — это заем из $(n - 1)$-го разряда. Дальнейшие рассуждения аналогичны.

Алгоритм умножения неотрицательных целых чисел столбиком. Опишем метод умножения целых чисел.

Вход: $u = u_1 \ldots u_n$ и $v = v_1 \ldots v_m$ в системе счисления с основанием b.

Выход: $w = uv = w_1 \ldots w_{m+n}$.

Шаг 1. Присвоить $w_{m+1} = 0, \ldots, w_{m+n} = 0$, $j = m$. (Значение j перемещается по номерам разрядов v от младших к старшим.)

Шаг 2. Если $v_j = 0$, то присвоить $w_j = 0$ и перейти на шаг 6.

Шаг 3. Присвоить $j = n$, $k = 0$. (Значение i идет по номерам разрядов числа u, k отвечает за перенос.)

Шаг 4. Присвоить $t = u_i v_j + w_{i+j} + k$, $w_{i+j} := t \pmod b$ — наименьший неотрицательный вычет в данном классе вычетов, $k = \left[\frac{t}{b}\right]$.

(По-прежнему, как и в подразд. 10.1, при вычислении w_{i+j} и k в ряде случаев можно обходиться без деления. Легко показать, что выполняются неравенства $0 \leq t < b^2$, $0 \leq k < b$.)

Шаг 5. $i = i - 1$. Если $i > 0$, то идти на шаг 4. Если $i = 0$, то присвоить $w_j = k$.

Шаг 6. $j = j - 1$. Если $j > 0$, то идти на шаг 2. Если $j = 0$, то закончить работу.

Конец алгоритма.

Опишем методы деления целых чисел многократной точности.

Алгоритм деления на одноразрядное число. Для начала рассмотрим алгоритм деления многоразрядного числа на одноразрядное.

Вход: u — целое число, v — одноразрядное число.

Выход: $w = w_{n-t} \ldots w_0$ — частное, $r = u - vw$ — остаток.

Шаг 1. $r := 0$, $j = 1$.

Шаг 2. $w - j = \left[\frac{rb + u_j}{v}\right]$; $r = rb + u_j \pmod v$ — наименьший неотрицательный вычет в данном классе вычетов.

Шаг 3. $j = j + 1$. Если $j \leq n$, то идти на шаг 2. Если $j > n$, то выдать $w = w_1 \ldots w_n$ и r.

Конец алгоритма.

Корректность алгоритма очевидна — это обычное деление углом.

Алгоритм деления. Теперь рассмотрим простой алгоритм деления многоразрядных целых чисел. Пусть b — основание системы счисления, и $u = u_n \ldots u_1u_0$, $v = v_t \ldots v_1v_0$ — натуральные числа, $n \geq t \geq 1$, $v_t \neq 0$ (случай $t = 0$, т. е. деление на одноразрядное число, рассмотрен выше).

Вход: u, v — два числа, b — основание.

Выход: $q = q_{n-t} \ldots q_0$ — частное, $r = r_t \ldots r_0$, $u = qv + r$, $0 \leq r < v$ — остаток.

Шаг 1. Для j от 0 до $n - t$ присвоить $q_j = 0$.

Шаг 2. До тех пор, пока $u \geq vb^{n-t}$, выполнять следующие действия:

$$q_{n-t} = q_{n-t} + 1,\ u - vb^{n-t}$$

(здесь определяется старшая цифра частного).

Шаг 3. Для $i = n, n - 1, \ldots, t + 1$ выполнять пункты 1—4 этого шага:

1) если $u_i \geq v_t$, то присвоить $q_{i-t-1} = b - 1$, иначе присвоить

$$q_{i-t-1} = \left[\frac{u_ib + u_{i-1}}{v_t}\right];$$

2) до тех пор, пока $q_{i-t-1}(v_tb + v_{t-1}) > u_ib^2 + u_{i-1}b + u_{i-2}$, выполнять $q_{i-t-1} = q_{i-t-1} - 1$;
(заметим, что всегда будет выполняться неравенство $q_{i-t-1} \geq 0$);
3) присвоить $u = u - q_{i-t-1}\, b^{i-t-1}v$;
4) если $u < 0$, то присвоить
$u = u + vb^{i-t-1}$, $q_{i-t-1} = q_{i-t-1} - 1$.
Шаг 4. $r = u$. Выдать q и r.
Конец алгоритма.

1.23. ЭЛЛИПТИЧЕСКИЕ КРИВЫЕ И ИХ ПРИЛОЖЕНИЯ В КРИПТОГРАФИИ

В этом подразделе мы предполагаем, что K — поле: либо поле R вещественных чисел, либо поле Q рациональных чисел, либо поле C комплексных чисел, либо поле F_q из $q = p^r$ элементов. Напомним, что *характеристикой поля* K называется такое натуральное число $p = \text{char } K$, что $p \cdot 1 = 0$, где 1 и 0 — единичный и нулевой элементы K соответственно.

Определение 1.22. Пусть K — поле характеристики, отличной от 2 и 3, и $x^3 + ax + b$ (где $(a, b) \in K$) — кубический многочлен без кратных корней. Эллиптическая кривая над K — это множество точек (x, y) $(x, y \in K)$, удовлетворяющих уравнению

$$y^2 = x^3 + ax + b \qquad (1.23)$$

вместе с единственным элементом, обозначаемым O и называемым «точка в бесконечности» (о ней подробнее будет сказано ниже).

Если K — поле характеристики 2, то эллиптическая кривая над K — это множество точек, удовлетворяющих уравнению либо типа

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

либо типа

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

(здесь кубические многочлены в правых частях могут иметь кратные корни), вместе с «точкой в бесконечности» O.

Если K — поле характеристики 3, то эллиптическая кривая над K — это множество точек, удовлетворяющих уравнению

$$y^2 = x^3 + ax^2 + bx + c,$$

где многочлен справа не имеет кратных корней.
Перед обсуждением конкретных примеров эллиптических кривых над различными полями мы отметим чрезвычайно ***важное свойство точек эллиптической кривой:*** *они образуют абелеву группу относительно операции сложения точек*, о которой будет подробнее сказано ниже. Чтобы объяснить наглядно, как это получается, положим, что $K = R$. т. е. что эллиптическая кривая — обычная плоская кривая (с добавлением еще одной точки O «в бесконечности»).

Определение 1.23. Пусть E — эллиптическая кривая над вещественными числами, и пусть P и Q — две точки на E. Определим точки $-P$ и $P + Q$ по следующим правилам:

1) точка O — тождественный элемент по сложению («нулевой элемент») группы точек. Пусть ни P, ни Q не являются точками в бесконечности;

2) точки P (x, y) и $-P$ имеют одинаковые x-координаты, а их y-координаты различаются только знаком, т. е. $-(x, y) = (x, -y)$. Из (1.23) сразу следует, что $(x, -y)$, — также точка на E;

3) если P и Q имеют различные x-координаты, то прямая $l = PQ$ имеет с E еще в точности одну точку пересечения R (за исключе-

нием двух случаев: когда она оказывается касательной в P, и мы тогда полагаем $R = P$, или касательной в Q, и мы тогда полагаем $R = Q$). Определяем теперь $P + Q$ как точку $-R$, т. е. как отражение от оси x третьей точки пересечения. Геометрическая интерпретация, дающее $P + Q$, приводится в примере 1.22;

4) если $Q = -P$ (т. е. x-координата Q та же, что и у P, а y-координата отличается лишь знаком), то полагаем $P + Q = O$ («точке в бесконечности»; это является следствием правила 1);

5) остается возможность $P = Q$. Тогда считаем, что l — касательная к кривой в точке P. Пусть R — единственная другая точка пересечения l с E. Полагаем $P + Q = -R$ (в качестве R берем P, если касательная прямая в P имеет «двойное касание», т. е. если P есть точка перегиба кривой).

Пример 1.22. на рис. 1.2 слева изображена эллиптическая кривая $y^2 = x^3 - x$ в плоскости xOy и приведен типичный случай сложения точек P и Q. Чтобы найти $P + Q$, проводимпрямую $\overline{PQ}$ и в качестве $P + Q$ берем точку, симметричную относительно оси x третьей точке, определяемой пересечением прямой PQ и кривой. Если бы P совпадала с Q, т. е. если бы нам нужно было найти $2P$, мы использовали бы касательную к кривой в P: тогда точка $2P$ симметрична третьей точке, в которой эта касательная пересекает кривую.

На рис. 1.2 справа аналогичным образом проиллюстрировано сложение точек R и Q на кривой $ly^2 = x^3 + x + 1$.

Теперь мы покажем, почему существует в точности еще одна точка, в которой прямая l, проходящая через P и Q, пересекает кривую; заодно мы выведем формулу для координат этой третьей точки и тем самым — для координат $P + Q$.

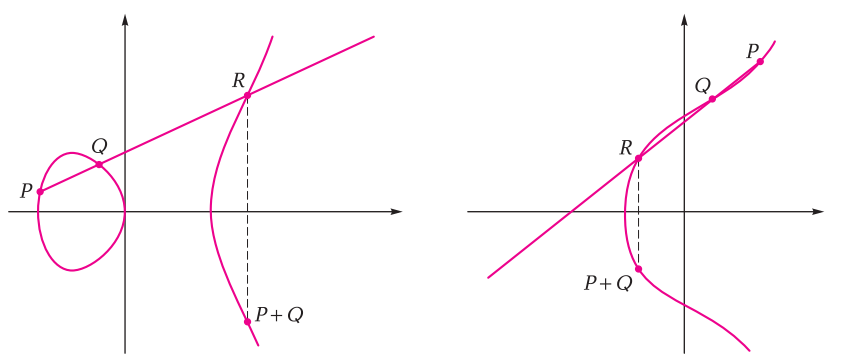

Рис. 1.2. Примеры эллиптических кривых

Обозначим (x_1, y_1), (x_2, y_2), (x_3, y_3) — координаты точек P, Q и $P + Q$ соответственно. Мы хотим выразить x_3 и y_3 через x_1, x_2, y_1, y_2.

Предположим, что мы находимся в ситуации п. 3 операции сложения точек и пусть $y = \alpha x + \beta$ есть уравнение прямой, проходящей через P и Q (в этой ситуации она не вертикальна). Тогда $\alpha = \dfrac{y_2 - y_1}{x_2 - x_1}$, $\beta = y_1 - \alpha x_1$. Точка на l, т. е. точка $(x, \alpha x + \beta)$, лежит на эллиптической кривой тогда и только тогда, когда

$$(\alpha x + \beta)^2 = x^3 + ax + b.$$

Таким образом, каждому корню кубического многочлена $x^3 - (\alpha x + \beta)^2 + ax + b$ соответствует точка пересечения. Имеется два корня: $(x_1, \alpha x_1 + \beta)$ и $(x_2, \alpha x_2 + \beta)$ — точки P и Q на кривой. Так как сумма корней нормированного многочлена равна взятому с обратным знаком коэффициенту при второй степени многочлена, то третий корень — это $x^3 = a^2 - x_1 - x_2$. Тем самым получаем выражение для x_3 и, следовательно, $P + Q = (x_3, \alpha x_3 + \beta)$:

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2, y_3 = -y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x_1 - x_3) \qquad (1.24)$$

Ситуация в п. 5 аналогична рассмотренной, только теперь α — производная $\dfrac{dy}{dx}$ в точке P. Дифференцирование функции, заданной уравнением (1.23), приводит к формуле $\alpha = \dfrac{3x_1^2 + a}{2y_1}$, и мы получаем формулы координат удвоенной точки:

$$x_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1, y_3 = -y_1 + \frac{3x_1^2 + a}{2y_1}(x_1 - x_3) \qquad (1.25)$$

Пример 1.23. Пусть $P = (-3, 9)$ и $Q = (-2, 8)$ — точки на эллиптической кривой $y^2 = x^3 - 36x$. Найти $P + Q$ и 2P.

Решение. Подстановка $x_1 = -3$, $y_1 = 9$, $x_2 = -2$, $y_2 = 8$ в первое из уравнений (1.) дает $x_3 = 6$; тогда из второго уравнения (1.) получаем $y_3 = 0$. Непосредственной подстановкой координат точки $P + Q = (6, 0)$ в уравнение кривой можно убедиться в том, что она также лежит на ней. Подставляя $x_1 = -3$, $y_1 = 9$, $a = -36$ в первое из уравнений (1.25), получаем для x-координаты точки 2P значение $\dfrac{25}{4}$, а второе из уравнений (1.25) дает для y-координаты значение

$-\frac{35}{8}$. Точка $2P = \left(\frac{25}{4}, -\frac{35}{8}\right)$ также принадлежит рассматриваемой кривой.

Если n — целое число, то, как и в любой абелевой группе, nP обозначает сумму n точек P при $n > 0$ и сумму $|n|$ точек –P, если $0 \le n$.

«Точка в бесконечности». По определению «точка в бесконечности» O — это ***нейтральный элемент группового закона***. В графической интерпретации следует себе представлять ее расположенной на оси y в предельном направлении, определяемом все более «крутыми» касательными к кривой. Она является третьей точкой пересечения с кривой для любой вертикальной прямой: такая прямая пересекается с кривой в точках вида (x_1, y_1) и $(x_1, -y_1)$ и в точке O.

ВОПРОСЫ И ЗАДАНИЯ ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ ЗНАНИЙ

1. Что такое взаимно простые числа?
2. Какую задачу решает алгоритм Евклида? Для чего применяется расширенный алгоритм Евклида?
3. Дайте определение функции Эйлера: каковы ее основные свойства?
4. Что такое множества? Какие существуют операции над множествами?
5. Что такое бинарная операция?
6. Дайте определение понятиям «группа», «кольцо», «поле».
7. Что такое отношение сравнимости и какие у него свойства?
8. Что такое модульная арифметика?
9. Что такое класс вычетов по модулю *m*?
10. Дайте определения понятиям «полная и приведенная система» вычетов.
11. В чем заключается теорема Ферма — Эйлера?
12. В чем заключается алгоритм быстрого возведения в степень по модулю?
13. Для чего применяется китайская теорема об остатках?
14. Что такое факторизация?
15. Какую идею для факторизации предложил Ферма?
16. В чем заключается задача дискретного логарифмирования?
17. Что такое эллиптическая кривая и каково ее применение в криптографии?

Глава 2

ВВЕДЕНИЕ В КРИПТОГРАФИЧЕСКУЮ ЗАЩИТУ ИНФОРМАЦИИ

2.1. ОТКРЫТЫЕ СООБЩЕНИЯ И ИХ ХАРАКТЕРИСТИКИ

Для обмена информацией современные люди используют речь и письменность. Информация, передаваемая между людьми, представляет собой сообщения. В основе речи и письменности лежит алфавит, с помощью которого человек отображает сообщение. Различают естественные алфавиты (например: русский, английский) и специальные (например: цифровые, буквенно-цифровые).

Будем понимать под ***сообщением***, с одной стороны, *логически законченную порцию информации* (или текст), имеющую идею, смысл и пригодную для общения людей, а с другой — *совокупность знаков*, отображающих определенным образом сообщение.

Криптография имеет дело с сообщениями, отображаемыми с помощью письменных средств общения какого-либо языка с определенной системой графики и орфографии. Сообщения, с содержанием которых может ознакомиться и понять их смысл любой человек, называются ***открытыми***. С точки зрения криптографии под открытыми сообщениями понимают сообщения (или текст), подлежащие зашифрованию. Открытые сообщения имеют определенные характеристики. Сообщения состоят из букв алфавита (русского, английского и т.д.). Обозначают алфавит следующим образом: например, русский алфавит строчных букв будет иметь следующее обозначение A = {а, б, в, ..., э, ю, я}. Количество знаков в алфавите называется *мощностью алфавита*. Так мощность английского алфавита — 26 знаков, русского — 33 знака. Обозначают мощность алфавита следующим образом: для русского алфавита мощность $|A|=33$. В криптографии в состав алфавита могут входить кроме букв другие знаки: цифры, знаки препинания, специальные знаки и т.д.

В настоящее время наиболее часто используют следующие алфавиты:

1) бинарный, представляющий собой множество $Z_2 = \{0, 1\}$;

2) шестнадцатеричный, представляющий собой множество $Z_{16} = 0, 1, ..., A, D, C, D, E, F\}$;

3) латинский алфавит, представляющий собой множество $Z_{26} = \{A, B, C, D, ..., X, Y, Z\}$;

4) усеченное множество букв кириллицы — $Z_{32} = \{А, Б, В, ...\}$;

5) символы, входящие в стандартные коды ASCII — множество Z_{256}.

В общем случае, если не оговорено иное, будем полагать, что алфавитом сообщений, подлежащих зашифрованию, является множество $Z_m = \{0, 1, ..., m-1\}$, а величину m будем называть мощностью или модулем алфавита исходных текстов.

Открытое сообщение характеризуется длиной. *Длина сообщения* — количество знаков алфавита, входящих в сообщение.

При анализе открытого текста выделяют его *характер*. С точки зрения содержания различают обычные литературные тексты, формализованные данные межмашинного обмена и т. д. Выделяют сообщения по определенной структуре тематики. Тематическое сообщение характеризуется вероятностными словами: «Сообщаю Вам», «Здравствуйте», «С уважением» и т. д.

Открытое сообщение характеризуется *частотой встречаемости знаков* в тексте и элементов сообщения (слов, сочетаний слов и т. д.). Статистика частот встречаемости знаков в тексте и элементов сообщения показывает их неравномерное распределение. В частности, для достаточно большого объема литературного текста на русском языке наиболее часто встречающимися буквами оказываются {о, и}, среди наиболее часто встречающихся букв английского языка — символы {*e*, *t*}. В табл. 2.1 приведены частоты встречаемости букв русского алфавита. Помимо отдельных знаков сообщения можно характеризовать частотой встречаемости буквосочетаний из двух или более знаков. Буквосочетание из двух знаков называется биграммой, из трех знаков — триграммой и т. д. Последовательность k знаков называют k-граммой. Анализируя частоту встречаемости знаков или их сочетаний в криптограммах, можно получить для них открытое сообщение. Этот метод, используемый для дешифрования сообщения, называется ***методом частотного анализа***.

Язык, на котором реализовано сообщение, обладает *избыточностью*. Смысл избыточности языка состоит в том, что не каждое сочетание букв образует слово. Одни буквы и буквенные сочетания употребляются очень часто, другие — гораздо реже; третьи — вообще не употребляются. Все это накладывает на язык множество запретов, и тем самым создается «избыточность» языка, используемая криптоаналитиками для взлома шифров. Лингвисты определили величину избыточности в самых разных языках мира. И везде она колеблется в пределах 70—80 %, т. е. в любом тексте две трети букв определяются не субъективной волей автора, а жесткими правилами грамматики.

Таблица 2.1. Частоты встречаемости букв русского алфавита

Буква, или символ	Частота	Буква	Частота	Буква	Частота	Буква	Частота
пробел	0,145	р	0,041	я	0,019	х	0,009
о	0,095	в	0,039	ы	0,016	ж	0,008
е	0,074	л	0,036	з	0,015	ю	0,007
а	0,064	к	0,029	ъ, ь	0,015	ш	0,006
и	0,064	м	0,026	б	0,015	ц	0,004
т	0,056	д	0,026	г	0,014	щ	0,003
н	0,056	п	0,024	ч	0,013	э	0,003
с	0,047	у	0,021	й	0,010	ф	0,002

Каждое сообщение характеризуется *источником сообщений*. Различают следующие виды источников открытых сообщений:

1) детерминированные источники открытых сообщений;

2) источник передачи данных;

3) вероятностные источники открытых сообщений.

Детерминированный источник открытых сообщений (детерминированная модель источника открытых сообщений) порождает открытые сообщения в виде последовательности символов некоторого алфавита, не содержащей запрещенные сочетания символов, в соответствии с правилами грамматики реализуемого языка. В ряде криптографических задач данная модель источника сообщений используется для различения открытых текстов от случайных последовательностей с помощью вычислительной техники.

Источник передачи данных. Появление систем телеобработки привело к появлению нового вида связи, так называемой передачи данных. Целью передачи данных является передача информа-

ции для обработки ее вычислительным машинам или же выдача ее этими машинами. Принципиальная новизна вида связи — передачи данных — состоит в том, что эта связь осуществляет обмен информацией между компьютерами, а также между компьютерами и человеком. Данные, предназначенные для машин, называют ***формализованным языком*** или языком машин. Эти данные передаются в цифровом виде (часто в виде двоичной последовательности).

Осмысливание их человеком может происходить только после их представления в соответствующей форме. В криптографических терминах понятия формализованного языка представляют собой словарные величины, а их условные формы — *кодообозначения*; последние изображаются в виде буквенных, цифровых и смешанных групп различной длины (разрядности). Формализованный документ оформляется в виде так называемого *формата*, т. е. формы, в которой размещение данных осуществляется по некоторым жестким правилам на местах, определяемых для данного формата шаблоном. Для чтения таких документов необходимо знать формальный язык и форматы документов. Для формализованных сообщений исчезает понятие открытого текста в общепринятом его понимании «читаемого» текста. Признаками «открытого» текста формализованного сообщения являются не его читаемость, а различные его детерминированные и статистические признаки, связанные с применяемыми способами сжатия и кодирования в системах дискретного фототелеграфа, телевидения, телекоммуникационных сетей.

Простейшие вероятностные источники сообщений будем рассматривать как источники случайных последовательностей. Считается, что источник генерирует конечную или бесконечную последовательность случайных символов $x_0, x_1, \ldots, x_{n-1}$ из алфавита I. Вероятность случайного сообщения $(a_0, a_1, \ldots, a_{n-1})$ определяется как вероятность совместного события

$$P(a_0, a_1, \ldots, a_{n-1}) = P(x_0 = a_0, x_1 = a_1, \ldots, x_{n-1} = a_{n-1}).$$

При этом, естественно, требуют выполнения следующих условий:

1) для любого случайного сообщения $(a_0, a_1, \ldots, a_{n-1})$

$$P(a_0, a_1, \ldots, a_{n-1}) \geq 0;$$

$$\sum P(a_0, a_1, \ldots, a_{n-1}) = 1;$$

$$(a_0, a_1, \ldots, a_{n-1})$$

2) для любого случайного сообщения $(a_0, a_1, \ldots, a_{n-1})$

$$P(a_0, a_1, \ldots, a_{n-1}) = \sum_{a_0, a_1, \ldots, a_{s-1}} P(a_0, a_1, \ldots, a_{s-1}), s \geq n.$$

Смысл последнего условия состоит в том, что вероятность всякого случайного сообщения длины n есть сумма вероятностей всех «продолжений» этого сообщения до длины s. Текст, порождаемый таким источником, является вероятностным аналогом языка. Он обладает одинаковыми с языком частотными характеристиками k-грамм. Задавая конкретное вероятностное распределение на множестве открытых текстов, мы задаем соответствующую модель источника сообщений. Рассмотрим часто используемые вероятностные модели источников открытых сообщений.

Среди простейших вероятностных источников сообщений выделяют:

- *стационарные источники независимых символов алфавита, в* которых предполагается, что вероятности сообщений полностью определяются вероятностями отдельных символов алфавита, межзнаковые зависимости в тексте игнорируются. Под открытым текстом понимается реализация последовательности независимых испытаний в полиномиальной вероятностной схеме с числом исходов, равным m. Исходу взаимно-однозначно соответствует символ алфавита I. Эта модель позволяет разделить буквы алфавита на классы высокой, средней и низкой частот использования;
- *стационарный источник независимых биграмм*. Эта модель точнее предыдущей модели отражает свойства языка. Под открытым текстом такого источника понимается реализация последовательности независимых испытаний в полиномиальной вероятностной схеме с числом исходов не более m^2. Множество результатов взаимно-однозначно соответствует множеству всех разрешенных биграмм алфавита;
- *модель независимых биграмм* классифицирует все биграммы источника сообщений по вероятности их появления в тексте. Согласно этой модели всякое сообщение, у которого на четном месте располагается первая буква запретной биграммы, имеет нулевую вероятность. В то же время моделью игнорируются запретные биграммы, у которых первая буква располагается на нечетном месте, а также игнорируются свойственные языкам зависимости между соседними биграммами;
- *стационарный источник марковски зависимых букв*. Открытый текст такого источника является реализацией последовательно-

сти испытаний, связанных простой однородной цепью Маркова с m состояниями. Эта модель учитывает все запретные биграммы (вероятность сообщения, содержащего запретную биграмму, равна нулю), но запретные s-граммы при $s > 2$ учитывает не все.

Рассмотренные стационарные модели можно уточнять и тем самым усложнить в направлении увеличения глубины зависимости вероятности очередной буквы текста от значений нескольких предыдущих букв.

Нестационарные источники открытых сообщений учитывают структуру сообщения, вероятности появления s-грамм в тексте зависят от их места в сообщении. Например, если источником сообщения является премьер-министр, а адресатом — король, то с большой вероятностью сообщение начнется со слов «Ваше Величество! ...», а завершится соответствующей подписью. Подобные стандарты играют важную роль в криптографическом анализе. В частности, удачно выбранная криптоаналитиком нестационарная модель источника открытых сообщений может в некоторых случаях упростить задачу дешифрования по шифрованному тексту, сведя ее к задаче дешифрования по открытому и шифрованному тексту.

Выбор подходящей модели для исследования источника открытых сообщений носит, как правило, компромиссный характер и осуществляется в зависимости от свойств конкретного шифра.

2.2. k-ГРАММНАЯ МОДЕЛЬ ОТКРЫТОГО ТЕКСТА

Для анализа текста, объединяя входящие в него буквы по определенным правилам, можно получать из двух букв — биграммы, из трех букв — триграммы и т. д., т. е. некоторые устойчивые словоформы реального человеческого языка (например: слоги, слова, сочетания слов). В общем виде, когда не указывается конкретное количество объединяемых букв, говорят о k-грамме.

При исследовании математическими методами свойств шифров используют *упрощенную модель открытого текста*. В качестве такой модели выступает k-граммная модель открытого текста. Основанием для использования такой модели открытого текста является устойчивость k-грамм в человеческом языке, а также теоретико-информационный подход, развитый в работах К. Шеннона.

Учет частот k-грамм приводит к следующей модели открытого текста. Пусть $P^{(k)}(A)$ представляет собой массив, состоящий из

приближений для вероятностей $p(b_1b_2...b_k)$ появления k-грамм $b_1b_2...b_k$ в открытом тексте, $k \in N$, N — множество натуральных чисел, $A = \{a_1, ..., a_n\}$ — алфавит открытого текста, $b_1 \in A$, $i = \overline{1,k}$.

Тогда источник открытого текста генерирует последовательность $(c_1, c_2, ..., c_k, c_{k+1}, ...)$ знаков алфавита A, в которой k-грамма $c_1, c_2, ..., c_k$ появляется с вероятностью $p(c_1c_2...c_k) \in P^{(k)}(A)$, следующая k-грамма $(c_2c_3...c_{k+1})$ появляется с вероятностью $p(c_2c_3...c_{k+1}) \in P^{(k)}(A)$, и т. д. Назовем построенную модель открытого текста вероятностной *моделью k-го приближения.*

Таким образом, простейшая модель открытого текста — *вероятностная модель первого приближения* — представляет собой последовательность знаков $c_1, c_2...$, в которой каждый знак c_i, $i = 1, 2, ...$, появляется с вероятностью $p(c_i) \in P^{(1)}(A)$, независимо от других знаков. Будем называть также эту модель *позначной моделью открытого текста*. В такой модели открытый текст $c_1, c_2, ..., c_l$ имеет вероятность

$$p(c_1c_2...c_l) = \prod_{i=1}^{l} p(c_i).$$

В вероятностной модели второго приближения первый знак c_1 имеет вероятность $p(c_1) \in P^{(1)}(A)$, а каждый следующий знак c_i зависит от предыдущего и появляется с вероятностью

$$p(c_i \,/\, c_{i-1}) = \frac{p(c_{i-1}c_i)}{p(c_{i-1})},$$

где $p(c_{i-1}c_i) \in P^{(2)}(A)$, $p(c_{i-1}) \in P^{(1)}(A)$, $i = 2, 3, ...$ Другими словами, модель открытого текста второго приближения представляет собой *простую однородную цепь Маркова*. В такой модели открытый текст $c_1, c_2, ..., c_l$ имеет вероятность

$$p(c_1c_2...c_l) = p(c_1)\prod_{i=2}^{l} p(c_i \,/\, c_{i-1}).$$

Модели открытого текста более высоких приближений учитывают зависимость каждого знака от большего числа предыдущих знаков. Ясно, что чем выше степень приближения, тем более «читаемыми» являются соответствующие модели.

Необходимость использования математических моделей открытого текста вызвана прежде всего следующими соображениями: во-первых, даже при отсутствии ограничений на временные и материальные затраты по выявлению закономерностей, имеющих место в открытых текстах, нельзя гарантировать, что такие

свойства указаны с достаточной полнотой (например, хорошо известно, что частотные свойства текстов в значительной степени зависят от их характера, поэтому при математических исследованиях свойств шифров прибегают к упрощающему моделированию, в частности, реальный открытый текст заменяется его моделью, отражающей наиболее важные его свойства); во-вторых, при автоматизации методов криптоанализа, связанных с перебором ключей, требуется «научить» компьютер отличать открытый текст от случайной последовательности знаков. Ясно, что соответствующий критерий может выявить лишь адекватность последовательности знаков некоторой модели открытого текста.

2.3. КРИТЕРИИ РАСПОЗНАВАНИЯ ОТКРЫТОГО ТЕКСТА

Заменив реальный открытый текст его моделью, можно построить критерий распознавания открытого текста. При этом пользуются либо стандартными методами различения статистических гипотез, либо наличием в открытых текстах некоторых запретов, таких, например, как биграмма ЪЪ в русском тексте. Проиллюстрируем первый подход при распознавании позначной модели открытого текста.

Итак, согласно изложенному выше, открытый текст представляет собой реализацию независимых испытаний случайной величины, значениями которой являются буквы алфавита $A = \{a_1, ..., a_n\}$, появляющиеся в соответствии с распределением вероятностей $P(A) = (p(a_1), ..., p(a_n))$. Требуется определить, является ли случайная последовательность $c_1, c_2, ..., c_l$ букв алфавита A открытым текстом или нет.

Пусть H_0 — гипотеза, состоящая в том, что данная последовательность — открытый текст, H_1 — альтернативная гипотеза. В простейшем случае последовательность $c_1, c_2, ..., c_l$ можно рассматривать при гипотезе H_1 как случайную и равновероятную. Эта альтернатива отвечает субъективному представлению о том, что при расшифровании криптограммы с помощью ложного ключа получается «бессмысленная» последовательность знаков. В более общем случае можно считать, что при гипотезе H_1 последовательность $c_1, c_2, ..., c_l$ представляет собой реализацию независимых испытаний некоторой случайной величины, значениями которой являются буквы алфавита $A = \{a_1, ..., a_n\}$, появляющиеся в соответствии с распределением вероятностей $Q(A) = (q(a_1), ..., q(a_n))$. При

таких договоренностях можно применить, например, *наиболее мощный критерий* различения двух простых гипотез, который дает лемма Неймана — Пирсона.

В силу своего вероятностного характера такой критерий может совершать ошибки двух родов. Критерий может принять открытый текст за случайный набор знаков. Такая ошибка обычно называется *ошибкой первого рода*, ее вероятность равна $\alpha = p\{H_1/H_0\}$. Аналогично вводится *ошибка второго рода* и ее вероятность $\beta = p\{H_0/H_1\}$. Эти ошибки определяют качество работы критерия. В криптографических исследованиях естественно минимизировать вероятность ошибки первого рода, чтобы не «пропустить» открытый текст. Лемма Неймана — Пирсона при заданной вероятности первого рода минимизирует также вероятность ошибки второго рода.

Критерии на открытый текст, использующие запретные сочетания знаков, например, k-граммы подряд идущих букв, будем называть *критериями запретных k-грамм*. Они устроены чрезвычайно просто. Отбирается некоторое число s редких k-грамм, которые объявляются запретными. Теперь, просматривая последовательно k-грамму за k-граммой анализируемой последовательности $c_1, c_2, ..., c_l$, мы объявляем ее случайной, как только в ней встретится одна из запретных k-грамм, и открытым текстом — в противном случае. Такие критерии также могут совершать ошибки в принятии решения. В простейших случаях их можно рассчитать. Несмотря на свою простоту, критерии запретных k-грамм являются весьма эффективными.

2.4. ОСНОВНЫЕ ЗАДАЧИ КРИПТОГРАФИИ

Криптография возникла как наука о методах шифрования, и долгое время именно шифрование (т. е. защита передаваемых или хранимых данных от несанкционированного чтения) оставалась единственной проблемой, изучаемой криптографией. Однако в последнее время, в связи с бурным развитием информационных технологий, возникло множество новых применений, на прямую не связанных с сокрытием секретной информации.

Необходимость применения криптографических методов вытекает из условий, в которых происходит хранение и обмен информацией. В современных информационных системах очень часто происходит обмен данными в коллективах, члены которых не доверяют друг другу. В таких ситуациях необходимы средства, га-

рантирующие, что в процессе обмена или хранения информация не будет подвергнута искажениям или не будет подменена целиком. Такую гарантию может дать только применение научно обоснованных криптографических методов.

В связи с этим основными задачами криптографии является обеспечение:

- ***конфиденциальности данных*** (предотвращение несанкционированного доступа к данным). Это одна из основных задач криптографии, для ее решения применяется шифрование данных, т. е. такое их преобразование, при котором прочитать их могут только законные пользователи, обладающие соответствующим ключом;
- ***целостности данных*** — гарантии того, что при передаче или хранении данные не были модифицированы пользователем, не имеющим на это права. Под модификацией понимается вставка, удаление или подмена информации, а также повторная пересылка перехваченного ранее текста;
- ***аутентификации***. Под аутентификацией понимается проверка подлинности субъектов (сторон при обмене данными, автора документов, и т. д.) или подлинности самой информации. Частным случаем аутентификации является идентификация — процедура доказательства субъектом того, что он действительно является именно тем, за кого себя выдает. Во многих случаях субъект (*X*) должен не просто доказать свои права, но сделать это так, чтобы проверяющий субъект (*Y*) не смог впоследствии сам использовать полученную информацию для того, чтобы выдать себя за *X*. Подобные доказательства называются «доказательствами с нулевым разглашением»;
- ***невозможности отказа от авторства***, предотвращение возможности отказа субъектов от совершенных ими действий (обычно — невозможности отказа от подписи под документом). Эта задача неотделима от другой — обеспечение невозможности приписывания авторства. Наиболее яркий пример ситуации, в которой стоит такая задача, — подписание договора двумя или большим числом лиц, не доверяющих друг другу. В такой ситуации все подписывающие стороны должны быть уверены в том, что в будущем, во-первых, ни один из подписавших не сможет отказаться от своей подписи и, во-вторых, никто не сможет модифицировать, подменить или создать новый документ (договор) и утверждать, что именно этот документ был подписан.

Основным способом решения рассмотренных задач криптографии является *использование электронной подписи*.

Помимо перечисленных основных задач можно назвать также электронное голосование, жеребьевку, разделение секрета (распределение секретной информации между несколькими субъектами таким образом, чтобы воспользоваться ей они могли только все вместе) и многое другое.

2.5. СИММЕТРИЧНОЕ И АСИММЕТРИЧНОЕ ШИФРОВАНИЕ

Будем понимать под шифрованием такое преобразование текста (сообщения), в результате которого прочитать преобразованный текст может только тот, кто обладает специальным ключом.

Процесс шифрования, или ***зашифрования***, включает в себя следующие элементы:

- исходный или открытый текст (сообщение), который обозначим буквой *M*;
- шифрованный текст (сообщение), обозначим его буквой *C*;
- алгоритм шифрования (способ преобразования открытого текста в шифрованный текст), обозначим его буквой *E* (начальная буква английского слова *Encryption* — шифрование). В данном случае будет использоваться криптографический алгоритм преобразования, т. е. алгоритм, зависящий от некоторого параметра, называемого ключом и удовлетворяющий определенным требованиям;
- ключ (важнейший компонент шифрования, определяющий выбор конкретного шифрующего преобразования), который обозначим буквой *k*. Обычно ключ представляет собой буквенную или цифровую последовательность.

Процесс зашифрования можно выразить следующей формулой:

$$C = E_{k1}(M).$$

Преобразование шифрованного текста в открытый текст на основе законно полученного ключа называется ***расшифрованием*** (в отличие от *дешифрования*, которое означает восстановление открытого текста без знания ключа).

Расшифрование включает в себя компоненты, аналогичные рассмотренным выше: шифрованный текст, открытый текст, ключ. Кроме того, расшифрование включает в себя алгоритм расшифрования. Обозначим его буквой *D*, тогда процесс расшифрования можно выразить следующей формулой:

$$M = D_{k2}(C).$$

Алгоритмы зашифрования и расшифрования должны удовлетворять следующему равенству:

$$M = D_{k2}(E_{k1}(M)).$$

Ключи k_1 и k_2 могут иметь одинаковое значение, а в общем случае могут быть разными для алгоритмов расшифрования и зашифрования.

В соответствии со значениями ключей зашифрования и расшифрования в современной криптографии различают симметричное и асимметричное шифрование.

Симметричное шифрование (как для зашифрования, так и для расшифрования) использует либо одинаковые ключи, либо ключи, у которых знание одного из них позволяет легко найти другой.

Асимметричное шифрование осуществляется с помощью двух разных ключей, связанных математически между собой и называемых *криптопарой*. Один из ключей называется *открытым* или *публичным*, другой — *секретным* или *закрытым*. При этом информация, зашифрованная на открытом ключе, может быть расшифрована только с помощью закрытого ключа и наоборот: то, что зашифровано закрытым, можно расшифровать только с помощью открытого ключа. Закрытый ключ владелец хранит в надежном месте, и никто, кроме него, этот ключ не знает, а копия открытого ключа раздается всем желающим. Таким образом, если кто-то захочет обменяться зашифрованными сообщениями с владельцем закрытого ключа, то он зашифровывает сообщение на открытом ключе, который доступен всем желающим, а расшифровать это сообщение можно будет только с помощью закрытого ключа. Как видно, асимметричность проявляется в назначении и использовании ключей зашифрования и расшифрования. Иногда асимметричное шифрование называют *шифрованием с открытым ключом*.

2.6. КЛАССИФИКАЦИЯ ШИФРОВ

Классифицировать шифры можно по различным признакам. Например, *по области применения* различают шифры ограниченного и общего использования, *по стойкости* различают совершенные, практически стойкие и нестойкие шифры. Наиболее часто применяют классификацию по особенностям используемых в шифрах преобразований (или алгоритмов шифров). Классифи-

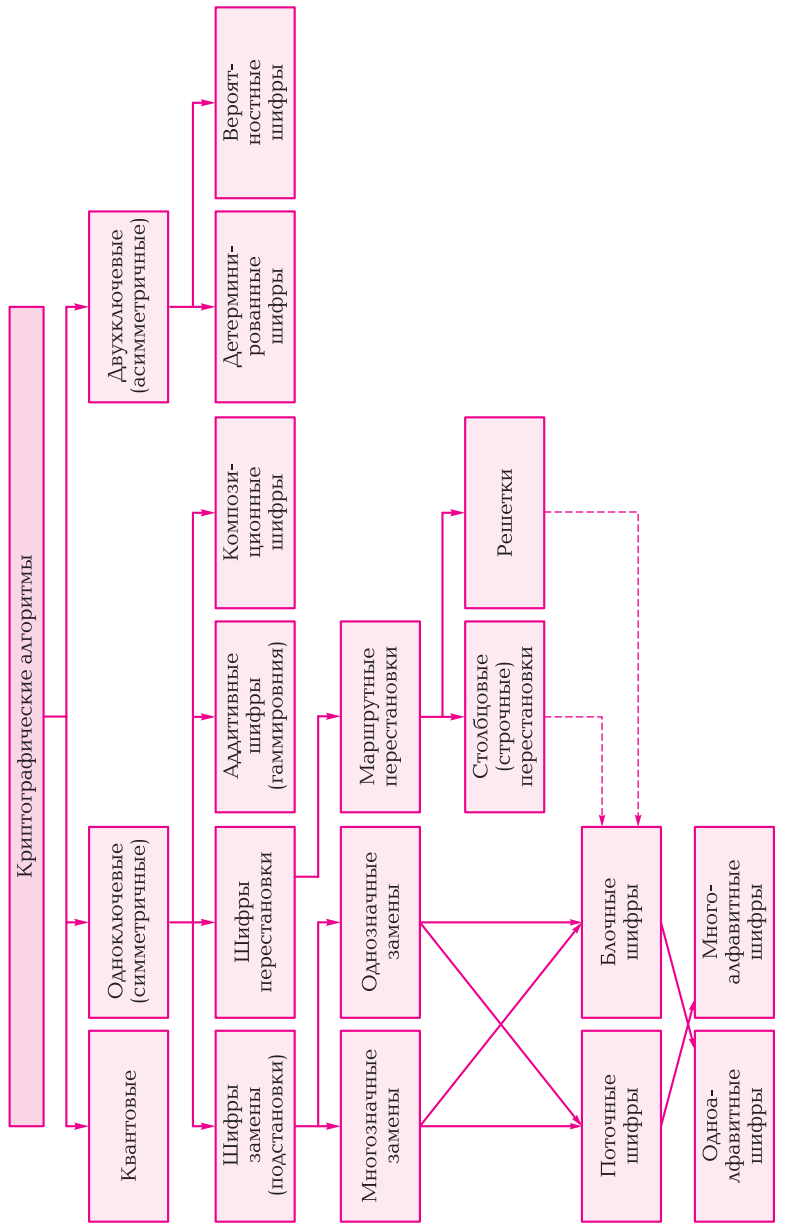

Рис. 2.1. Схема классификации шифров

кация *на основе особенностей алгоритмов шифрования* приведена на рис. 2.1.

Выделим особенности алгоритмов, приведенных в данной классификации шифров.

Одноключевыми, или *симметричными*, *шифрами* называются шифры, в которых для зашифрования и расшифрования используется одинаковый (один и тот же) ключ.

Двухключевыми, или *асимметричными*, *шифрами* называются шифры, в которых для зашифрования и расшифрования используются разные ключи. Один из ключей является открытым, а другой — секретным.

Квантовые шифры основаны на квантово-механическом принципе неопределенности. Процесс отправки и приема информации выполняется посредством объектов квантовой механики (например, с помощью электронов в электрическом токе или фотонов в линиях волоконно-оптической связи). Самым ценным свойством этого вида шифрования является то, что при посылке сообщения отправляющая и принимающая стороны с достаточно большой вероятностью могут установить факт перехвата противником зашифрованного сообщения.

Детерминированные шифры — шифры, в которых каждому тексту открытого сообщения ставится в соответствие ровно один шифрованный текст, т. е. при шифровании одного и того же сообщения одним и тем же ключом всегда будет получаться один и тот же шифротекст.

В *вероятностных шифрах* в процедуре шифрования используется дополнительная случайная величина (число) — в результате при шифровании одного и того же исходного сообщения одним и тем же ключом могут получиться разные шифротексты, которые при расшифровке дадут один и тот же результат (исходное сообщение).

Композиционные шифры построены путем комбинирования относительно простых криптографических преобразований. Например, перестановки и гаммирования, гаммирования и гаммирования, перестановки и замены и т. п. Идея такого подхода заключается в том, что композиция шифров, не являющихся совершенными, может дать шифр, близкий к совершенному шифру.

Шифры замены (*подстановки*) представляют собой шифры, в которых позиции букв в криптограмме остаются теми же, что и у открытого текста, но символы открытого текста заменяются символами другого алфавита.

В *шифрах перестановки* все буквы алфавита открытого текста остаются в криптограмме, но меняют свои позиции в соответствии с определенными правилами. Из шифров перестановки наибольшее распространение получили маршрутные перестановки, основанные на использовании геометрических фигур. Открытый текст записывается в такую фигуру по некоторой траектории. Шифрованный текст получается путем считывания текста по другой траектории. При использовании столбцовых (строчных) перестановок открытый текст вписывается в таблицу по одному маршруту, а шифрованный текст получают путем считывания символов по другому маршруту. Например, открытый текст вписывается в таблицу по столбцам в их естественном порядке, а шифрованный текст получают путем считывания столбцов в порядке, определяемым ключом.

При шифровании открытого текста с помощью решетки, представляющей собой трафарет с отверстиями, в них вписывают либо буквы, либо слоги, либо слова открытого текста. Затем решетку убирают, свободное место заполняют более-менее осмысленным текстом.

В аддитивных шифрах буквы алфавита заменяются числами, к которым затем добавляются числа секретной случайной (псевдослучайной) числовой последовательности (гаммы), после чего берется остаток от деления по модулю (операция mod). Если исходное сообщение и гамма представляются в битовом виде, то при зашифровании и расшифровании применяется логическая операция «исключающее ИЛИ» (XOR, сложение по модулю 2).

По размеру обрабатываемого (шифруемого) блока различают *блочные и поточные* (посимвольные) *шифры*.

При однозначной замене символ открытого текста заменяется одним символом алфавита шифрованного текста. При многозначной замене символ открытого текста может быть заменен одним из нескольких символов алфавита шифрованного текста.

Для одноалфовитных шифров в процессе шифрования используется один алфавит шифрованного текста. Для многоалфавитных шифров в процессе шифрования используется несколько одноалфавитных шифров.

Кроме приведенной шкалы шифры могут быть классифицированы по различным другим критериям, касающимся в основном способов реализации шифраторов, например:

- по виду обрабатываемого сигнала (дискретные и аналоговые);
- по типу засекречиваемых сообщений (для засекречивания телеграфных, речевых, факсимильных сообщений или передачи данных);
- по типу связи межу процессом шифрования и расшифрования (линейного и предварительного шифрования);

- по виду синхронизации между процессом шифрования и расшифрования (автономные, с каналом синхронизации и полуавтономные).

2.7. МОДЕЛИ ШИФРОВ

Первая математическая модель шифра была предложена К. Шенноном. В настоящее время различают две модели шифров: алгебраическую и вероятностную.

Алгебраическая модель шифра. Пусть X, K, Y — конечные множества возможных открытых текстов, ключей и шифрованных текстов соответственно. Обозначим через E множество правил зашифрования, E_k — правило зашифрования на ключе $k \in K$, D — множество правил расшифрования, D_k — правило расшифрования на ключе $k \in K$. Если k представляется в виде $k = (k_з, k_p)$, где $k_з$ — ключ зашифрования, а k_p _ ключ расшифрования, причем $k_з \neq k_p$, то E_k понимается как функция $E_{kз}$, а D_k понимается как функция $D_{kp.}$ Зашифрование открытого текста можно представить формулой

$$y = E_{kз}(x),$$

а для расшифрования справедливо соотношение

$$x = D_{kp}(y),$$

где $x \in X$, $y \in Y$, $(k_з, k_p) \in K$.

Моделью шифра, или шифр-системой, назовем совокупность

$$\sum_A = (X, K, Y, E, D)$$

введенных множеств, для которых выполняются следующие свойства:

1) для любых $x \in X$, $k \in K$ выполняется равенство

$$x = D_k(E_k(x));$$

2) справедливо

$$Y = \bigcup_{kK} E_k(x).$$

Другими словами, модель шифра можно определить как совокупность множеств открытых текстов, возможных ключей, возможных шифрованных текстов, правил зашифрования и правил расшифрования.

Условие 1 означает требование однозначности расшифрования. При этом необходимо учитывать, что одному x может соответствовать несколько y. Условие 2 означает, что любой элемент $y \in Y$ может быть представлен в виде $E_k(x)$ для соответствующих $x \in X$ и $k \in K$. Заметим, что в общем случае утверждение «для любых $k \in K$ и $y \in E_k(X)$ выполняется равенство $E_k(D_k(y)) = y$» будет неверным.

Из условия 1 следует свойство инъективности функции E_k, т. е. если x_1, $x_2 \in X$, $x_1 \neq x_2$, то при любом $k \in K$ выполняется неравенство $E_k(x_1) \neq E_k(x_2)$.

Рассмотрим модель шифра простой замены в алфавите A. Пусть

$$X = Y = \bigcup_{i=1}^{L} A^i \text{ и } K \subseteq S(A),$$

где $S(A)$ — симметрическая группа подстановок множества A и L — натуральное число. X и Y представляют собой объединения декартовых степеней множества A. Для любого ключа $k \in K$, открытого текста $x = (x_1, ..., x_n)$, шифрованного текста $y = (y_1, ..., y_n)$ правила зашифрования и расшифрования шифра простой замены в алфавите A будут определяться формулами:

$$E_k(x) = (k(x_1), ..., k(x_L)),$$

$$D_k(y) = (k^{-1}(y_1), ..., k^{-1}(y_L)),$$

где k^{-1} — подстановка, обратная к k. В общем случае X и Y могут быть разными.

Для шифра перестановки можно использовать следующую алгебраическую модель. Пусть $X = Y = A^L$, $K \subseteq S_L$, где S_L — симметрическая группа подстановок множества $\{1 \ldots L\}$. Можно считать L количеством символов алфавита A в шифруемом блоке. Для любого ключа k, открытого текста $x = (x_1, ..., x_L)$ и шифрованного текста $y = (y_1, ..., y_L)$ правила зашифрования и расшифрования будут иметь вид:

$$E_k(x) = (x_{k(1)}, ..., x_{k(L)}),$$

$$D_k(y) = (y_k^{-1}{}_{(1)}, ..., y_k^{-1}{}_{(L)}),$$

где k^{-1} — подстановка, обратная к k.

Вероятностная модель шифра. Определим априорные распределения вероятностей $P(X)$ и $P(K)$ на множествах X и K соответственно. Тем самым для любого $x \in X$ определена вероятность $p_X(x) \in P(X)$ и для любого $k \in K$ — вероятность $p_K(k) \in P(K)$, для которых выполняются равенства

$$\sum_{xX} p_x(x) = 1 \text{ и } \sum_{kK} p_k(k) = 1.$$

Тогда вероятностная модель шифра выглядит следующим образом:

$$\sum_B = (X, K, Y, E, D, P(X), P(K)).$$

Потребность в математических моделях шифров и открытого текста определяется в первую очередь исследованиями, проводимыми в различных областях криптографии.

2.8. ОСНОВНЫЕ ТРЕБОВАНИЯ К ШИФРАМ

К настоящему времени разработано много различных способов шифрования, но далеко не все можно использовать для защиты информации. Чтобы шифр можно было использовать для защиты информации, он должен удовлетворять следующим основным требованиям:

- зашифрованное сообщение должно поддаваться чтению только при наличии ключа;
- в секрете держится только определенный набор параметров алгоритма (ключ), а остальные детали могут быть открыты без снижения стойкости алгоритма ниже допустимой величины;
- алгоритмы зашифрования и расшифрования должны быть известными;
- число операций, необходимых для расшифровывания информации путем перебора всевозможных ключей, должно иметь строгую нижнюю оценку и выходить за пределы возможностей современных компьютеров (с учетом возможности использования сетевых вычислений);
- число операций, необходимых для определения использованного ключа шифрования по фрагменту шифрованного сообщения и соответствующего ему открытого текста, должно быть не меньше общего числа возможных ключей;
- длина ключа должна быть достаточной, чтобы исключить возможность анализа с помощью перебора всех вариантов ключей;
- любой ключ из множества возможных должен обеспечивать надежную защиту информации;
- незначительное изменение ключа должно приводить к существенному изменению вида зашифрованного сообщения даже при шифровании одного и того же исходного текста;

- незначительное изменение исходного текста должно приводить к существенному изменению вида зашифрованного сообщения даже при использовании одного и того же ключа;
- не должно быть простых и легко устанавливаемых зависимостей между ключами, последовательно используемыми в процессе шифрования;
- зависимость шифрованного текста от открытого текста и от ключа должна быть сложной и неочевидной;
- дополнительные биты, вводимые в сообщение в процессе шифрования, должны быть полностью и надежно скрыты в шифрованном тексте;
- длина блока при блочном шифровании должна быть достаточной для того, чтобы исключить возможность статистического анализа;
- структурные элементы алгоритма шифрования должны быть неизменными;
- длина зашифрованного текста должна быть равной длине исходного текста;
- функции зашифрования и расшифрования должны быть легко вычислимы для любых текстов, если известен ключ/ключи, и трудно вычислимы в ином случае;
- вычисление ключей зашифрования/расшифрования при наличии некоторого набора пар открытый текст — шифрованный текст, а также вычисление ключа зашифрования из ключа расшифрования и наоборот (при атаке на шифр, а не при создании ключей) должно быть трудной вычислительной задачей;
- алгоритм должен допускать как программную, так и аппаратную реализацию, при этом изменение длины ключа не должно вести к качественному ухудшению алгоритма шифрования.

Криптографы, разрабатывающие шифры, обязательно учитывают эти требования.

2.9. КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ

Важную роль в обеспечении защиты информации играют криптографические протоколы.

Будем понимать под ***криптографическим протоколом*** алгоритм, в процессе выполнения которого два участника (или более) последовательно выполняют определенные действия и обмениваются сообщениями посредством использования криптографических средств.

Криптографические протоколы можно классифицировать различными способами, например:

- *по числу участников* различают двусторонний, трехсторонний, многосторонний протоколы;
- *по числу передаваемых сообщений* выделяют интерактивный (есть взаимный обмен сообщениями) и неинтерактивный (только однократная передача) протоколы. Неинтерактивные протоколы часто называют схемами.

Наиболее содержательным является подход, при котором в основе классификации лежит *функциональное* (целевое) *назначение протокола*. Если предположить, что протокол выполняет одну функцию, то получаем следующие типы протоколов:

а) протокол обеспечения целостности сообщений:

- с аутентификацией источника;
- без аутентификации источника;

б) протокол (схема) электронной подписи:

- протокол индивидуальной/групповой электронной подписи;
- с восстановлением / без восстановления сообщения;
- протокол электронной подписи вслепую;
- протокол конфиденциальной электронной подписи;
- протокол электронной подписи с доказуемостью подделки;

в) протокол идентификации (аутентификации участников):

- односторонней аутентификации;
- двусторонней (взаимной) аутентификации;

г) конфиденциальная передача:

- обычный обмен сообщениями;
- широковещательная / циркулярная передача;
- честный обмен секретами;
- забывающая передача;
- протокол привязки к биту (строке);

д) протокол распределения ключей:

- протокол (схема) предварительного распределения ключей;
- протокол передачи ключа (обмена ключами);
- протокол совместной выработки ключа (открытого распределения ключей);
- протокол парный / групповой;
- протокол (схема) разделения секрета;
- протокол (распределения ключей) телеконференции.

Поясним некоторые термины.

Протокол аутентификации сообщений — криптографический протокол, предназначенный для обеспечения целостности сообщений, передаваемых от одного участника к другому. Под целостностью понимается гарантируемая получателю возможность удостовериться, что сообщение поступило от заявленного отправителя и в неискаженном виде.

Протокол (или схема) *идентификации* — протокол аутентификации сторон, участвующих во взаимодействии и не доверяющих друг другу.

Схема электронной подписи в простейшем случае состоит из двух алгоритмов: формирования и проверки подписи. Если требуется скрыть содержание документа от подписывающего участника (подпись вслепую) либо если без участия подписывающего лица процедура проверки подписи невозможна (конфиденциальная электронная подпись), требуются специальные протоколы формирования и проверки подписи.

Задача распределения ключей, необходимых для функционирования криптографической системы, является одной из центральных в криптографии, поскольку в большинстве случаев стойкость криптографической системы основана на недоступности ключей. Поэтому протоколы распределения ключей аккумулируют большинство свойств, предъявляемых к криптографическим протоколам. Они должны обеспечивать не только конфиденциальность, но и аутентификацию всех аспектов взаимодействия (ключа, источника, отправителя и получателя, времени сеанса), чтобы гарантировать, что никакой другой участник, кроме заранее определенного второго участника (и, возможно, других доверенных участников), не мог получить доступа ни к одному секретному ключу.

Протокол обмена секретами — криптографический протокол с двумя участниками, в котором обмен секретами организован таким образом, чтобы в случае его прерывания (по любой причине) знания участников о секретах друг друга были приблизительно одинаковыми.

Протокол привязки к биту — криптографический протокол с двумя участниками (отправителем и получателем), посредством которого отправитель передает получателю бит информации (битовое обязательство) таким образом, что выполняются два условия:

- после передачи бита получателю (так называемого этапа привязки) отправитель уже не может изменить его значение;
- получатель не может самостоятельно определить значение бита и узнает его только после выполнения отправителем так называемого этапа раскрытия.

Протокол подбрасывания (по телефону) *монеты* — криптографический протокол, позволяющий двум не доверяющим друг дру-

гу участникам сгенерировать общий случайный бит. Главное свойство таких протоколов состоит в том, что если хотя бы один из участников честный, то сгенерированный бит будет случайным, независимо от действий другого участника. Имеются обобщения на случай конечных битовых строк, а также на случай произвольного числа участников.

Среди протоколов распределения ключей выделяют так называемые *групповые протоколы*, в которых ключи распределяются между группами участников. Если все группы, имеющие на это право, формируют одинаковые ключи, то такой протокол называют протоколом разделения секрета. Если же у различных групп должны быть разные ключи, то это протокол телеконференции.

В протоколах групповой подписи предполагается одновременное участие заранее определенной группы участников, причем в случае отсутствия хотя бы одного участника группы, формирование подписи невозможно.

Приведенную выше классификацию можно уточнить, если при этом отдельно рассматривать примитивные и прикладные криптографические протоколы.

Примитивный (простой) *криптографический протокол* — криптографический протокол, который не имеет самостоятельного прикладного значения, но используется как базовый компонент при построении прикладных криптографических протоколов. Как правило, он решает какую-либо одну абстрактную задачу, например: протокол обмена секретами, протокол привязки к биту, протокол подбрасывания монеты (по телефону).

Прикладной криптографический протокол предназначен для решения практических задач обеспечения функций — сервисов безопасности с помощью криптографических систем. Следует заметить, что прикладные протоколы, как правило, обеспечивают не одну, а сразу несколько функций безопасности. Более того, такие протоколы, как IPsec, на самом деле, являются большими семействами различных протоколов, включающими много разных вариантов для различных ситуаций и условий применения.

ВОПРОСЫ И ЗАДАНИЯ ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ ЗНАНИЙ

1. Какие существуют виды источников открытых сообщений?
2. Чем вызвана необходимость использования математических моделей открытого текста при исследовании свойств шифра?
3. Назовите основные элементы процесса шифрования (зашифрования).
4. Назовите основные различия симметричного и асимметричного шифрования.
5. В чем отличие одноалфавитного шифра от многоалфавитного?
6. Какие существуют способы классификации криптографических протоколов?
7. Что понимают под атакой на протокол?
8. Дайте определение понятия «аутентификация».
9. Что представляет собой протокол обмена секретами?
10. Назовите основную цель применения криптографических методов.
11. Перечислите основные задачи криптографии.
12. Когда можно применить критерий различения двух простых гипотез, который дает лемма Неймана — Пирсона?
13. Дайте определение криптографического протокола.
14. Какие виды криптографических протоколов знаете?
15. Назовите основные недостатки модели независимых биграмм.
16. Какие различают характеристики открытых сообщений?
17. Перечислите требования к шифрам.
18. Какой протокол называют примитивным?
19. Как работает протокол обмена секретами?
20. Какой протокол называют прикладным?

Глава 3

КОДИРОВАНИЕ, СЖАТИЕ И ШИФРОВАНИЕ ИНФОРМАЦИИ

3.1. СИМВОЛЬНОЕ И СМЫСЛОВОЕ КОДИРОВАНИЕ ИНФОРМАЦИИ

В технической литературе достаточно часто используются два понятия, обозначающие преобразование текста: кодирование и шифрование. Слова «кодирование» и «шифрование» часто используются как синонимы. Однако в современной прикладной математике (к которой можно отнести и криптографию) эти термины разделяются.

В общем случае ***кодированием*** называется любое преобразование данных из одной формы представления в другую.

Под шифрованием будем понимать преобразование данных с использованием криптографических алгоритмов.

Будем понимать под кодом набор условных обозначений для представления информации.

Тогда под кодированием будем понимать процесс представления информации в виде кода.

Процесс кодирования информации заключается в замене смысловых конструкций исходной информации (слов, предложений) кодами.

Кодирование может быть символьным и смысловым.

При ***символьном кодировании*** в качестве кодов могут использоваться буквы, цифры, сочетания букв, цифр, букв и цифр, различные символы, отличные от букв и цифр. Примером символьного кодирования служит *азбука Морзе*.

При ***смысловом кодировании*** и обратном преобразовании используются специальные таблицы или словари. В качестве исходного алфавита используются не только отдельные символы (буквы), но и слова и даже наиболее часто встречающиеся фразы. Существует *одноалфавитное* и *многоалфавитное* смысловое кодирование.

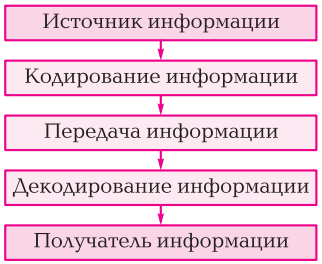

Рис. 3.1. Этапы передачи кодированной информации

Для *обратимости кода* необходимо, чтобы:

1) разным символам входного алфавита *A* были сопоставлены разные кодовые комбинации;

2) никакая кодовая комбинация не составляла начальной части какой-нибудь другой кодовой комбинации.

Процесс передачи информации с помощью кодирования проиллюстрирован на рис. 3.1.

Под ***декодированием*** будем понимать процесс обратный кодированию. В результате декодирования из полученного сообщения восстанавливается первоначальная информация, понятная получателю.

Основными задачами кодирования при передаче информации по каналам связи являются:

- обеспечение экономичности передачи информации посредством устранения избыточности;
- обеспечение надежности (помехоустойчивости) передачи информации;
- согласование скорости передачи информации с пропускной способностью канала.

3.2. ПРЕДСТАВЛЕНИЕ ИНФОРМАЦИИ В ДВОИЧНОМ КОДЕ. ТАБЛИЦА ASCII

Начиная с 60-х гг. прошлого века компьютеры все больше стали использовать для обработки текстовой информации, и в настоящее время бóльшая часть персональных компьютеров (ПК) в мире занята обработкой именно текстовой информации. Современный компьютер может обрабатывать числовую, текстовую, графическую, звуковую и видеоинформацию. Все эти виды информации

в компьютере представлены в двоичном коде, т. е. используется алфавит мощностью два (всего два символа: 0 и 1). Связано это с тем, что удобно представлять информацию в виде последовательности электрических импульсов: импульс отсутствует (0), импульс есть (1). Такое кодирование принято называть двоичным, а сами логические последовательности нулей и единиц — машинным языком.

Традиционно для кодирования одного символа используется количество информации, равное 1 байту (1 байт = 8 битов). Учитывая, что каждый бит принимает значение 1 или 0, получаем, что с помощью 1 байта можно закодировать 256 различных символов ($2^8 = 256$). Кодирование заключается в том, что каждому символу ставиться в соответствие уникальный двоичный код от 00000000 до 11111111 (или десятичный код от 0 до 255). Важно, что присвоение символу конкретного кода — это вопрос соглашения, которое фиксируется кодовой таблицей. ***Таблица кодировки*** — таблица, содержащая упорядоченный некоторым образом перечень кодируемых символов, в соответствии с которой происходит преобразование символа в его двоичный код и обратно.

Таблица ASCII. *Международным стандартом* для ПК стала таблица ASCII (Американский стандартный код для информационного обмена). Таблица кодов ASCII делится на две части. Международным стандартом является лишь первая половина таблицы, т. е. символы с номерами от 0 (00000000) до 127 (01111111). Структура таблицы ASCII приведена в табл. 3.1. В таблице кодировки буквы (прописные и строчные) располагаются в алфавитном порядке, а цифры упорядочены по возрастанию значений. Такое соблюдение лексикографического порядка в расположении символов называется *принципом последовательного кодирования алфавита*.

Таблица 3.1. Структура таблицы ASCII

Порядковый номер	Код	Символ
0 - 31	00000000 — 00011111	Символы с номерами от 0 до 31 принято называть *управляющими*. Их функция — управление процессом вывода текста на экран или печать, подача звукового сигнала, разметка текста и т. п.
32 - 127	00100000 — 01111111	*Стандартная часть таблицы (английская)*. Сюда входят строчные и прописные буквы латинского алфавита, десятичные цифры, знаки препинания, всевозможные скобки, коммерческие и другие символы. Символ 32 — пробел, т. е. пустая позиция в тексте. Все остальные отражаются определенными знаками.

Окончание табл. 3.1

Порядковый номер	Код	Символ
128 - 255	10000000 — 11111111	*Альтернативная часть таблицы (русская)*. Вторая половина кодовой таблицы ASCII, называемая кодовой страницей (128 кодов, начиная с 10000000 и кончая 11111111), может иметь различные варианты, каждый вариант имеет свой номер.

Для букв русского алфавита также соблюдается принцип последовательного кодирования.

3.3. СЖАТИЕ ИНФОРМАЦИИ

Целью сжатия является сокращение объема информации. Но сжатая информация не может быть прочитана или использована без обратного преобразования.

Все методы сжатия информации можно условно разделить на два больших непересекающихся класса: *сжатие с потерей информации* и *сжатие без потери информации*.

Сжатие без потерь применяется в тех случаях, когда информацию нужно восстановить с точностью до бита. Такой подход является единственно возможным при сжатии, например, текстовых данных. В общем случае можно выделить три базовых варианта, на которых строятся ***алгоритмы сжатия***.

Первая группа методов — ***преобразование потока***. Алгоритм предполагает описание новых поступающих несжатых данных через уже обработанные. При этом не вычисляется никаких вероятностей, кодирование символов осуществляется только на основе

тех данных, которые уже были обработаны, как, например, в *LZ*-методах (названных по имени Абрахама Лемпеля и Якоба Зива). В этом случае второе и дальнейшие вхождения некой подстроки, уже известной кодировщику, заменяются ссылками на ее первое вхождение.

Вторая группа методов — это ***статистические методы сжатия***. В свою очередь, эти методы делятся на *адаптивные* (или поточные) и *блочные*.

В первом (адаптивном) варианте вычисление вероятностей для новых данных происходит по данным, уже обработанным при кодировании. К этим методам относятся адаптивные варианты алгоритмов Хаффмана и Шеннона — Фано.

Во втором (блочном) случае статистика каждого блока данных высчитывается отдельно и добавляется к самому сжатому блоку. Сюда можно отнести статистические варианты методов Хаффмана, Шеннона — Фано и арифметического кодирования.

Третья группа методов — так называемые ***методы преобразования блока***. Входящие данные разбиваются на блоки, которые затем трансформируются целиком. При этом некоторые методы (особенно основанные на перестановке блоков) могут не приводить к существенному (или вообще какому-либо) уменьшению объема данных. Однако после подобной обработки структура данных значительно улучшается, и последующее сжатие другими алгоритмами проходит более успешно и быстро.

Все методы сжатия данных основаны на простом логическом принципе. Если представить, что наиболее часто встречающиеся элементы закодированы более короткими кодами, а реже встречающиеся — более длинными, то для хранения всех данных потребуется меньше места, если бы все элементы представлялись кодами одинаковой длины.

При сжатии цифровых данных с регулируемыми потерями обеспечивается экономное представление цифровых данных, но процедура сжатия не является полностью обратимой, т. е. распаковка не позволяет во всех случаях восстановить исходный массив данных до отдельного бита. Как правило, потери допускаются только в той части данных, которая не является существенной при дальнейшем использовании распакованного сообщения. Ясно, что в общем случае «существенность» потерь данных оценить невозможно, поэтому сжатие с потерями возможно только для данных, которые допускают некоторую потерю. Обычно это аналоговые по своей природе данные, например оцифрованные изображения или звук: цифровые фотографии, видеоряды, звукозаписи.

Хотя сжатие с потерями обычно обеспечивает более высокую степень сжатия, чем сжатие без потерь, его нельзя применять к текстовым документам, базам данных и тем более к программному коду, тогда как сжатие без потерь можно применять для любых типов данных.

Методы сжатия реализуются соответствующими алгоритмами. Существуют программные и аппаратные реализации алгоритмов.

Программой сжатия называют программный продукт, предназначенный в первую очередь для компактного представления определенных наборов данных.

Сжатие информации используется также в архиваторах. ***Архиваторами*** называют программы, позволяющие помещать множество файлов в единственный файл (архив).

Аппаратная реализация алгоритма сжатия и распаковки называется кодеком. ***Кодек*** — специальное аппаратно-программное средство, реализующее алгоритм сжатия и распаковки.

3.4. ШИФРЫ ПЕРЕСТАНОВКИ. ТРАДИЦИОННЫЕ ШИФРЫ ПЕРЕСТАНОВКИ

Доподлинно не известно, когда появился шифр перестановки, но вполне возможно, что писцы в древности переставляли буквы в имени своего царя ради того, чтобы скрыть его подлинное имя или в ритуальных целях. Одним из древнейших шифров перестановки является ***шифр Сцитала***. Достоверно известно, что шифр Сцитала использовался в войне Спарты против Афин в конце V в. до н. э. Сцитала представляет собой цилиндрический жезл и узкую полоску пергамента, обматывавшуюся вокруг него по спирали, на которой в свою очередь писалось сообщение. на рис. 3.2 [4, с. 512] представлено устройство Сцитала с намотанным на ней пергаментом.

Шифруемый текст писался на пергаментной ленте по длине жезла. После того как длина жезла оказывалась исчерпанной, он поворачивался и текст писался далее, пока либо не заканчивался текст, либо не исписывалась вся пергаментная лента. В последнем случае использовался очередной кусок пергаментной ленты. Для расшифровки адресат использовал жезл такого же диаметра, на который он наматывал пергамент, чтобы прочитать сообщение. Однако такой шифр может быть легко взломан. Например, метод взлома Сциталы был предложен еще Аристотелем. Он состоит в том, что, не зная точного диаметра жезла, можно использовать

Рис. 3.2. Сцитала

конус, имеющий переменный диаметр и перемещать пергамент с сообщением по его длине до тех пор, пока текст не начнет читаться, — таким образом дешифруется диаметр Сциталы.

Примеры использования криптографии можно встретить в священных иудейских книгах, в том числе в книге пророка Иеремии (VI в. до н. э.), где использовался простой метод шифрования под названием *атбаш*.

Все шифры перестановки делятся на два подкласса:

- шифры одинарной (простой) перестановки. При шифровании символы перемещаются с исходных позиций в новые один раз;
- шифры множественной (сложной) перестановки. При шифровании символы перемещаются с исходных позиций в новые несколько раз.

Шифры одинарной перестановки. В общем случае для данного класса шифров при зашифровании и расшифровании используется таблица перестановок. Пример таблицы перестановок приведен на рис. 3.3.

В первой строке данной таблицы указывается позиция символа в исходном сообщении, а во второй — его позиция в шифрограмме. Таким образом, максимальное количество ключей для шифров перестановки равно $n!$, где n — длина сообщения.

В шифре простой одинарной перестановки для зашифрования и расшифрования используется таблица перестановок, аналогичная показанной на рис. 3.4.

Например, если для шифрования исходного сообщения «АБРАМОВ» использовать таблицу, представленную на рис. 3.4, то шифрограммой будет «РАВБОМА». Для использования на практике такой шифр не удобен, так как при больших значениях n приходится работать с длинными таблицами и для сообщений разной длины необходимо иметь свою таблицу перестановок.

1	2	3	···	n
I_1	I_2	I_3	···	I_n

Рис. 3.3. Таблица перестановок (пример 1)

1	2	3	4	5	6	7
2	4	1	7	6	5	3

Рис. 3.4. Таблица перестановок (пример 2)

1	2	3
2	3	1

Рис. 3.5. Таблица перестановок (пример 3)

Для шифра блочной одинарной перестановки задается таблица перестановки блока символов, которая последовательно применяется до тех пор, пока исходное сообщение не закончится. Если исходное сообщение не кратно размеру блока, тогда оно при шифровании дополняется произвольными символами. Для примера выберем размер блока, равный 3 и примем таблицу перестановок, показанную на рис. 3.5. Дополним исходное сообщение «АБРАМОВ» буквами «Ь» и «Э», чтобы его длина была кратна 3. В результате шифрования получим «РАБОАМЭВЬ».

Количество ключей для данного шифра при фиксированном размере блока равно $m!$, где m — размер блока.

В качестве примера шифра маршрутной перестановки рассмотрим следующий шифр. Зашифруем, например, фразу:

КРИПТОГРАФИЯНАУКАОШИФРАХ

используя прямоугольник размером 3×8. Впишем фразу по горизонтали, начиная с верхнего левого угла поочередно слева направо и справа налево. Запись примет вид, приведенный на рис. 3.6.

Выпишем сообщение по другому маршруту: по вертикали, начиная с правого верхнего угла и двигаясь поочередно сверху вниз и снизу вверх.

Зашифрованная фраза выглядит так:

РАХАФГОИРФЯТПНИШАИРУОАКК

Теоретически маршруты могут быть значительно более изощренными, например, обход конем шахматной доски таким обра-

К	Р	И	П	Т	О	Г	Р
К	У	А	Н	Я	И	Ф	А
А	О	Ш	И	Ф	Р	А	Х

Рис. 3.6. Прямоугольник с записанным в него текстом (пример 1)

5	1	4	3	6	2
К	Р	И	П	Т	О
А	Н	А	Л	И	З
Н	А	У	К	А	О
В	З	Л	О	М	Е
Ш	И	Ф	Р	О	В

Рис. 3.7. Прямоугольник с записанным в него текстом (пример 2)

зом, чтобы в каждой клетке побывать один раз. Один из таких замкнутых маршрутов был найден знаменитым математиком Леонардом Эйлером в 1759 г.

В шифре вертикальной перестановки также используется прямоугольная таблица, в которую сообщение записывается по строкам слева направо. Выписывается шифрованный текст по вертикалям, при этом столбцы выбираются в порядке, определяемом ключом. Зашифруем фразу «Криптоанализ наука о взломе шифров», используя прямоугольник размером 5×6 и ключ (5, 1, 4, 3, 6, 2). Таблица имеет вид, представленный на рис. 3.7.

Выпишем буквы по столбцам в порядке, указанном ключом. В результате получим следующий шифрованный текст:

РНАЗИОЗОЕВПЛКОРИАУЛФКАНВШТИАМО

При расшифровании следует определить число столбцов прямоугольника. Для этого необходимо число букв в сообщении разделить на длину числового ключа. В нашем случае получим пять. Затем разбить текст сообщения на блоки по пять знаков. Буквы шифрованного текста надо поместить на их собственные места в прямоугольнике, и текст сообщения будет прочитан.

3.5. РАЗНОВИДНОСТИ ШИФРОВ ПЕРЕСТАНОВКИ: ОДНО- И ДВУНАПРАВЛЕННЫЕ, ПОТОЧНЫЕ И БЛОЧНЫЕ

Простейший вариант перестановки — прямоугольная таблица с секретным размером столбца показана на рис. 3.8. Открытый текст записывается в таблицу по столбцам, а шифрованный текст считывается по строкам.

Такие шифры перестановки относят к ***однонаправленным шифрам***.

Г	Д	А	К	Е	Р
Р	О	В	У	Р	О
У	С	И	Р	У	М
З	Т	Т	Ь	Т	*

Рис. 3.8. Шифр табличной перестановки

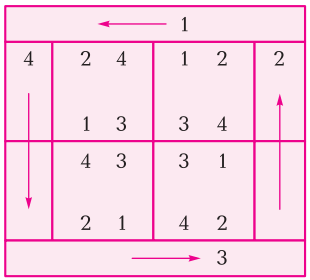

Рис. 3.9. Шифровальный квадрат

Для того, чтобы затруднить процесс дешифрования используют различные усложнения при шифровании. Например, в квадрате (рис. 3.9), состоящем из четырех малых квадратов с определенной нумерацией клеток, вырезают четыре клетки под разными номерами. Квадрат кладется в начальное положение («1» — вверху) и в отверстия (слева направо/сверху вниз) вписываются буквы открытого сообщения.

Затем квадрат поворачивается против часовой стрелки на 90 градусов («2» — вверху) и также вписываются следующие буквы, потом повторяем процесс для положения «3» и «4». Если остаются свободные клетки — они заполняются произвольными символами. Шифрованный текст получают, считав по столбцам или строкам последовательность записанных в прямоугольнике букв. В данном случае используют ***двунаправленное шифрование***.

Рассмотренные способы шифрования являются ***поточными***, т.к. шифруется каждый символ открытого текста.

Суть методов ***блочной перестановки*** заключается в разделении исходного текста на блоки фиксированной длины и последующей перестановке символов внутри каждого блока по определенному алгоритму. Перестановки получаются за счет разницы путей записи исходной информации и путей считывания зашифрованной информации в пределах геометрической фигуры. Примером простейшей двунаправленной перестановки является запись блока исходной информации в матрицу по строкам, а считывание — по столбцам. Последовательность заполнения строк матрицы и считывания зашифрованной информации по столбцам может задаваться ключом.

Криптостойкость метода зависит от длины блока (размерности матрицы). Так, для блока длиной 64 символа (размерность матрицы 8×8) возможны $1{,}6 \times 10^9$ комбинаций ключа. Для блока длиной

256 символов (матрица размерностью 16×16) число возможных ключей достигает $1{,}4 \times 10^{26}$. Решение задачи перебора ключей в последнем случае даже для современных ПК представляет существенную сложность.

Перестановки используются также в методе, основанном на применении ***маршрутов Гамильтона***. Этот метод реализуется путем выполнения следующих шагов.

Шаг 1. Исходная информация разбивается на блоки. Если длина шифруемой информации не кратна длине блока, то на свободные места последнего блока помещаются специальные служебные символы-заполнители (например,*).

Шаг 2. Символами блока заполняется таблица, в которой для каждого порядкового номера символа в блоке отводится вполне определенное место.

Шаг 3. Считывание символов из таблицы осуществляется по одному из маршрутов. Увеличение числа маршрутов повышает криптостойкость шифра. Маршруты выбираются либо последовательно, либо их очередность задается ключом *K*.

Шаг 4. Зашифрованная последовательность символов разбивается на блоки фиксированной длины *L*. Величина *L* может отличаться от длины блоков, на которые разбивается исходная информация на шаге 1.

Расшифрование производится в обратном порядке: в соответствии с ключом выбирается маршрут и заполняется таблица согласно этому маршруту. Из таблицы символы считываются в порядке следования номеров элементов. Ниже приводится пример шифрования информации с использованием маршрутов Гамильтона.

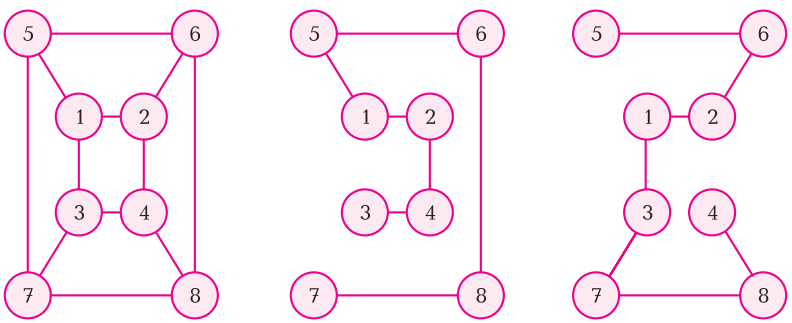

Рис. 3.10. Вариант 8-элементной таблицы и маршрутов Гамильтона

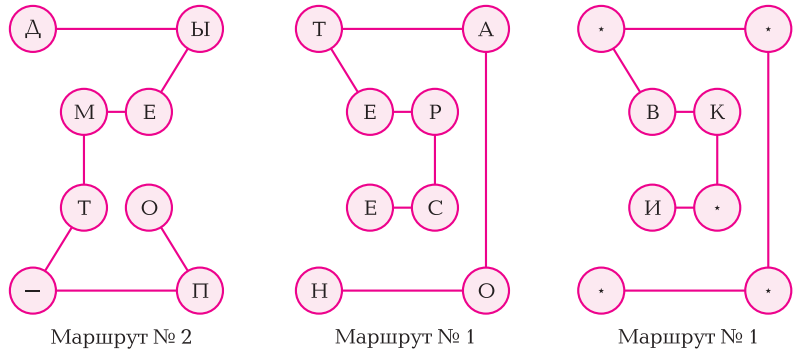

Рис. 3.11. Пример шифрования с помощью маршрутов Гамильтона

П р и м е р. Пусть требуется зашифровать исходный текст T_0 = (МЕТОДЫ_ПЕРЕСТАНОВКИ).

Ключ и длина зашифрованных блоков соответственно равны: K = (2, 1, 1), *L* = 4. Для шифрования используются таблица и два маршрута, представленные на рис. 3.10.

Для заданных условий маршруты с заполненными матрицами имеют вид, показанный на рис. 3.11.

Шаг 1. Исходный текст разбивается на три блока:

Б1 = (МЕТОДЫ_П);

Б2 = (ЕРЕСТАНО);

Б3 = (ВКИ*****).

Шаг 2. Заполняются три матрицы с маршрутами 2, 1, 1 (см. рис. 3.11).

Шаг 3. Получение шифротекста путем расстановки символов в соответствии с маршрутами.

*T*1 = (ОП_ТМЕЫДЕСРЕТАОНИ*КВ****).

Шаг 4. Разбиение на блоки шифротекста

*T*1 = (ОП_Т МЕЫД ЕСРЕ ТАОН И*КВ ****).

На практике большое значение имеет использование специальных аппаратных схем, реализующих метод перестановок.

3.6. ПОТОЧНЫЕ ШИФРЫ ЗАМЕНЫ: ОСНОВНЫЕ ПРИНЦИПЫ ШИФРОВАНИЯ

Поточные шифры замены преобразуют посимвольно открытый текст в шифрованный. Поточный алгоритм зашифровывает *i*-й символ x_i из алфавита *A* в *j*-й символ y_j алфавита *A* шифрован-

ного текста. Ключом такого шифра является подстановка k на множестве A, верхняя строка которой представляет собой естественную последовательность букв алфавита, а нижняя — систематически перемешанную или случайную последовательность букв из A. Например, зашифруем поточным шифром замены слово CIPHER. Пусть подстановка для данного шифра имеет следующий вид:

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
b	e	x	g	w	i	q	v	l	o	u	m	p	j	r	s	t	n	k	h	f	y	z	a	d	c

Результат шифрования будет иметь вид, представленный в табл. 3.2.

В качестве алфавита шифрованного текста можно использовать не только буквы, но и числа, символы и т. д. Рассмотренный пример шифра является шифром простой однобуквенной замены. Одним из первых подобных шифров, известных из истории, был так называемый *шифр Цезаря*, для которого вторая строка является последовательностью, записанной в алфавитном порядке, но начинающейся с третьей буквы алфавита.

Другим примером шифра замены может служить *лозунговый шифр*. Здесь запоминание ключевой последовательности основано на лозунге — легко запоминаемом слове. Например, выберем слово-лозунг «учебник» и заполним вторую строку таблицы по следующему правилу: сначала выписываем слово-лозунг, а затем выписываем в алфавитном порядке буквы алфавита, не вошедшие в слово-лозунг. Вторая строка примет вид

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

Рассмотренные шифры имеют одну слабость. Они не стойки к частотному анализу.

Для повышения стойкости шифра используют ***многоалфавитные подстановки***, в которых для замены символов исходного текста используются символы нескольких алфавитов. Существует несколько разновидностей многоалфавитной подстановки, наиболее известными из которых являются одно- (обыкновенная и монофоническая) и многоконтурная.

Таблица 3.2. Результат шифрования слова CIPHER

Исходное слово	C	I	P	H	E	R
Зашифрованное слово	X	L	S	V	W	N

При ***многоалфавитной одноконтурной обыкновенной подстановке*** для замены символов исходного текста используется несколько алфавитов, причем смена алфавитов осуществляется последовательно циклически, т. е. первый символ заменяется соответствующим символом первого алфавита, второй — символом второго алфавита и т. д. до тех пор, пока не будут использованы все выбранные алфавиты. После этого использование алфавитов повторяется.

Схема шифрования Вижинера. Таблица Вижинера представляет собой квадратную матрицу с n^2 элементами, где n — число символов используемого алфавита. Каждая строка получена циклическим сдвигом алфавита на символ. Для шифрования выбирается буквенный ключ, в соответствии с которым формируется рабочая матрица шифрования.

Шифрование осуществляется по следующему алгоритму: из полной таблицы выбирается первая строка и те строки, первые буквы которых соответствуют буквам ключа. Первой размещается первая строка, а под нею — строки, соответствующие буквам ключа в порядке следования этих букв в ключе. Пример такой рабочей матрицы для ключа Сальери приведен на рис. 3.12.

Процесс шифрования осуществляется следующим образом:

1) под каждой буквой шифруемого текста записываются буквы ключа, при этом ключ повторяется необходимое количество раз;

2) каждая буква шифруемого текста заменяется по рабочей матрице буквами, находящимися на пересечении линий, соединяю-

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

Рис. 3.12. Рабочая матрица для ключа Сальери

щих буквы шифруемого текста в первой строке рабочей матрицы и находящихся под ними букв ключа;

3) полученный текст может разбиваться на группы по несколько знаков.

Пусть, например, нужно зашифровать сообщение: МАКСИМАЛЬНО ДОПУСТИМАЯ ЦЕНА. В соответствии с первым правилом записываем под буквами шифруемого текста буквы ключа. Получаем:

М А К С И М А Л Ь Н О Д О П У С Т И М А Я Ц Е Н А
С А Л Ь Е Р И С А Л Ь Е Р И С А Л Ь Е Р И С А Л Ь

Дальше осуществляется непосредственное шифрование в соответствии со вторым правилом, а именно: берем первую букву шифруемого текста (М) и соответствующую ей букву ключа (С); по букве шифруемого текста (М) входим в рабочую матрицу шифрования и выбираем под ней букву, расположенную в строке, соответствующей букве ключа (С), — в нашем примере такой буквой является Э; выбранную таким образом букву помещаем в шифрованный текст. Эта процедура циклически повторяется до зашифрования всего текста.

В ходе проведения экспериментов выяснилось, что при использовании такого метода статистические характеристики исходного текста практически не проявляются в зашифрованном сообщении. Нетрудно заметить, что замена по таблице Вижинера эквивалентна простой замене с циклическим изменением алфавита, т. е. здесь имеем многоалфавитную подстановку, причем число используемых алфавитов определяется числом букв в слове ключа. Поэтому стойкость такой замены определяется произведением стойкости прямой замены на число используемых алфавитов, т. е. на число букв в ключе.

Расшифровка текста производится в следующей последовательности:

1) над буквами зашифрованного текста последовательно надписываются буквы ключа, причем ключ повторяется необходимое количество раз;

2) в строке рабочей матрицы Вижинера, соответствующей букве ключа, отыскивается буква, соответствующая букве зашифрованного текста. Находящаяся над ней буква первой строки рабочей матрицы и будет буквой исходного текста;

3) полученный текст группируется в слова по смыслу.

Одним из недостатков шифрования по таблице Вижинера является то, что при небольшой длине ключа надежность шифрования остается невысокой, а формирование длинных ключей сопряжено с определенными трудностями.

Нецелесообразно выбирать ключ с повторяющимися буквами, так как при этом стойкость шифра не возрастает. В то же время ключ должен легко запоминаться, чтобы его можно было не записывать. Последовательность букв, не имеющую смысла, запомнить трудно.

С целью повышения стойкости шифрования можно использовать *усовершенствованные варианты таблицы Вижинера*. Приведем некоторые из них:

1) во всех (кроме первой) строках таблицы буквы располагаются в произвольном порядке;

2) в качестве ключа используются случайные последовательности чисел.

Из таблицы Вижинера выбираются десять произвольных строк, которые кодируются натуральными числами от 0 до 10. Эти строки используются в соответствии с чередованием цифр в выбранном ключе. Известны и другие модификации метода.

Частным случаем рассмотренной *многоалфавитной замены* является так называемая ***монофоническая замена***. Особенность этого метода заключается в том, что количество и состав алфавитов выбираются таким образом, чтобы частоты появления всех символов в зашифрованном тексте были одинаковыми. При таком положении затрудняется криптоанализ зашифрованного текста с помощью его статистической обработки. Выравнивание частот появления символов достигается за счет того, что для часто встречающихся символов исходного текста предусматривается использование бóльшего числа заменяющих элементов, чем для редко встречающихся. Пример монофонического шифра для английского алфавита показан на рис. 3.13.

Шифрование осуществляется так же, как и при простой замене, с той лишь разницей, что после шифрования каждого знака соответствующий ему столбец алфавитов циклически сдвигается вверх на одну позицию. Таким образом, столбцы алфавита как бы образуют независимые друг от друга кольца, поворачиваемые вверх на один знак каждый раз после шифрования соответствующего знака.

В качестве примера зашифруем монофоническим шифром следующий текст

INTHISBOOKTHEREADERWILLFIND ...

Зашифрованный текст имеет вид A(-,)VNC/LjpgZ+f.] = hg ...

	Алфавит открытого текста																									
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Алфавит шифротекста	f	N	Q	.	G	т	D	.	A	e	L	.	R	(	C	x	I	Z	V	_	W	S	h	u	K	t
	*	N	Q	b	+	[	D	p	)	e	L	O	R	у	/	x	I	=	$	j	W	S	h	u	K	t
	k	N	Q	i	]	W	D	r	q	e	L	и	R	(	#	x	I	a	d	:	W	S	h	u	K	t
	f	N	Q	.	I	т	D	.	A	e	L	A.	R	у	п	x	I	Z	V	C	W	S	h	u	K	t
	*	N	Q	b	G	[	D	p	)	e	L	O	R	(	C	x	I	=	$	_	W	S	h	u	K	t
	R	N	Q	i	+	W	D	r	q	e	L	и	R	у	/	x	I	a	d	j	W	S	h	u	K	t
	f	N	Q	.	]	т	D	.	A	e	L	A.	R	(	#	x	I	Z	V	:	W	S	h	u	K	t
	*	N	Q	b	I	[	D	p	)	e	L	O	R	у	п	x	I	=	$	C	W	S	h	u	K	t
	k	N	Q	i	G	W	D	r	q	e	L	и	R	(	c	x	I	a	d	_	W	S	h	u	K	t
	f	N	Q	.	+	т	D	.	A	e	L	O	R	у	/	x	I	Z	V	j	W	S	h	u	K	t
	*	N	Q	b	]	[	D	p	)	e	L	A	R	(	#	x	I	=	$	:	W	S	h	u	K	t
	k	N	Q	i	I	W	D	r	q	e	L	и	R	у	п	x	I	a	d	C	W	S	h	u	K	t

Рис. 3.13. Монофонический шифр для английского алфавита

Многоалфавитная многоконтурная замена заключается в том, что для шифрования используется циклически несколько наборов (контуров) алфавитов, причем каждый контур в общем случае имеет свой индивидуальный период применения. Этот период исчисляется, как правило, количеством знаков, после зашифровки которых меняется контур алфавитов. Частным случаем многоконтурной многоалфавитной подстановки является замена по таблице Вижинера, если для шифрования используется несколько ключей, каждый из которых имеет свой период применения.

Общий принцип шифрования подстановкой может быть представлен следующей формулой:

$$y_i = x_i + \beta \bmod (n-1),$$

где y_i — символ зашифрованного текста; x_i — символ исходного текста; β — целое число в диапазоне $0 \div (n-1)$; n — число символов используемого алфавита.

Если β фиксировано, то формула описывает одноалфавитную подстановку, если β выбирается из последовательности $\beta_1, \beta_2, \ldots, \beta_m$, то получается многоалфавитная подстановка с периодом m.

Если в многоалфавитной подстановке $n > k$ (где k — число знаков шифруемого текста) и любая последовательность $\beta_1, \beta_2, \ldots, \beta_m$ используется только один раз, то такой шифр является теоретически не раскрываемым, если, конечно, злоумышленник не имеет доступа к исходному тексту. Такой шифр получил название *шифра Вернама*.

Традиционно шифры замены строились по принципу поточного шифрования, и в качестве шифруемых символов использовались буквы или биграммы. В электронных поточных системах в качестве шифруемых символов чаще всего фигурируют биты или байты.

3.7. ПРИМЕНЕНИЕ ГЕНЕРАТОРОВ ПСЕВДОСЛУЧАЙНЫХ ЧИСЕЛ В КРИПТОГРАФИИ

Многие прикладные задачи криптографии требуют случайных чисел:

- генерация ключей;
- одноразовые случайные числа;
- одноразовые шифроблокноты;
- соль* в схемах электронной подписи.

Криптографически стойкий генератор псевдослучайных чисел (КСГПСЧ) — это генератор псевдослучайных чисел с определенными свойствами, позволяющими использовать его в криптографии.

Требования к КСГПСЧ можно разделить на две группы: во-первых, они должны проходить статистические тесты на случайность; а во-вторых, они должны сохранять непредсказуемость, даже если часть их исходного или текущего состояния становится известна криптоаналитику.

Рассмотрим три класса реализации КСГПСЧ:

1) на основе криптографических алгоритмов;
2) на основе вычислительно сложных математических задач;
3) специальные реализации.

Возможны следующие реализации на основе криптографических алгоритмов:

- безопасный блочный шифр можно преобразовать в КСГПСЧ, запустив его в режиме счетчика. Для этого, выбрав случайный ключ, можно получать следующий случайный блок, применяя алгоритм к последовательным натуральным числам. Счет можно начинать с произвольного натурального числа. Очевидно, что периодом будет не больше чем $2n$ чисел для n-битного блочного шифра. Безопасность такой схемы полностью зависит от секретности ключа;

* *Соль*, или *модификатор*, — строка данных, которая передается хеш-функции вместе с паролем.

- криптографически стойкая хеш-функция также может быть преобразована в КСГПСЧ. В таком случае исходное значение счетчика должно оставаться в секрете;
- большинство потоковых шифров работает на основе генерации псевдослучайного потока бит, которые некоторым образом комбинируются (почти всегда с помощью операции XOR) с битами открытого текста. Запуск такого шифра на последовательности натуральных чисел даст новую псевдослучайную последовательность, возможно, даже с более длинным периодом. Такой метод безопасен, только если в самом потоковом шифре используется надежный КСГПСЧ (что не всегда так). Заметим: начальное состояние счетчика должно оставаться секретным.

При реализации на основе математических задач можно использовать следующие алгоритмы:

- *алгоритм Блюма — Блюма — Шуба*. Данный алгоритм имеет высокую криптостойкость, основанную на предполагаемой сложности факторизации целых чисел. Однако, этот алгоритм отличается очень медленной работой;
- *алгоритм Блюма — Микали*. Данный алгоритм основан на задаче дискретного логарифмирования.

Специальные реализации включают в себя целый ряд практически используемых генераторов псевдослучайных чисел (ПСЧ), которые разрабатывались с учетом криптографической стойкости, например:

- *алгоритм Ярроу*, который пытается определить энтропию входных данных. Он используется в следующих операционных системах: FreeBSD, Open BSD и Mac OSX;
- *алгоритм Fortuna*, разработанный на основе предыдущего алгоритма;
- *специальный файл* ОС UNIX/dev/random, в частности, /dev/urandom, реализованный в Linux;
- *функция CryptGenRandom*, представленная в CryptoAPI компании Microsoft.

3.8. МЕТОДЫ ПОЛУЧЕНИЯ ПСЕВДОСЛУЧАЙНЫХ ПОСЛЕДОВАТЕЛЬНОСТЕЙ: ЛКГ, МЕТОД ФИБОНАЧЧИ, МЕТОД BBS

Для ряда криптографических преобразований используют случайные первичные состояния либо целые последовательности. Стойкость криптоалгоритма, использующего такие состояния или последовательности, напрямую зависит от алгоритма генерации случайных чисел и последовательностей, точнее, от их степени случайности.

Последовательность называется ***случайной***, если воспроизвести ее, зная алгоритм и все исходные данные, не представляется возможным (иначе говоря: дважды запустив генератор в тех же условиях, получим разные последовательности). Но компьютерные системы детерминированы, т. е. они могут находиться лишь в конечном количестве состояний. Это приводит к тому, что генерируемые ими последовательности будут периодичны — такие последовательности называются ***псевдослучайными***.

Последовательность называется ***криптографически надежной псевдослучайной*** последовательностью, если вычислительно неосуществимо предсказать следующий бит, имея полное знание алгоритма и аппаратуры и всех предшествующих битов потока.

Кроме того, случайные и псевдослучайные последовательности, используемые в криптографических преобразованиях, должны подчиняться ***закону равномерного распределения***. Примером такого распределения может служить следующая последовательность: время, остающееся до начала движения поезда в метро, с момента спуска под землю (с учетом того, что поезд ходит через равные интервалы времени, а спускаемся в метро мы в случайный дискретный момент времени). Реализовать такую последовательность математически можно, используя *конгруэнтные генераторы псевдослучайных чисел*.

Линейные конгруэнтные генераторы (ЛГК). Генераторы ПСЧ могут работать по разным алгоритмам. Одним из простейших генераторов является так называемый *линейный конгруэнтный генератор*, который для вычисления очередного числа k_i использует формулу

$$k_i = (ak_{i-1} + b) \bmod c,$$

где a, b, с — некоторые константы, а k_{i-1} — предыдущее псевдослучайное число. Для получения k_1 задается начальное значение k_0. Возьмем в качестве примера $a = 5$, $b = 3$, $c = 11$ и пусть $k_0 = 1$. В этом случае мы сможем по приведенной выше формуле получать значения от 0 до 10 (так как с = 11). Вычислим несколько элементов последовательности:

$$k_1 = (5 \cdot 1 + 3) \bmod 11 = 8;$$
$$k_2 = (5 \cdot 8 + 3) \bmod 11 = 10;$$
$$k_3 = (5 \cdot 10 + 3) \bmod 11 = 9;$$
$$k_4 = (5 \cdot 9 + 3) \bmod 11 = 4;$$
$$k_5 = (5 \cdot 4 + 3) \bmod 11 = 1.$$

Полученные значения (8, 10, 9, 4, 1) выглядят похожими на случайные числа. Однако следующее значение k_6 будет снова равно 8:

$$k_6 = (5 \cdot 1 + 3) \bmod 11 = 8,$$

а значения k_7 и k_8 будут равны 10 и 9 соответственно:

$$k_7 = (5 \cdot 8 + 3) \bmod 11 = 10;$$

$$k_8 = (5 \cdot 10 + 3) \bmod 11 = 9.$$

Данный генератор псевдослучайных чисел повторяется, порождая периодически числа 8, 10, 9, 4, 1. К сожалению, это свойство характерно для всех линейных конгруэнтных генераторов. Изменяя значения основных параметров a, b и c, можно влиять на длину периода и на сами порождаемые значения k_i. Так, например, увеличение числа с в общем случае ведет к увеличению периода. Если параметры a, b и c выбраны правильно, то генератор будет порождать случайные числа с максимальным периодом, равным c. При программной реализации значение с обычно устанавливается равным 2^{b-1} или 2^b, где b — длина слова ПК в битах.

Достоинством линейных конгруэнтных генераторов псевдослучайных чисел является их простота и высокая скорость получения псевдослучайных значений. Линейные конгруэнтные генераторы находят применение при решении задач моделирования и математической статистики, однако в криптографических целях их нельзя рекомендовать к использованию, так как специалисты по криптоанализу научились восстанавливать всю последовательность псевдослучайных чисел по нескольким значениям.

Метод Фибоначчи с запаздыванием. Один из методов генерации псевдослучайных чисел — метод Фибоначчи с запаздыванием (Lagged Fibonacci Generator). Он позволяет получить более высокое «качество» псевдослучайных чисел.

Наибольшую популярность фибоначчиевы датчики получили в связи с тем, что скорость выполнения арифметических операций с вещественными числами сравнялась со скоростью целочисленной арифметики, а фибоначчиевы датчики, естественно, реализуются в вещественной арифметике.

Известны разные схемы использования метода Фибоначчи с запаздыванием. Один из широко распространенных фибоначчиевых датчиков основан на следующей рекуррентной формуле:

$$k_i = \begin{cases} k_{i-a} - k_{i-b}, & \text{если } k_{i-a} \geq k_{i-b}; \\ k_{i-a} - k_{i-b} + 1, & \text{если } k_{i-a} < k_{i-b}, \end{cases}$$

где k_i — вещественные числа из диапазона [0, 1], a, b — целые положительные числа, параметры генератора. Для работы фибоначчиеву датчику требуется знать max {a, b} предыдущих сгенерированных случайных чисел. При программной реализации для хранения сгенерированных случайных чисел необходим некоторый объем памяти, зависящий от параметров a и b.

П р и м е р. Вычислим последовательность из первых десяти чисел, генерируемую методом Фибоначчи с запаздыванием, начиная с k_5, при следующих исходных данных: $a = 4$, $b = 1$, $k_0 = 0{,}1$; $k_1 = 0{,}7$; $k_2 = 0.3$; $k_3 = 0{,}9$; $k_4 = 0{,}5$:

$$k_5 = k_1 - k_4 = 0{,}7 - 0{,}5 = 0{,}2;\ k_6 = k_2 - k_5 = 0{,}3 - 0{,}2 = 0{,}1;$$
$$k_7 = k_3 - k_6 = 0{,}9 - 0{,}1 = 0{,}8;\ k_8 = k_4 - k_7 + 1 = 0{,}5 - 0{,}8 + 1 = 0{,}7;$$
$$k_9 = k_5 - k_8 + 1 = 0{,}2 - 0{,}7 + 1 = 0{,}5;\ k_{10} = k_6 - k_9 + 1 = 0{,}1 - 0{,}5 + 1 = 0{,}6;$$
$$k_{11} = k_7 - k_{10} = 0{,}8 - 0{,}6 = 0{,}2;\ k_{12} = k_8 - k_{11} = 0{,}7 - 0{,}2 = 0{,}5;$$
$$k_{13} = k_9 - k_{12} + 1 = 0{,}5 - 0{,}5 + 1 = 1;\ k_{14} = k_{10} - k_{13} + 1 = 0{,}6 - 1 + 1 = 0{,}6.$$

Видим, что генерируемая последовательность чисел внешне похожа на случайную. И действительно, исследования подтверждают, что получаемые случайные числа обладают хорошими статистическими свойствами.

Для генераторов, построенных по методу Фибоначчи с запаздыванием, существуют рекомендуемые параметры a и b, так сказать, протестированные на качество. Например, исследователи предлагают следующие значения: (a, b) = (55, 24), (17, 5) или (97,33). Качество получаемых случайных чисел зависит от значения константы a: чем оно больше, тем выше размерность пространства, в котором сохраняется равномерность случайных векторов, образованных из полученных случайных чисел. В то же время, с увеличением величины константы a увеличивается объем используемой алгоритмом памяти.

Генераторы ПСЧ, основанные на методе Фибоначчи с запаздыванием, использовались для целей криптографии. Кроме того, они применяются в математических и статистических расчетах, а также при моделировании случайных процессов.

Алгоритм BBS. Широкое распространение получил алгоритм генерации псевдослучайных чисел, называемый алгоритмом BBS (от фамилий авторов: L. Blum, M. Blum, M. Shub) или *генератором с квадратичным остатком*. Для целей криптографии этот метод был предложен в 1986 г. Он заключается в следующем. Вначале выбираются два больших простых числа p и q. Числа p и q должны быть оба сравнимы с 3 по модулю 4, то есть при делении p и q на 4 должен получаться одинаковый остаток 3. Далее вычисляется чис-

ло $M = pq$, называемое целым числом Блюма. Затем выбирается другое случайное целое число x, взаимно простое (то есть не имеющее общих делителей, кроме единицы) с M. Вычисляется $x_0 = x^2 \pmod M$, x_0 называется стартовым числом генератора. на каждом n-м шаге работы генератора вычисляется $x_{n+1} = x_n^2 \pmod M$. Результатом n-го шага является один (обычно младший) бит числа x_{n+1}. Иногда в качестве результата принимают бит четности, то есть количество единиц в двоичном представлении элемента. Если количество единиц в записи числа четное — бит четности принимается равным 0, нечетное — бит четности принимается равным 1.

Например, пусть $p = 11$, $q = 19$ (убеждаемся, что 11 (mod 4) = 3, 19 (mod 4) = 3). Тогда $M = pq = 11 \cdot 19 = 209$. Выберем x, взаимно простое с M: пусть $x = 3$. Вычислим стартовое число генератора x_0:

$$x_0 = x^2 \bmod M = 3^2 \bmod 209 = 9 \bmod 209 = 9.$$

Вычислим первые десять чисел x_i по алгоритму BBS. В качестве случайных бит будем брать младший бит в двоичной записи числа x_i:

$x_1 = 9^2 \bmod 209 = 81 \bmod 209 = 81$	младший бит:	1
$x_2 = 81^2 \bmod 209 = 6561 \bmod 209 = 82$	младший бит:	0
$x_3 = 82^2 \bmod 209 = 6724 \bmod 209 = 36$	младший бит:	0
$x_4 = 36^2 \bmod 209 = 1296 \bmod 209 = 42$	младший бит:	0
$x_5 = 42^2 \bmod 209 = 1764 \bmod 209 = 92$	младший бит:	0
$x_6 = 92^2 \bmod 209 = 8464 \bmod 209 = 104$	младший бит:	0
$x_7 = 104^2 \bmod 209 = 10816 \bmod 209 = 157$	младший бит:	1
$x_8 = 157^2 \bmod 209 = 24649 \bmod 209 = 196$	младший бит:	0
$x_9 = 196^2 \bmod 209 = 38416 \bmod 209 = 169$	младший бит:	1
$x_{10} = 169^2 \bmod 209 = 28561 \bmod 209 = 137$	младший бит:	1

Самым интересным для практических целей свойством этого метода является то, что для получения n-го числа последовательности не нужно вычислять все предыдущие n чисел x_i. Оказывается x_n можно сразу получить по формуле

$$x_n = x_0^{2^n \bmod ((p-1)(q-1))} \bmod M$$

Например, вычислим x_{10} сразу из x_0:

$$x_{10} = x_0^{2^{10} \bmod ((11-1)(19-1))} \bmod 209 = x_0^{1024 \bmod 180} \bmod 209 =$$
$$9^{124} \bmod 209 = (9^4)^{31} \bmod 209 = 81^{31} \bmod 209 =$$
$$\left(81^{15} \bmod 209\right)\left(81^{16} \bmod 209\right) = \left(\left(81^3\right)^5 \bmod 209\right)\left(\left(81^4\right)^4 \bmod 209\right) =$$
$$\left(26^5 \bmod 209\right)\left(42^4 \bmod 209\right) = (144 * 104) \bmod 209 = 14976 \bmod 209 = 137$$

В результате действительно получили такое же значение, как и при последовательном вычислении, — 137. Вычисления кажутся достаточно сложными, однако, на самом деле их легко оформить в виде небольшой процедуры или программы и использовать при необходимости.

Возможность «прямого» получения x_n позволяет использовать алгоритм BBS при потоковой шифрации, например, для файлов с произвольным доступом или фрагментов файлов с записями базы данных.

Безопасность алгоритма BBS основана на сложности разложения большого числа M на множители. Утверждается: если M достаточно велико, его можно даже не держать в секрете; до тех пор, пока M не разложено на множители, никто не сможет предсказать выход генератора ПСЧ. Это связано с тем, что задача разложения чисел вида $n = pq$ (p и q — простые числа) на множители является вычислительно очень трудной, если мы знаем только n, а p и q — большие числа, состоящие из нескольких десятков или сотен бит (это так называемая задача факторизации).

Кроме того, можно доказать, что злоумышленник, зная некоторую последовательность, сгенерированную генератором BBS, не сможет определить ни предыдущие до нее биты, ни следующие. Генератор BBS непредсказуем ни в левом направлении, ни в правом направлении. Это свойство очень полезно для целей криптографии, и оно также связано с особенностями разложения числа M на множители.

Самым существенным недостатком алгоритма является то, что он недостаточно быстр, что не позволяет использовать его во многих областях, например, при вычислениях в реальном времени, а также, к сожалению, и при потоковом шифровании.

Зато этот алгоритм выдает действительно хорошую последовательность псевдослучайных чисел с большим периодом (при соответствующем выборе исходных параметров), что позволяет использовать его для криптографических целей при генерации ключей для шифрования.

ВОПРОСЫ И ЗАДАНИЯ ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ ЗНАНИЙ

1. Какой код считается обратимым?
2. Какие задачи решает кодирование при передаче информации по каналам связи?

3. На каких методах основываются алгоритмы сжатия?
4. Каков принцип работы шифров перестановки?
5. Назовите шифры перестановки.
6. Как работает поточное шифрование?
7. Какой генератор псевдослучайных чисел является криптографически стойким?
8. Где используется генератор псевдослучайных чисел?
9. Какая последовательность называется криптографически надежной псевдослучайной последовательностью?
10. Каков принцип работы и достоинства линейных конгруэнтных генераторов псевдослучайных чисел?
11. Каков принцип работы метода Фибоначчи?
12. Каков принцип работы алгоритма BBS?
13. Перечислите достоинства и недостатки алгоритма BBS.
14. Сформулируйте требования к криптографически стойким генераторам псевдослучайных чисел.
15. В чем состоит принцип работы схемы Вижинера?

Раздел II

СИСТЕМЫ ШИФРОВАНИЯ

Глава 4

СИММЕТРИЧНЫЕ СИСТЕМЫ ШИФРОВАНИЯ

4.1. СТРУКТУРНАЯ СХЕМА СИММЕТРИЧНЫХ КРИПТОГРАФИЧЕСКИХ СИСТЕМ

Будем понимать под ***криптографической системой*** (криптосистемой) — пару алгоритмов, реализующих зашифрование и расшифрование информации.

Симметричная криптографическая система использует симметричные алгоритмы шифрования, в которых зашифрование и расшифрование отличаются только порядком выполнения и направлением некоторых шагов. Эти алгоритмы используют один и тот же секретный элемент (ключ), и второе действие (расшифрование) является простым обращением первого (зашифрования). Поэтому обычно каждый из участников обмена может как зашифровать, так и расшифровать сообщение. Схематичная структура такой системы представлена на рис. 4.1 [16, с. 33].

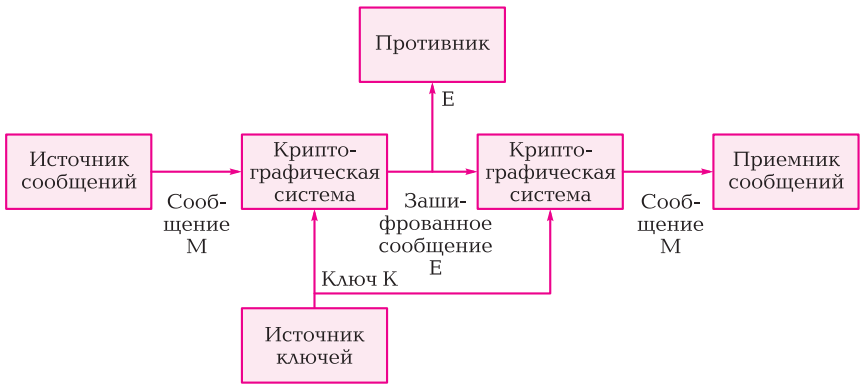

Рис. 4.1. Общая структура симметричной системы

На передающей стороне имеются источник сообщений и источник ключей. Источник ключей выбирает конкретный ключ k среди всех возможных ключей данной системы. Этот ключ k передается некоторым способом принимающей стороне, причем предполагается, что его нельзя перехватить, например, ключ передается специальным курьером (поэтому симметричное шифрование называется также шифрованием с закрытым ключом). Источник сообщений формирует некоторое сообщение M, которое затем шифруется с использованием выбранного ключа. В результате процедуры шифрования получается зашифрованное сообщение C (называемое также *криптограммой* или шифрограммой). Далее криптограмма C передается по каналу связи. Так как канал связи является открытым, незащищенным, например, радиоканал или компьютерная сеть, то передаваемое сообщение может быть перехвачено противником. на принимающей стороне криптограмму C с помощью ключа расшифровывают и получают исходное сообщение M.

Если M — сообщение, K — ключ, а C — зашифрованное сообщение, то процесс зашифрования можно записать так

$$C = f\,(M,\, K),$$

т. е. зашифрованное сообщение C является некоторой функцией от исходного сообщения M и ключа K. Используемый в криптографической системе метод или алгоритм шифрования и определяет функцию f в приведенной выше формуле.

По причине большой избыточности естественных языков непосредственно в зашифрованное сообщение чрезвычайно трудно внести осмысленное изменение, поэтому классическая криптография обеспечивает также *защиту от навязывания ложных данных*. Если же естественной избыточности оказывается недостаточно для надежной защиты сообщения от модификации, избыточность может быть искусственно увеличена путем добавления к сообщению специальной контрольной комбинации, называемой *имитовставкой*.

Наряду с вычислительной простотой и интуитивной понятностью симметричных криптосистем, они обладают рядом серьезных недостатков. К основным недостаткам симметричных криптосистем относят проблему распространения симметричных ключей и проблему их хранения.

При использовании симметричных криптосистем для шифрования информации между пользователями криптографической сети необходимо обеспечить безопасную передачу ключей шифрования между всеми доверенными пользователями (участниками криптографического обмена). При этом передача ключа шифрования обязательно должна осуществляться по закрытому каналу, так как перехват злоумышленником данного ключа ведет к компрометации всей криптографической сети, и дальнейшее шифрование информации теряет смысл. Однако наличие закрытого канала связи позволяет передавать и сам открытый текст по данному каналу. Таким образом, необходимость шифрования как бы отпадает. Аргументы вида «ключ шифрования необходимо передавать достаточно редко по сравнению с передачей закрытых сообщений» хотя и приемлемы, но оставляют данную проблему нерешенной.

Проблема хранения симметричных ключей шифрования заключается в том, что все участники криптографической сети должны обладать ключом шифрования, то есть иметь к нему доступ. При большом количестве участников криптографического обмена данный факт значительно повышает вероятность компрометации ключей шифрования. В связи с этим использование симметричных алгоритмов предполагает наличие взаимного доверия сторон. Недобросовестность отношения одного из тысячи участников криптографического обмена к вопросу хранения ключей может привести к утечке ключевой информации, из-за чего пострадают все участники, в том числе и добросовестно относящиеся к своим обязанностям по хранению ключей. Вероятность компрометации ключей тем выше, чем большее количество пользователей входит в криптографическую сеть. Это является большим недостатком симметричных криптосистем.

4.2. ПРИНЦИПЫ ПОСТРОЕНИЯ КРИПТОГРАФИЧЕСКИХ АЛГОРИТМОВ

Современные криптографические алгоритмы строятся на основе использования простых криптографических алгоритмов. Комбинация нескольких подряд примененных простых шифров, (например, перестановки или замены) дает в результате более сложное преобразование, называемое ***комбинированным*** (композиционным) ***шифром***. Этот шифр обладает более сильной криптографической стойкостью, чем отдельная перестановка или замена.

Рассмотрим следующий пример. Пусть период перестановки d равен 6, а ключ k равен 436 215. Это означает, что в каждом блоке из шести символов четвертый символ становится на первое место,

третий — на второе, шестой — на третье и т.д. Зашифруем с помощью выбранного ключа слово СИГНАЛ:

$$\text{СИГНАЛ} \xrightarrow{k=436215} \text{НГЛИСА}.$$

Будем предполагать, что противнику известен метод шифрования, но неизвестен ключ. Если злоумышленник перехватит сообщение НГЛИСА, ему понадобится не более 720 попыток (при использовании метода полного перебора), чтобы получить открытый текст. Для того чтобы изучить 720 вариантов, на самом деле, требуется не так уж много времени. Предположим, что на изучение каждого варианта у злоумышленника уходит одна секунда. Тогда на все 720 попыток потребуется всего 12 минут. Таким образом, не более чем за 12 минут работы злоумышленник узнает ключ и сможет в дальнейшем расшифровывать все сообщения, закрытые тем же ключом. Если же анализ производится с использованием компьютера, для дешифрования текста НГЛИСА и поиска ключа потребуется гораздо меньше времени.

Необходимо усложнить задачу криптоанализа данного шифра. Для этого можно увеличивать размер периода перестановки, то есть блока, в котором переставляются символы, например, до тысячи знаков. Однако, во-первых, перебор сотен и тысяч знаков на современных компьютерах производится за доли минуты, а во-вторых, при этом до тысячи символов возрастет и размер ключа. Такой ключ уже достаточно трудно запомнить и использовать. Попробуем пойти по другому пути и применим перед перестановкой в блоке из шести символов простую замену, используя шифр Цезаря. Обозначим ключ в шифре Цезаря k_1 ($1 \le k_1 \le 31$), а ключ при перестановке — k_2. Тогда общий ключ $K = (k_1, k_2)$. Таким образом, если K = (5, 436 215), это значит, что вначале шифруемые символы заменяются в соответствии с шифром Цезаря с ключом 5, а затем в каждом блоке из шести символов производится перестановка с ключом 436 215. Выполним в два этапа шифрование слова СИГНАЛ:

$$\text{1 этап (замена): СИГНАЛ} \xrightarrow{k_1=5} \text{ЦОИТЕР}$$

$$\text{2 этап (перестановка): ЦОИТЕР} \xrightarrow{k=436215} \text{ТИРОЦЕ}.$$

Количество возможных ключей в шифре Цезаря равно в нашем случае 31, поэтому общее число вариантов возможных ключей (пространство ключей) в примененном комбинированном шифре равно $3 \cdot 720 = 22\,320$. Таким образом, действительно, полученный комбинированный шифр значительно сильнее отдельно выполненных замены и перестановки.

В реальных шифрах также используется комбинация нескольких простейших операций над блоками знаков. Для повышения криптостойкости эти операции выполняются циклически несколько раз. на стойкость шифра влияют такие факторы, как размер блока, размер ключа, количество повторений операций. Современные шифры с закрытым ключом обрабатывают только двоичные данные, поэтому в них помимо обычных замены и перестановки применяются некоторые другие специфичные для двоичных чисел операции.

Алгоритмы симметричного шифрования могут обрабатывать исходный текст блоками или потоком. В зависимости от этого различают блочные алгоритмы симметричного шифрования и поточные. Блок текста рассматривается как неотрицательное целое число либо как несколько независимых неотрицательных целых чисел. Длина блока всегда выбирается равной степени двойки, например: 64, 128, 256 бит.

В большинстве алгоритмов симметричного шифрования используют ряд операций.

Одна из часто используемых операций — операция *побитового сложения по модулю* 2, обозначаемая XOR либо «ИЛИ». При сложении по модулю 2 операнды обрабатываются поразрядно. В разряде результата ставится единица, если в соответствующих разрядах операндов присутствует нечетное число единиц. Например, сложим по модулю 2 два 16-разрядных числа:

Номер разряда	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Операнд 1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
Операнд 2	0	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1
Сумма по модулю 2	1	1	0	1	1	0	0	1	1	0	0	1	1	0	0	1

Эта операция имеет очень удобное свойство: вычитание по модулю два есть то же самое, что и сложение, поэтому один из операндов может быть получен путем прибавления к сумме другого операнда.

Также в блочных алгоритмах шифрования широко используется операция *сложения по модулю* 2^{32} или по модулю 2^{16}. Эта операция представляет собой обыкновенное сложение двоичных чисел без учета переноса в старший 32-й или 16-й разряд результата.

Циклический сдвиг передвигает цепочку битов на некоторое число разрядов влево или вправо. При циклическом сдвиге влево биты, выходящие слева за разрядную сетку, дописываются справа

на освободившиеся места. При циклическом сдвиге вправо все биты передвигаются цепочкой вправо, а те, которым не хватает места, переносятся в конец цепочки.

При выполнении *табличной подстановки* группа битов отображается в другую группу битов. При этой операции один блок двоичных данных заменяется по определенному правилу или таблице другим блоком. В общем случае для *n*-битовых блоков таблица замен должна содержать 2^n элементов. Табличную подстановку иногда называют заменой с использованием *S*-блоков или *S-box*. (Буква *S* взята от английского слова *substitution* — подстановка.)

С помощью операции *перемещения* биты сообщения переупорядочиваются. Перемещение называют также *permutation* или *P*-блоком.

В алгоритмах симметричного шифрования часто используются операции сложения по модулю 2, сложения по модулю 2^{16} или 2^{32}, циклического сдвига, замены и перестановки.

Эти операции циклически повторяются в алгоритме *N* раз, образуя так называемые *раунды*, или *шаги*. Исходными данными для каждого раунда являются выход предыдущего раунда и ключ, который получен по определенному алгоритму из общего ключа шифрования *K*. Ключ раунда называется *подключом* K_i.

Блочные алгоритмы шифрования применяются к двоичным данным. В общем случае процедура блочного шифрования преобразовывает *n*-битный блок открытого текста в *k*-битный блок зашифрованного текста. Число блоков длины *n* равно 2^n. Для того чтобы преобразование было обратимым, каждый из таких блоков должен преобразовываться в свой уникальный блок зашифрованного текста. Длина блока всегда выбирается равной степени двойки, например: 64, 128, 256 бит.

К современным алгоритмам блочного шифрования предъявляют достаточно жесткие требования, связанные с областью применения, возможностью реализации на различных вычислительных платформах и другими факторами. Перечислим основные из требований.

1. Алгоритм должен обеспечивать высокий уровень стойкости, и эта стойкость не должна основываться на сохранении в тайне самого алгоритма.

2. Незначительное изменение исходного сообщения должно приводить к существенному изменению зашифрованного сообщения даже при использовании одного и того же ключа.

3. Алгоритм шифрования должен успешно противостоять атакам по выбранному тексту, то есть таким, чтобы нельзя было узнать ключ, даже зная достаточно много пар (зашифрованное сообщение — незашифрованное сообщение), полученных при шифровании с использованием данного ключа.

4. Возможность быть реализованным на разных платформах, которые предъявляют различные требования.

5. Использовать простые операции, которые эффективны на микропроцессорах, т. е. исключающее «ИЛИ», сложение, табличные подстановки, умножение по модулю. Не должно использоваться сдвигов переменной длины, побитных перестановок или условных переходов.

6. Реализация на специализированной аппаратуре, предназначенной для выполнения операций шифрования и расшифрования, то есть реализация алгоритма в виде электронных устройств должна быть экономичной.

7. Алгоритм шифрования должен быть применим во многих приложениях. Он должен быть эффективен при шифровании файлов данных или большого потока данных, при создании определенного количества случайных битов, а также должна быть возможность его использования для формирования односторонней хеш-функции.

8. Быть простым для написания кода, чтобы минимизировать вероятность программных ошибок. Также это дает возможность анализа и уменьшает закрытость алгоритма.

9. Допускать любую случайную строку битов нужной длины в качестве возможного ключа (это называется иметь плоское пространство ключей). Не должно быть «слабых» ключей, облегчающих криптоанализ.

10. Алгоритм должен легко модифицироваться для различных уровней безопасности и удовлетворять как минимальным, так и максимальным требованиям.

4.3. ОТЕЧЕСТВЕННЫЕ КРИПТОГРАФИЧЕСКИЕ АЛГОРИТМЫ

Криптографический алгоритм ГОСТ 28147—89. В России в качестве стандарта на блочные алгоритмы шифрования с закрытым ключом в 1989 г. был принят ГОСТ 28147—89. Он используется для криптографической защиты данных. Шифр, предлагаемый ГОСТ Р 28147—89, построен по тем же принципам, что и американский алгоритм DES, однако по сравнению с ним отечественный стандарт шифрования более удобен для программной реализации.

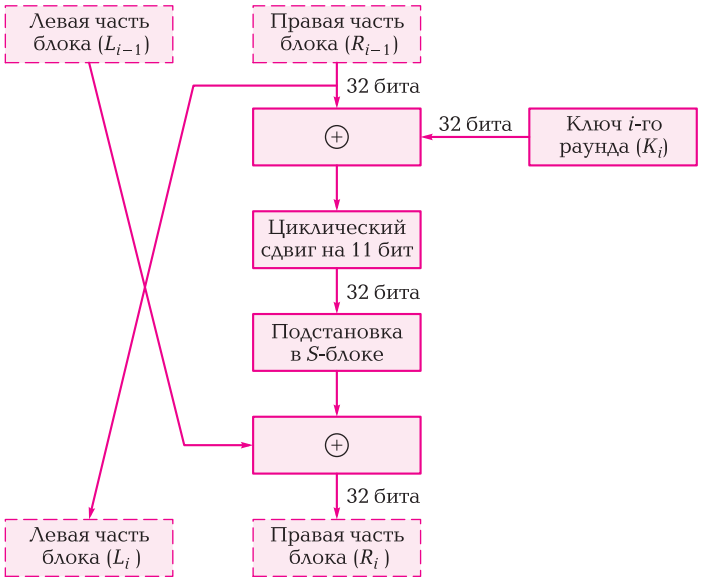

Рис. 4.2. Структура одного раунда ГОСТ 28147 – 89

Основные параметры алгоритма криптографического преобразования данных ГОСТ Р 28147—89 следующие: размер блока составляет 64 бита, размер ключа — 256 бит, количество раундов — 32. Алгоритм представляет собой классическую сеть Фейстеля. Структура одного раунда ГОСТ 28147—89 приведена на рис. 4.2 [16, с. 38]. Шифруемый блок данных разбивается на две части, которые затем обрабатываются как отдельные 32-битовые целые числа без знака. Сначала правая половина блока и подключ раунда складываются по модулю 2^{32}, затем производится поблочная подстановка. 32-битовое значение, полученное на предыдущем шаге (обозначим его S), интерпретируется как массив из восьми 4-битовых блоков кода: $S = (S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_7)$.

Далее значение каждого из восьми блоков заменяется на новое, которое выбирается по таблице замен следующим образом: значение блока S_i заменяется на S_i-й по порядку элемент (нумерация с нуля) i-го узла замен (т. е. i-й строки таблицы замен, нумерация также с нуля). Другими словами, в качестве замены для значения блока выбирается элемент с номером строки, равным номеру заменяемого блока, и номером столбца, равным значению заменяемого блока как 4-битового целого неотрицательного числа. В каждой строке таблицы замен записаны числа от 0 до 15 в произвольном порядке без повторений. Значения элементов таблицы замен взяты от 0 до 15, так как в четырех битах, которые подвергаются подстановке, может быть записано целое число без знака в диапазоне от 0 до 15. Например, первая строка S-блока может содержать такие значения: 5, 8, 1, 13, 10, 3, 4, 2, 14, 15, 12, 7, 6, 0, 9, 11. В этом случае значение блока S_0 (четыре младших бита 32-разрядного числа S) заменится на число, стоящее на позиции, номер которой равен значению заменяемого блока. Если $S_0 = 0$, то оно заменится на 5, если $S_0 = 1$, то оно заменится на 8 и т. д.

После выполнения подстановки все 4-битовые блоки снова объединяются в единое 32-битное слово, которое затем циклически сдвигается на 11 битов влево. Наконец, с помощью побитовой операции «сумма по модулю 2» результат объединяется с левой половиной, вследствие чего получается новая правая половина R_i. Новая левая часть L_i берется равной младшей части преобразуемого блока: $L_i = R_{i-1}$.

Полученное значение преобразуемого блока рассматривается как результат выполнения одного раунда алгоритма шифрования.

Шифр ГОСТ Р 28147—89 предусматривает следующие режимы шифрования данных: простая замена, гаммирование, гаммирование с обратной связью и один дополнительный режим выработки имитовставки.

В любом из этих режимов данные обрабатываются блоками по 64 бита, на которые разбивается шифруемый массив, именно поэтому ГОСТ Р 28147—89 относится к блочным шифрам. В режимах гаммирования есть возможность обработки неполного блока данных размером меньше 8 байт, что существенно при шифровании массивов данных с произвольным размером, который может быть не кратным 8 байтам.

В *режиме простой замены* каждый блок исходных данных шифруется независимо от остальных блоков, с применением одного и того же ключа шифрования. Особенностью этого режима является то, что одинаковые блоки исходного текста преобразуются в одинаковый шифротекст. Поэтому ГОСТ Р 28147—89 рекомендует использовать режим простой замены только для шифрования ключей.

Режимы гаммирования и гаммирования с обратной связью могут использоваться для шифрования данных произвольного размера. В *режиме гаммирования* биты исходного текста складываются по модулю 2 с гаммой, которая вырабатывается с помощью алгоритма шифрования по ГОСТ Р 28147—89. То есть алгоритм шифрования по ГОСТ Р 28147—89 в данном режиме ис-

пользуется в качестве генераторов 64-разрядных блоков гаммы. При шифровании каждого нового блока данных гамма, использованная на предыдущем шаге, зашифровывается и используется уже как новая гамма. В качестве начального массива данных, шифруемых для получения самой первой гаммы, используется так называемая *синхропосылка* — 64-разрядный начальный блок данных, который должен быть одинаковым на шифрующей и расшифровывающей стороне. Благодаря тому, что наложение и снятие гаммы осуществляется при помощи одной и той же операции сложения по модулю 2, алгоритмы зашифрования и расшифрования в режиме гаммирования совпадают. Так как все элементы гаммы различны для реальных шифруемых массивов, то результат зашифрования даже двух одинаковых блоков в одном массиве данных будет различным. Кроме того, хотя элементы гаммы и вырабатываются одинаковыми порциями в 64 бита, использоваться может и часть такого блока с размером, равным размеру шифруемого блока. Именно это дает возможность шифрования неполных блоков данных.

Режим *гаммирования с обратной связью* похож на режим гаммирования и отличается от него только способом выработки элементов гаммы. При гаммировании с обратной связью очередной 64-битный элемент гаммы вырабатывается как результат преобразования по базовому циклу алгоритма ГОСТ Р 28147—89 предыдущего блока зашифрованных данных. Для зашифрования первого блока массива данных элемент гаммы вырабатывается как результат преобразования по тому же циклу синхропосылки. Этим достигается зацепление блоков: каждый блок шифротекста в этом режиме зависит от соответствующего и всех предыдущих блоков открытого текста. Поэтому данный режим иногда называется *гаммированием с зацеплением блоков*. на стойкость шифра факт зацепления блоков не оказывает никакого влияния.

Для решения задачи обнаружения искажений в зашифрованном массиве данных в ГОСТ Р 28147—89 предусмотрен дополнительный режим криптографического преобразования — выработка имитовставки. Имитовставка — это контрольная комбинация, зависящая от открытых данных и секретной ключевой информации. Целью использования имитовставки является обнаружение всех случайных или преднамеренных изменений в массиве информации. В режиме выработки имитовставки входной текст обрабатывается блоками следующим образом:

$$Y = f\left((X_{i-1}),\ K\right)$$

для всех i от 1 до n, где f — базовый цикл по ГОСТ 28147—89; X_i — 64-разрядный блок исходного текста; K — ключ.

В качестве имитовставки берется часть блока Y_n, полученного на выходе, обычно 32 его младших бита.

Таким образом, злоумышленник, не владея ключом шифрования, не может вычислить имитовставку для заданного открытого массива информации, а также подобрать открытые данные под заданную имитовставку.

Криптографический алгоритм «Магма». В 2015 г. был принят новый государственный стандарт шифрования ГОСТ Р 34.12—2015 «Информационная технология. Криптографическая защита информации. Блочные шифры». Стандарт содержит описание двух блочных шифров с длиной блока 128 и 64 бит.

Шифр «Магма» (Magma), изложенный в ГОСТ Р 34.12—2015, представляет собой симметричный блочный алгоритм шифрования с размером блока входных данных 64 бита, секретным ключом 256 бит и 32 раундами шифрования. В основе алгоритма шифра «Магма» лежит алгоритм шифра ГОСТ Р 28147—89. Главным отличием этих алгоритмов стало использование зафиксированных блоков нелинейной подстановки. Ниже приведены закрепленные в стандарте блоки замен:

S_0 = (12, 4, 6, 2, 10, 5, 11, 9, 14, 8, 13, 7, 0, 3, 15, 1);

S_1 = (6, 8, 2, 3, 9, 10, 5, 12, 1, 14, 4, 7, 11, 13, 0, 15);

S_2 = (11, 3, 5, 8, 2, 15, 10, 13, 14, 1, 7, 4, 12, 9, 6, 0);

S_3 = (12, 8, 2, 1, 13, 4, 15, 6, 7, 0, 10, 5, 3, 14, 9, 11);

S_4 = (7, 15, 5, 10, 8, 1, 6, 13, 0, 9, 3, 14, 11, 4, 2, 12);

S_5 = (5, 13, 15, 6, 9, 2, 12, 10, 11, 7, 8, 1, 4, 3, 14, 0);

S_6 = (8, 14, 2, 5, 6, 9, 1, 12, 15, 4, 11, 0, 13, 10, 3, 7);

S_7 = (1, 7, 14, 13, 0, 5, 8, 3, 4, 15, 10, 6, 9, 12, 11, 2).

Описанный в стандарте набор S-блоков обеспечивает наилучшие характеристики, определяющие стойкость криптоалгоритма к дифференциальному и линейному криптоанализу. В остальном алгоритмы шифров совпадают.

Криптографический алгоритм «Кузнечик». Шифр «Кузнечик» изложен в документе ГОСТ Р 34.12—2015 «Информационная технология. Криптографическая защита информации. Режимы работы блочных шифров». Данный ГОСТ вступил в действие с 1 января 2016 г. Шифр «Кузнечик» обладает следующими параметрами:

размер блока — 128 бит; размер ключа — 256 бит; принцип действия — SP-сеть.

Операция зашифрования блока состоит из девяти циклов шифрования и одного конечного преобразования. В начале выполнения шифрования переменные *a* и *Key* содержат исходные данные и ключ шифрования соответственно. Алгоритм содержит следующую общую формулу зашифрования (в формуле ключи нумеруются с 1):

$$E_{K1...K10}(a) = X[K_{10}]\ LSX[K_9]\ ...\ LSX[K_2]\ LSX[K_1](a).$$

Операция расширения (развертывания) ключа *ExpKey* вычисляет 128-битные итерационные ключи $K_0...K_9$ из 256-битного исходного ключа с использованием 32 итерационных констант C_i, полученных по формуле $C_i = L(i)$, $i = \overline{1,...,32}$. Ключи вычисляются парами по принципу сети Фейстеля по следующему алгоритму:

1) $K_0 = k_{255} \| ... \| k_{128}$ («||» — конкатенация, k_n — *n*-й бит ключа *Key*);

2) $K_1 = k_{127} \| ... \| k_0$;

3) $(K_{2\cdot i}, K_{2\cdot i+1}) = F'(K_{2\cdot i-2}, K_{2\cdot i-1}), i = \overline{1,4}$.

Операция *F'* состоит в 8-кратном повторении совокупности операций

$$F(K', K'') = \left(L\left(S\left(X\left(C_{8(i-1)+j}, K'\right)\right)\right) \oplus K'', K'\right), j = \overline{1,8}.$$

Шифрование состоит из следующих операций:

1) *X* — сумма по mod 2 двух входных параметров (в данном случае — итерационного ключа K_i и сообщения *a*);

2) *S* — замена байтов по таблице замен P (табл. 2.1);

3) *L* — операция, в которой над входным блоком данных 16 раз выполняется операция *R*.

Таблица 4.1. Массив замены P

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00	Fc	ee	dd	11	cf	6e	31	16	fb	c4	fa	da	23	c5	04	4d
10	e9	77	f0	db	93	2e	99	ba	17	36	f1	bb	14	cd	5f	c1
20	f9	18	65	5a	e2	5c	ef	21	81	1c	3c	42	8b	01	8e	4f
30	05	84	02	ae	e3	6a	8f	a0	06	0b	ed	98	7f	d4	d3	1f
40	Eb	34	2c	51	ea	c8	48	ab	f2	2a	68	a2	fd	3a	ce	cc
50	b5	70	0e	56	08	0c	76	12	bf	72	13	47	9c	b7	5d	87

Окончание табл. 4.1

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
60	15	a1	96	29	10	7b	9a	c7	f3	91	78	6f	9d	9e	b2	b1
70	32	75	19	3d	ff	35	8a	7e	6d	54	c6	80	c3	bd	0d	57
80	Df	f5	24	a9	3e	a8	43	c9	d7	79	d6	f6	7c	22	b9	03
90	e0	0f	ec	de	7a	94	b0	bc	dc	e8	28	50	4e	33	0a	4a
a0	a7	97	60	73	1e	00	62	44	1a	b8	38	82	64	9f	26	41
b0	Ad	45	46	92	27	5e	55	2f	8c	a3	a5	7d	69	d5	95	3b
c0	07	58	b3	40	86	ac	1d	f7	30	37	6b	e4	88	d9	e7	89
d0	e1	1b	83	49	4c	3f	f8	fe	8d	53	aa	90	ca	d8	85	61
e0	20	71	67	a4	2d	2b	09	5b	cb	9b	25	d0	be	e5	6c	52
f0	59	a6	74	d2	e6	f4	b4	c0	d1	66	af	c2	39	4b	63	b6

Операция *R* заключается в сдвиге (не циклическом) вправо на 8 бит входной последовательности данных и записи в старший байт результата операции линейного преобразования *l* входных данных:

$$R(a_{15} \| ... \| a_0) = l(a_{15} ... a_0) \| a_{15} \| ... \| a_1.$$

Операция *l* заключается в умножении в поле $GF(2^8)$ по модулю $p^h(x) = 0x1C3$ байтов входной последовательности на константы D_i, приведенные в табл. 4.2, и сложении в том же поле полученных произведений:

$$\ell(a_{15}...a_0) = \oplus_{i=0}^{15}(D_i \cdot a_i).$$

Таблица 4.2. Таблица констант D

Номер байта (*i*)	Константа	Номер байта (*i*)	Константа	Номер байта (*i*)	Константа	Номер байта (*i*)	Константа
0	0×01	4	0×10	8	0×fb	12	0×10
1	0×94	5	0×c2	9	0×01	13	0×85
2	0×20	6	0×c0	10	0×c0	14	0×20
3	0×85	7	0×01	11	0×c2	15	0×94

Расшифрование блока происходит симметрично зашифрованию с использованием операций, обратных операциям S, L, R. Операция развертывания ключа не имеет обратного преобразования, так как относится (наряду с процедурой разбиения текста на блоки) к операции подготовки. Операция X обратна самой себе. Данный ГОСТ содержит следующую общую формулу расшифрования:

$$D_{K1...K10}(a) = X[K_1]\, S^{-1} L^{-1} X[K_2] \ldots S^{-1} L^{-1} X[K_9]\, S^{-1} L^{-1} X[K_{10}](a).$$

Расшифрование состоит из следующих обратных операций:

1) S^{-1} (Inv S) — замена байтов слова по таблице Inv P, обратной P (табл. 2.3);

2) L^{-1} (Inv L) — 16-кратное повторение операции R^{-1}.

Операция R^{-1} выполняет сдвиг влево на 8 бит входных данных и запись в младший байт результата операции l от циклического сдвига на 8 бит входных данных:

$$R^{-1}(a_{15} \| \ldots \| a_0) = a_{14} \| \ldots \| a_0 \| l(a_{14}, a_{13} \ldots a_0, a_{15}).$$

Таблица 4.3 Обратный массив замены Inv P

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00	a5	2d	32	8f	0e	30	38	c0	54	e6	9e	39	55	7e	52	91
10	64	03	57	5a	1c	60	07	18	21	72	a8	d1	29	c6	a4	3f
20	e0	27	8d	0c	82	ea	ae	b4	9a	63	49	e5	42	e4	15	b7
30	c8	06	70	9d	41	75	19	c9	aa	fc	4d	bf	2a	73	84	d5
40	c3	af	2b	86	a7	b1	b2	5b	46	d3	9f	fd	d4	0f	9c	2f
50	9b	43	ef	d9	79	b6	53	7f	c1	f0	23	e7	25	5e	b5	1e
60	a2	df	a6	fe	ac	22	f9	e2	4a	bc	35	ca	ee	78	05	6b
70	51	e1	59	a3	f2	71	56	11	6a	89	94	65	8c	bb	77	3c
80	7b	28	ab	d2	31	de	c4	5f	cc	cf	76	2c	b8	d8	2e	36
90	db	69	b3	14	95	be	62	a1	3b	16	66	e9	5c	6c	6d	ad
a0	37	61	4b	b9	e3	ba	f1	a0	85	83	da	47	c5	b0	33	fa
b0	96	6f	6e	c2	f6	50	ff	5d	a9	8e	17	1b	97	7d	ec	58
c0	f7	1f	fb	7c	09	0d	7a	67	45	87	dc	e8	4f	1d	4e	04
d0	eb	f8	f3	3e	3d	bd	8a	88	dd	cd	0b	13	98	02	93	80
e0	90	d0	24	34	cb	ed	f4	ce	99	10	44	40	92	3a	01	26
f0	12	1a	48	68	f5	81	8b	c7	d6	20	0a	08	00	4c	d7	74

4.4. ЗАРУБЕЖНЫЕ КРИПТОГРАФИЧЕСКИЕ АЛГОРИТМЫ

Сети Фейстеля. Большой вклад в исследования принципов разработки блочных шифров внес американский ученый Х. Фейстель (Horst Feistel). Фейстель предложил структуру, называемую в настоящее время ***сетью Фейстеля***. Сети Фейстеля получили широкое распространение, так как, с одной стороны, они удовлетворяют всем требованиям к алгоритмам симметричного шифрования, а с другой — достаточно просты и удобны в использовании.

Раунд, организованный по сети Фейстеля, имеет следующую структуру. Входной блок делится на несколько частей равной длины. Эти части блока называются *ветвями*. Так, например, если блок имеет длину 64 бита, используются две ветви по 32 бита каждая. Ветви обрабатываются по отдельности, после чего осуществляется циклический сдвиг всех ветвей влево. В случае двух ветвей каждый раунд имеет структуру, показанную на рис. 4.3 [16, с. 73].

Функция F называется *образующей*. Каждый раунд состоит из вычисления функции F для одной ветви и побитового выполнения операции «сумма по модулю 2» результата F с другой ветвью. После этого ветви меняются местами. Именно тем, как определяется функция F, системы шифрования и отличаются друг от друга. В некоторых алгоритмах введены также начальные преобразования входного блока данных, придающие некоторую «случайность» входному тексту (это называется *рандомизацией* данных.)

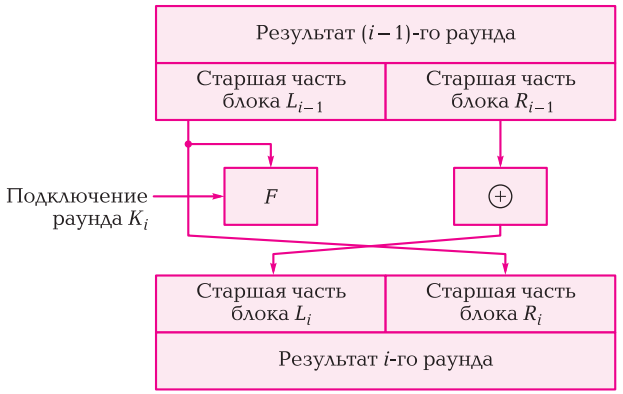

Рис. 4.3. i-й раунд сети Фейстеля

Рандомизация производится для того, чтобы уменьшить естественную избыточность входного сообщения. Число раундов может быть различным для разных алгоритмов. В некоторых алгоритмах рекомендуется от 8 до 32 раундов, в других — больше. В целом увеличение количества раундов увеличивает криптостойкость алгоритма. В последнее время все чаще используются различные разновидности сети Фейстеля для 128-битного блока с четырьмя ветвями. Увеличение количества ветвей, а не размерности каждой ветви, связано с тем, что наиболее популярными до сих пор остаются процессоры с 32-разрядными словами, следовательно, оперировать 32-разрядными словами эффективнее, чем 64-разрядными.

Криптографический алгоритм DES. Одной из наиболее известных криптографических систем с закрытым ключом является алгоритм DES — Data Encryption Standard. Эта система первой получила статус государственного стандарта в области шифрования данных. Она разработана специалистами фирмы IBM и вступила в действие в США 1977 г. Алгоритм DES широко использовался при хранении и передаче данных между различными вычислительными системами; в почтовых системах, в электронных системах чертежей и при электронном обмене коммерческой информацией. Стандарт DES реализовывался как программно, так и аппаратно. Предприятиями разных стран был налажен массовый выпуск цифровых устройств, использующих DES для шифрования данных. Все устройства проходили обязательную сертификацию на соответствие стандарту.

Несмотря на то что уже некоторое время эта система не имеет статуса государственного стандарта, она по-прежнему широко применяется и заслуживает внимания при изучении блочных шифров с закрытым ключом.

Длина ключа в алгоритме DES составляет 56 бит. Общая структура DES представлена на рис. 4.4 [16, с. 81]. Процесс шифрования каждого 64-битового блока исходных данных можно разделить на три этапа:

1) начальная подготовка блока данных;
2) 16 раундов основного цикла;
3) конечная обработка блока данных.

На первом этапе выполняется начальная перестановка 64-битного исходного блока текста, во время которой биты определенным образом переупорядочиваются. Эта операция иногда называется «забеливание» — whitening. При начальной перестановке биты блока данных определенным образом переупорядочиваются.

Рис. 4.4. Общая схема алгоритма DES

Эта операция придает некоторую «хаотичность» исходному сообщению, снижая возможность использования криптоанализа статистическими методами.

Одновременно с начальной перестановкой блока данных выполняется начальная перестановка 56 бит ключа. В каждом из раундов используется соответствующий 48-битный частичный ключ K_i. Ключи K_i получаются по определенному алгоритму, используя каждый из битов начального ключа по несколько раз. В каждом раунде 56-битный ключ делится на две 28-битовые половинки. Затем половинки сдвигаются влево на один или два бита в зависимости от номера раунда. После сдвига определенным образом выбирается 48 из 56 битов. Так как при этом не только выбирается подмножество битов, но и изменяется их порядок, то эта операция называется «перестановка со сжатием». Ее результатом является набор из 48 битов. В среднем каждый бит исходного 56-битного ключа используется в 14 из 16 подключей, хотя не все биты используются одинаковое количество раз.

Далее выполняется основной цикл преобразования, организованный по сети Фейстеля и состоящий из 16 одинаковых раундов. При этом в каждом раунде, показанном на рис. 4.5 [16, с. 83] получается промежуточное 64-битное значение, которое затем обрабатывается в следующем раунде.

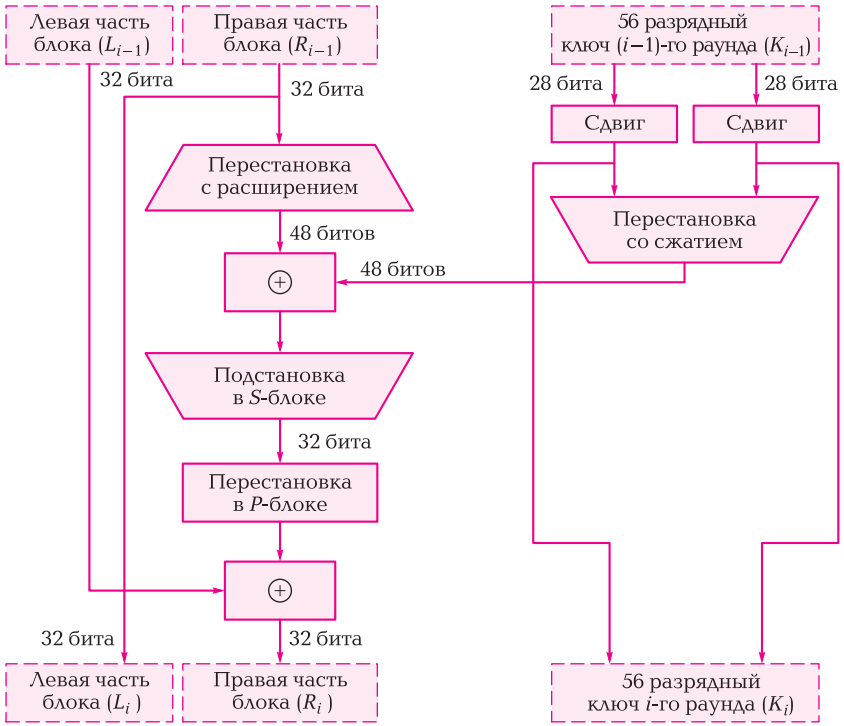

Рис. 4.5. Структура одного раунда алгоритма DES

Левая и правая ветви каждого промежуточного значения обрабатываются как отдельные 32-битные значения, обозначенные L и R.

Вначале правая часть блока R_i расширяется до 48 битов, используя таблицу, которая определяет перестановку, плюс расширение на 16 битов. Эта операция приводит размер правой половины в соответствие с размером ключа для выполнения операции XOR. Кроме того, за счет выполнения этой операции быстрее возрастает зависимость всех битов результата от битов исходных данных и ключа (это называется *лавинным эффектом*). Чем сильнее проявляется лавинный эффект при использовании того или иного алгоритма шифрования, тем лучше.

После выполнения перестановки с расширением для полученного 48-битного значения выполняется операция XOR с 48-битным подключом K_i. Затем полученное 48-битное значение подается на вход блока подстановки S, результатом которой является 32-битное значение. Подстановка выполняется в восьми блоках

подстановки, или восьми S-блоках. При выполнении этой операции 48 битов данных делятся на восемь 6-битовых подблоков, каждый из которых по своей таблице замен заменяется четырьмя битами. Подстановка с помощью S-блоков является одним из важнейших этапов DES. Таблицы замен для этой операции специально спроектированы специалистами так, чтобы обеспечивать максимальную безопасность. В результате выполнения этого этапа получаются восемь 4-битовых блоков, которые вновь объединяются в единое 32-битовое значение.

Далее полученное 32-битовое значение обрабатывается с помощью перестановки P, которая не зависит от используемого ключа. Целью перестановки является такое максимальное переупорядочивание битов, чтобы в следующем раунде шифрования каждый бит с большой вероятностью обрабатывался другим S-блоком.

И наконец, результат перестановки объединяется с помощью операции XOR с левой половиной первоначального 64-битового блока данных. Затем левая и правая половины меняются местами, и начинается следующий раунд.

После шестнадцати раундов шифрования выполняется конечная перестановка результата. Эта перестановка инверсна (обратна) начальной перестановке.

После выполнения всех указанных шагов блок данных считается полностью зашифрованным и можно переходить к шифрованию следующего блока исходного сообщения.

При расшифровании на вход алгоритма подается зашифрованный текст. Единственное отличие состоит в обратном порядке использования частичных ключей K_i. K_{16} используется на первом раунде, K_1 — на последнем раунде.

После последнего раунда процесса расшифрования две половины выхода меняются местами так, чтобы вход заключительной перестановки был составлен из R_{16} и L_{16}. Выходом этой стадии является незашифрованный текст.

Алгоритм TripleDES (3DES). Для усиления стойкости алгоритм DES применяют несколько раз. Широкое распространение получил алгоритм TripleDES (3DES): метод трехкратного шифрования, использующий три разных ключа (на каждом из этапов свой). Общая длина ключа в таком методе возрастает (112 + 56 = 168), но хранение нескольких сотен бит обычно не является проблемой. Использование алгоритма 3DES позволяет решить проблему короткого ключа, используемого в алгоритме DES.

Тройной DES является достаточно популярной альтернативой DES и используется при управлении ключами в стандартах ANSI

X9.17 и ISO 8732. Некоторые криптоаналитики предлагают для еще более надежного шифрования использовать пятикратный DES с тремя или пятью ключами.

Криптографический алгоритм AES. Усовершенствованный алгоритм AES. AES (Advanced Encryption Standard) — американский стандарт шифрования, опубликованный в 2001 г. В современных криптографических продуктах, наверное, не найдется таких, которые бы не использовали AES. Он используется в Wi-Fi, WinRAR, PGP.

Алгоритм AES представляет блок данных в виде двумерного байтового массива размером 4×4. Все операции производятся над отдельными байтами массива, а также над независимыми столбцами и строками. В каждом раунде алгоритма выполняется ряд преобразований.

Операция *SubBytes*, представляющая собой табличную замену каждого байта массива данных.

Операция *ShiftRows*, которая выполняет циклический сдвиг влево всех строк массива данных, за исключением нулевой. Сдвиг i-й строки массива (для $i = 1, 2, 3$) производится на i байт.

Операция *MixColumns* выполняет умножение каждого столбца массива данных на фиксированный полином $a(x)$:

$$a(x) = 3x^3 + x^2 + x + 2.$$

Умножение выполняется по модулю $x^4 + 1$.

Операция *AddRoundKey* выполняет наложение на массив данных материала ключа. На i-й столбец массива данных ($i = 0, ..., 3$) побитовой логической операцией «исключающее ИЛИ» (XOR) накладывается определенное слово расширенного ключа W_{4r+i}, где r — номер текущего раунда алгоритма, начиная с 1. Количество раундов алгоритма R зависит от размера ключа (табл. 4.4).

Перед первым раундом алгоритма выполняется предварительное наложение ключа с помощью операции *AddRoundKey*, которая выполняет наложение на открытый текст первых четырех

Таблица 4.4. Зависимость количества раундов алгоритма от размера ключа

Размер ключа, бит	Количество раундов алгоритма AES
128	10
192	12
256	14

слов расширенного ключа $W_0 ... W_3$. Последний же раунд отличается от предыдущих тем, что в нем не выполняется операция *MixColumns*.

Расшифрование выполняется применением обратных операций в обратной последовательности. Соответственно, перед первым раундом расшифрования выполняется операция *AddRoundKey* (которая является обратной самой себе), выполняющая наложение на шифротекст четырех последних слов расширенного ключа, т. е. $W_{4R} ... W_{4R+3}$. Затем выполняется R раундов расшифрования, каждый из которых выполняет следующие преобразования. Операция *InvShiftRows* выполняет циклический сдвиг вправо трех последних строк массива данных на то же количество байт, на которое выполнялся сдвиг операцией *ShiftRows* при зашифровании. Операция *InvSubBytes* выполняет побайтно обратную табличную замену. Операция *AddRoundKey*, как и при зашифровании, выполняет наложение на обрабатываемые данные четырех слов расширенного ключа $W_{4r} ... W_{4r+3}$. Однако, нумерация раундов r при расшифровании производится в обратную сторону — от R–1 до 0.

Операция *InvMixColumns* выполняет умножение каждого столбца массива данных аналогично прямой операции *MixColumns*, однако, умножение производится на полином $a - 1(x)$, определенный следующим образом:

$$a - 1(x) = Bx^3 + Dx^2 + 9x + E.$$

Аналогично зашифрованию, последний раунд расшифрования не содержит операцию *InvMixColumns*.

AES использует ключи шифрования трех фиксированных размеров: 128, 192, и 256 бит. В зависимости от размера ключа, конкретный вариант алгоритма AES может обозначаться как AES-128, AES-192 и AES-256 соответственно.

Задача процедуры расширения ключа состоит в формировании нужного количество слов расширенного ключа для их использования в операции *AddRoundKey*. Как было сказано выше, под «словом» здесь понимается 4-байтный фрагмент расширенного ключа, один из которых используется в первичном наложении ключа и по одному — в каждом раунде алгоритма. Таким образом, в процессе расширения ключа формируется $4(R + 1)$ слов.

Расширение ключа выполняется в два этапа, на первом из которых производится инициализация слов расширенного ключа (обозначаемых как W_i): первые N_k — размер исходного ключа шифрования K в словах, т. е. 4, 6 или 8) слов W_i (т. е. $i = 0 ... N_{k-1}$) формируются их последовательным заполнением байтами ключа.

Последующие слова W_i формируются следующей последовательностью операций для каждого $i = N_k \ldots 4\ (R + 1) - 1$.

Шаг 1. Инициализируется временная переменная T:

$$T = W_{i-1};$$

Шаг 2. Данная переменная модифицируется следующим образом:

а) если i кратно N_k то:

$$T = SubWord\ (RotWord\ (T))RCn/N_k;$$

константы *RCn* представляют собой слова, в которых все байты, кроме первого являются нулевыми, а первый байт имеет значение $2^{n-1} (\text{mod } 256)$;

б) если $N_k = 8$ и $i\ (\text{mod } N_k) = 4$ то

$$T = SubWord\ (T);$$

в) *в* остальных случаях модификация переменной T не выполняется.

Шаг 3. Формируется i-е слово расширенного ключа:

$$W_i = W_i - N_k T.$$

Операция *SubWord* выполняет над каждым байтом входного значения табличную замену, которая была описана выше — см. операцию *SubBytes*.

Операция *RotWord* побайтно вращает входное слово на 1 байт влево.

Как видно, процедура расширения ключа является достаточно простой по сравнению со многими другими современными алгоритмами шифрования. Процедура расширения ключа имеет также несомненное достоинство в том, что расширение ключа может быть выполнено «на лету» (on-the-fly), т. е. параллельно с зашифрованием данных.

4.5. ПРОБЛЕМА РАСПРЕДЕЛЕНИЯ КЛЮЧЕЙ СИММЕТРИЧНОГО ШИФРОВАНИЯ

Рассмотрим процесс распределения ключей для криптографической системы, использующей симметричное шифрование. Отправитель, представляющий собой источник сообщений, и получатель (приемник зашифрованных сообщений) договариваются о выборе приемлемого шифра и ключа. Затем отправитель шифрует свое сообщение с использованием выбранного алгоритма шифрования и ключа и пересылает полученный шифротекст по (открытому) каналу связи. Получатель расшифровывает его, используя шифр и ключ.

Злоумышленник, скорее всего, может перехватить зашифрованное сообщение, так как предполагается, что оно передается по открытому каналу связи. В этом случае криптоаналитик злоумышленника может попытаться вскрыть шифротекст. Будем предполагать, что отправитель и получатель сообщения используют достаточно надежный шифр и что вероятность его вскрытия невысока. В этом случае безопасность шифрования полностью зависит от безопасности ключа. Раскрытие ключа приведет к раскрытию передаваемых данных. Таким образом, ключ должен храниться в секрете до тех пор, пока он используется для шифрования данных. Поэтому для первоначального распределения ключей необходим надежный канал связи. Самым надежным способом первоначального распределения ключей является обмен ключами при личной встрече абонентов сети передачи данных. Для доставки ключей можно также использовать специальных курьеров. Если в обмене секретными сообщениями планируется участие небольшого количества сторон, например двух или трех, то оба указанных способа вполне допустимы. Если же количество взаимодействующих абонентов велико, то задача распределения ключей превращается в настоящую проблему.

При использовании секретных ключей существуют и другие трудности. Например, ключи должны с определенной периодичностью меняться. Это связано с тем, что чем дольше используется ключ, тем больше вероятность его компрометации (раскрытия). Чем дольше используется ключ, тем больше потери от его компрометации, так как тем большее количество сообщений сможет раскрыть злоумышленник при получении ключа. Даже если ключ не будет раскрыт, проводить криптоанализ противнику удобнее, имея в своем распоряжении достаточное количество сообщений, зашифрованных одним и тем же ключом. Оптимальным считается использовать для каждого сеанса обмена зашифрованными сообщениями свой уникальный ключ, так называемый *сеансовый ключ*.

Таким образом, при большом числе взаимодействующих сторон требуется предварительная рассылка значительного количества ключей, а также последующее их хранение и при необходимости — смена.

На практике применяются специальные автоматизированные *системы управления ключами*. Такие системы позволяют генерировать ключи, хранить их и архивировать, восстанавливать уте-

рянные ключи, заменять или изымать из обращения старые и ненужные ключи. Важнейшей частью системы управления ключами является ***центр распределения ключей*** (Key Distribution Center — KDC), функциями которого являются генерация, распределение и передача ключей.

Специалистами разработаны специальные процедуры (или протоколы), которые позволяют центру распределения ключей доставлять пользователям ключи для проведения отдельных сеансов связи (сеансовые ключи).

Рассмотрим один из возможных протоколов обмена ключами. Предположим, при вступлении в сообщество пользователей сети обмена данными центром распределения ключей всем новым абонентам выдается индивидуальный секретный ключ. Вот как может выглядеть процедура распределения секретных ключей для проведения сеанса связи между двумя абонентами сети с использованием центра распределения ключей (для краткости будем называть его просто Центром):

1) абонент *A* обращается в Центр и запрашивает сеансовый ключ для связи с абонентом *B*;

2) в Центре создается случайный сеансовый ключ. Зашифровываются две копии этого сеансового ключа: одна с использованием секретного ключа абонента *A*, другая — с использованием секретного ключа абонента *B*. Затем обе зашифрованные копии пересылаются из Центра абоненту *A*;

3) абонент *A* расшифровывает свою копию сеансового ключа и пересылает вторую зашифрованную копию абоненту *B*;

4) абонент B расшифровывает свою копию сеансового ключа;

5) абоненты *A* и B используют полученный сеансовый ключ для секретного обмена информацией.

Указанный протокол достаточно прост и может быть автоматизирован с помощью, например, программы передачи данных. Однако приведенная процедура распределения сеансовых ключей имеет несколько явных недостатков. Первым недостатком данной системы является то, что Центр участвует во всех обменах, поэтому сбои в работе Центра нарушат работу всей системы. Второй недостаток заключается в том, что Центр должен хранить в какомлибо виде секретные ключи всех абонентов сети. Если злоумышленник найдет доступ к секретным ключам пользователей системы (взломает систему, подкупит администратора и т. д.), то он сможет читать и изменять все передаваемые сообщения.

И наконец, остается проблема первоначального распределения секретных ключей при вступлении пользователя в сеть. Первоначальный секретный ключ должен быть доставлен по абсолютно надежному каналу связи, иначе весь протокол теряет всякий смысл. Хорошо, если первоначальный ключ может быть выдан лично новому пользователю, однако в некоторых случаях это невозможно, например, при территориальной распределенности сети передачи данных.

Эти и другие недостатки алгоритмов симметричного шифрования обнаружились разработчиками телекоммуникационных сетей при первых попытках построения защищенных систем передачи данных в 70-х гг. XX в. Решением проблемы распределения ключей (а также некоторых других серьезных проблем) стало использование асимметричных алгоритмов шифрования.

4.6. АЛГОРИТМ ДИФФИ — ХЕЛЛМАНА

Первая публикация данного алгоритма появилась в 70-х гг. XX в. в статье Диффи и Хеллмана, в которой вводились основные понятия криптографии с открытым ключом, или криптографии на основе асимметричных алгоритмов шифрования. Алгоритм Диффи — Хеллмана предназначен для распределения ключей. Он позволяет двум или более пользователям обменяться без посредников ключом, который может быть использован затем для симметричного шифрования. Алгоритм основан на трудности вычислений дискретных логарифмов. В этом алгоритме, как и во многих других алгоритмах с открытым ключом, вычисления производятся по модулю некоторого большого простого числа *P*. Вначале специальным образом подбирается некоторое натуральное число *A*, меньшее *P*. Если надо зашифровать значение *X*, то вычисляем

$$Y = A^X \pmod{P}.$$

Причем, имея *X*, вычислить *Y* легко. Обратная задача вычисления *X* из *Y* является достаточно сложной. Экспонента *X* как раз и называется ***дискретным логарифмом*** *Y*. Таким образом, зная о сложности вычисления дискретного логарифма, число *Y* можно открыто передавать по любому каналу связи, так как при большом модуле *P* исходное значение *X* подобрать будет практически невозможно. на этом математическом факте основан алгоритм Диффи — Хеллмана для формирования ключа.

Пусть два пользователя, которых условно назовем «пользователь 1» и «пользователь 2», желают сформировать общий ключ для алгоритма симметричного шифрования. Вначале они должны вы-

брать большое простое число P и некоторое специальное число A, $1 < A < P–1$, такое, что все числа из интервала [1, 2, ..., $P–1$] могут быть представлены как различные степени A (mod P). Эти числа должны быть известны всем абонентам системы и могут выбираться открыто. Это будут так называемые общие параметры.

Затем первый пользователь выбирает число X_1 ($X_1 < P$), которое желательно формировать с помощью датчика случайных чисел. Это будет закрытый ключ первого пользователя, и он должен держаться в секрете. на основе закрытого ключа пользователь 1 вычисляет число

$$Y_1 = A^X{}_1 \pmod{P},$$

которое он посылает второму абоненту. Аналогично поступает и второй пользователь, генерируя X_2 и вычисляя

$$Y_2 = A^X{}_2 \pmod{P},$$

Это значение пользователь 2 отправляет первому пользователю. После этого у пользователей должна быть информация, указанная в табл. 4.45.

Из чисел Y_1 и Y_2, а также своих закрытых ключей каждый из абонентов может сформировать общий секретный ключ Z для сеанса симметричного шифрования. Вот как это должен сделать первый пользователь:

$$Z = (Y_2)^X{}_1 \pmod{P}.$$

Никто другой, кроме пользователя 1, этого сделать не может, так как число X_1 секретно. Второй пользователь может получить то же самое число Z, используя свой закрытый ключ и открытый ключ своего абонента следующим образом:

$$Z = (Y_1)^X{}_2 \pmod{P}.$$

Если весь протокол формирования общего секретного ключа выполнен верно, значения Z у одного и второго абонента должны получиться одинаковыми. Причем, что самое важное, злоумышленник, не зная секретных чисел X_1 и X_2, не сможет вычислить

Таблица 4.5. Информация, содержащаяся у абонентов

	Общие параметры	Открытый ключ	Закрытый ключ
Пользователь 1	P, A	Y_1	X_1
Пользователь 2		Y_2	X_2

число Z. Не зная X_1 и X_2, злоумышленник может попытаться вычислить Z, используя только передаваемые открыто P, A, Y_1 и Y_2.

Безопасность формирования общего ключа в алгоритме Диффи — Хеллмана вытекает из того факта, что, хотя относительно легко вычислить экспоненты по модулю простого числа, очень трудно вычислить дискретные логарифмы. Для больших простых чисел размером сотни и тысячи бит задача считается неразрешимой, так как требует колоссальных затрат вычислительных ресурсов.

4.7. УПРАВЛЕНИЕ КЛЮЧАМИ. ПРОТОКОЛ KERBEROS

Первоначально протокол Kerberos (русск. — *протокол Цербер*) был разработан в Массачусетсом технологическом институте (США) для *проекта Athena*. Протокол Kerberos спроектирован для работы в сетях с протоколом TCP/IP и предполагает участие в аутентификации и распределении ключей третьей доверенной стороны. Служба Kerberos, работающая в сети, действует как доверенный посредник, обеспечивающий надежную аутентификацию в сети, разрешая законному пользователю доступ к различным компьютерам в сети. Протокол Kerberos основывается на симметричной криптосистеме (реализован алгоритм DES, хотя возможно применение и других симметричных криптоалгоритмов).

Основной протокол Kerberos является вариантом протокола аутентификации и распределения ключей Нидхема — Шрёдера. В основном протоколе Kerberos участвуют две взаимодействующие стороны *A и B* и доверенный сервер *KS* (Kerberos Server). Стороны *A* и *B*, каждая по отдельности, разделяют свой секретный ключ с сервером *KS*. Доверенный сервер *KS* выполняет роль центра распределения ключей (ЦРК). Схема протокола Kerberos представлена на рис. 4.6.

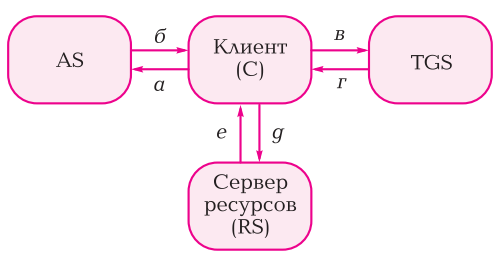

Рис. 4.6. Схема протокола Kerberos

Основные компоненты *протокола аутентификации Kerberos*:

C — клиент;

(*KS* = *AS* + *TGS*) — сервер системы Kerberos;

AS — сервер аутентификации;

TGS — сервер выдачи разрешений (предоставления доступа к сетевым ресурсам);

RS — сервер ресурсов (в сети).

Рассмотрим на примере работу протокола Kerberos. Пусть сторона *A* хочет получить сеансовый ключ для информационного обмена со стороной *B*. Сторона *A* инициирует фазу распределения ключей, посылая по сети серверу *KS* идентификаторы id_A и id_B.

Сервер *KS* генерирует сообщение с временной отметкой *T*, сроком действия мандата *L* (длительностью сеанса), случайным сеансовым ключом K и идентификатором id_A. *KS* шифрует это сообщение секретным ключом, который разделяет со стороной *B*. Затем сервер *KS* берет временную отметку *T*, срок действия *L*, сеансовый ключ *K*, идентификатор id_B стороны B и шифрует все это секретным ключом, который разделяет со стороной *A*. Оба эти зашифрованные сообщения он отправляет стороне *A*.

Сторона *A* расшифровывает первое сообщение своим секретным ключом, проверяет отметку времени *T*, чтобы убедиться, что это сообщение не является повторением предыдущей процедуры распределения ключей.

Затем сторона *A* генерирует сообщение со своим идентификатором id_A и отметкой времени *T*, шифрует его сеансовым ключом K и отправляет стороне *B*. Кроме того, *A* отправляет для B сообщение от *KS*, зашифрованное ключом стороны *B*.

Только сторона B может расшифровать сообщения (в). Сторона B получает (после расшифрования) отметку времени *T*, срок действия *L*, сеансовый ключ K и идентификатор id_A. Затем сторона B расшифровывает сеансовым ключом K вторую часть сообщения. Совпадение значений *T* и id_A в двух частях сообщения подтверждают подлинность *A* по отношению к *B*.

Для взаимного подтверждения подлинности сторона B создает сообщение, состоящее из отметки времени *T* плюс 1, шифрует его ключом K и отправляет стороне *A*.

Если после расшифрования сообщения сторона *A* получает ожидаемый результат, она знает, что на другом конце линии связи находится действительно *B*.

Этот протокол успешно работает при условии, что часы каждого участника синхронизированы с часами сервера *KS*. Следует отметить, что в этом протоколе необходим обмен с *KS* для получения сеансового ключа каждый раз, когда *A* желает установить связь с *B*. Протокол обеспечивает надежное соединение объектов *A* и B при условии, что ни один из ключей не скомпрометирован и сервер *KS* защищен.

Система Kerberos обеспечивает защиту сети от несанкционированного доступа, базируясь исключительно на программных решениях, и предполагает многократное шифрование передаваемой по сети управляющей информации. Система Kerberos имеет структуру типа «клиент — сервер» и состоит из клиентских частей *C*, установленных на все машины сети (рабочие станции пользователей и серверы), и Kerberos-сервера *KS*, располагающегося на каком-либо (не обязательно выделенном) компьютере.

ВОПРОСЫ И ЗАДАНИЯ ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ ЗНАНИЙ

1. Назовите виды криптосистем. Охарактеризуйте каждую из них.
2. В чем заключается проблема хранения симметричных ключей шифрования?
3. Какими способами алгоритмы симметричного шифрования могут обрабатывать исходный текст?
4. Перечислите основные требования к блочному алгоритму шифрования.
5. Какой стандарт принят в России в качестве стандарта на блочные алгоритмы шифрования?
6. Какие режимы шифрования данных предусматривает ГОСТ 28147—89?
7. Назовите, в чем состоит отличие режима гаммирования с обратной связью от режима гаммирования.
8. Дайте характеристику криптографическому алгоритму «Магма».
9. Какие вы знаете зарубежные криптографические алгоритмы?
10. Опишите общую структуру криптографического алгоритма DES.
11. Какие проблемы возникают при распределении ключей симметричного шифрования?
12. Дайте определение понятия «сеансовый ключ».
13. Каково основное назначение алгоритма Диффи — Хеллмана?
14. Опишите процесс передачи ключей в соответствии с алгоритмом Диффи — Хеллмана.
15. Какие стороны участвуют в основном протоколе Kerberos?
16. Назовите основные этапы протокола Kerberos.

Глава 5

АСИММЕТРИЧНЫЕ СИСТЕМЫ ШИФРОВАНИЯ

5.1. СИСТЕМЫ ШИФРОВАНИЯ С ОТКРЫТЫМ КЛЮЧОМ. НЕОБРАТИМОСТЬ СИСТЕМ

При использовании симметричного шифрования, как указывалось ранее, возникают две достаточно серьезные проблемы. Первая проблема заключается в изготовлении секретных ключей и доставке их участникам информационного обмена. Второй проблемой является обеспечение подлинности партнеров при электронном общении.

Для решения этих проблем были предложены асимметричные алгоритмы шифрования. Асимметричные алгоритмы шифрования называются также алгоритмами с открытым ключом. В отличие от алгоритмов симметричного шифрования (алгоритмов шифрования с закрытым ключом), в которых для шифрования и расшифрования используется один и тот же ключ, в асимметричных алгоритмах один ключ используется для шифрования, а другой, отличный от первого, — для расшифрования. Алгоритмы называются асимметричными, так как ключи шифрования и расшифрования разные, следовательно, отсутствует симметрия основных криптографических процессов. Один из двух ключей является открытым (public key) и может быть объявлен всем, а второй — закрытым (private key) и должен держаться в секрете. Какой из ключей, открытый или закрытый, используется для шифрования, а какой для расшифрования, определяется назначением криптографической системы.

Алгоритмы шифрования с открытым ключом можно использовать для решения, как минимум, трех задач:

1) для шифрования передаваемых и хранимых данных в целях их защиты от несанкционированного доступа;

2) формирования электронной подписи под электронными документами;

3) распределения секретных ключей, используемых потом при шифровании документов симметричными методами.

Все алгоритмы шифрования с открытым ключом основаны на использовании так называемых необратимых, или односторонних, функций. ***Односторонней функцией*** (one-way function) называется математическая функция, которую относительно легко вычислить, но трудно найти по значению функции соответствующее значение аргумента, т. е., зная x, легко вычислить $f(x)$, но по известному $f(x)$ трудно найти подходящее значение x. Под словом «трудно вычислить» понимают, что для этого потребуется не один год расчетов с использованием компьютеров. Односторонние функции применяются в криптографии также в качестве хеш-функций. Использовать односторонние функции для шифрования сообщений с целью их защиты не имеет смысла, так как обратно расшифровать зашифрованное сообщение уже не получится. Для целей шифрования используются специальные односторонние функции — *односторонние функции с люком*, или с *секретом* (особый вид односторонних функций, имеющих некоторый секрет (люк), позволяющий относительно быстро вычислить обратное значение функции).

Для односторонней функции с секретом f справедливы следующие утверждения:

- зная x, легко вычислить $f(x)$;
- по известному значению $f(x)$ трудно найти x;
- зная дополнительно некоторую секретную информацию, можно легко вычислить x.

5.2. СТРУКТУРНАЯ СХЕМА ШИФРОВАНИЯ С ОТКРЫТЫМ КЛЮЧОМ

Схема шифрования с открытым ключом основана на использовании двух разных ключей, хотя и связанных между собой, но устроенных так, что вычислить по одному из них (открытому) другой (закрытый) практически невозможно. Закрытый ключ должен быть известен только одному лицу — его владельцу. Такой принцип использования асимметричных алгоритмов получил название открытого шифрования (или шифрованием с открытым ключом). Согласно этому принципу, любой желающий может зашифровать сообщение открытым ключом. Расшифровать сообщение сможет только владелец закрытого ключа. Пусть, например, пользователи A и B, имеющие возможность обмениваться электронными сооб-

щениями, используют схему открытого шифрования. Предположим, пользователь A должен передать секретное сообщение пользователю B так, чтобы никто другой не смог его прочитать. Для этого необходимо выполнить следующие действия:

1) пользователь B посылает пользователю A свой открытый ключ U по любому каналу связи, например, по электронной почте;

2) пользователь A шифрует свое сообщение M полученным открытым ключом U и получает зашифрованное сообщение C;

3) зашифрованное сообщение C пересылается пользователю B;

4) пользователь B расшифровывает полученное сообщение C своим закрытым ключом R.

Использование открытого шифрования снимает проблему распределения ключей. Теперь абоненты не должны заботиться о возможности компрометации секретного ключа. Пользователи системы связи могут совершенно свободно обмениваться открытыми ключами и зашифрованными ими сообщениями. Если пользователь надежно хранит свой закрытый ключ, никто не сможет прочитать передаваемые сообщения.

Для упрощения процедуры обмена в сети передачи сообщений обычно используется база данных, в которой хранятся открытые ключи всех пользователей. При необходимости любой пользователь системы может запросить из базы открытый ключ другого человека и использовать полученный ключ для шифрования сообщений.

5.3. ОТЕЧЕСТВЕННЫЕ СТАНДАРТЫ АСИММЕТРИЧНОГО ШИФРОВАНИЯ

ГОСТ Р 34.10—94. В ГОСТ Р 34.10—94 «Информационная технология. Криптографическая защита информации. Процедуры выработки и проверки электронной подписи на базе асимметричного криптографического алгоритма», вступившем в силу в 1995 г., используются следующие параметры:

p — большое простое число длиной от 509—512 бит либо от 1 020—1 024 бит;

q — простой сомножитель числа $(p - 1)$, имеющий длину 254—256 бит;

a — любое число, a $< (p - 1)$, $a^q \pmod p = 1$;

p, q, a являются открытыми и используются всеми пользователями сети;

x — некоторое число, $x < q$, и является секретным ключом;

$y = a^x \pmod p$ является открытым ключом.

Алгоритм использует однонаправленную хеш-функцию $h(x)$ ГОСТ Р 34.11—94. Для передачи документа M отправитель:

1) генерирует случайное секретное число $k < q$;

2) вычисляет значения $r = (a^k \bmod p) \pmod q$ и $s = (xr + k(h(M))) \pmod q$.

Если $r = 0$, то k присваивают другое значение и начинают сначала, если $h(M) \pmod q = 0$, то $h(M) \pmod q$ принимают равным 1;

3) документ M и электронная подпись, представляющая пару $(r \pmod{2^{116}}, s \pmod{2^{116}})$, отправляются по сети получателю.

Получатель вычисляет:

$$u_1 = [(h(\mathrm{m})^{-1} \pmod q)\ s \pmod q],$$

$$u_2 = [(h(\mathrm{m})^{-1} \pmod \mathrm{q})\ (q - r) \pmod q],$$

$$v = [(a^{u_1} y^{u_2}) \pmod p] \pmod q;$$

4) подпись корректна, если $v = r$.

Подпись, созданная с использованием стандарта ГОСТ Р 34.10—94 называется *рандомизированной*, так как для одного и того же сообщения с использованием одного и того же закрытого ключа каждый раз будут создаваться разные подписи (r, s) из-за нового значения числа k.

ГОСТ Р 34.10—2001. ГОСТ Р 34.10—2001 разрабатывался взамен ГОСТ Р 34.10—94 для обеспечения большей стойкости алгоритма. ГОСТ Р 34.10—2001 основан на эллиптических кривых. Его стойкость основывается на сложности вычисления дискретного логарифма в группе точек эллиптической кривой, а также на стойкости хеш-функции по ГОСТ Р 34.11—94. Принцип подписания электронного документа заключается в шифровании по ГОСТ Р 34.10—2001 полученного хеш алгоритма ГОСТ Р 34.11—94 закрытым ключом на передающей стороне. на рис. 5.1 показана схема формирования электронной подписи.

После подписывания сообщения M к нему дописывается электронная подпись размером 512 или 1 024 бит и текстовое поле. В текстовом поле могут содержаться, например, дата и время отправки или различные данные об отправителе:

| Сообщение M | + | Цифровая подпись | Текст |

Данный алгоритм не описывает механизм генерации параметров, необходимых для формирования подписи, а только определяет, каким образом на основании таких параметров получить

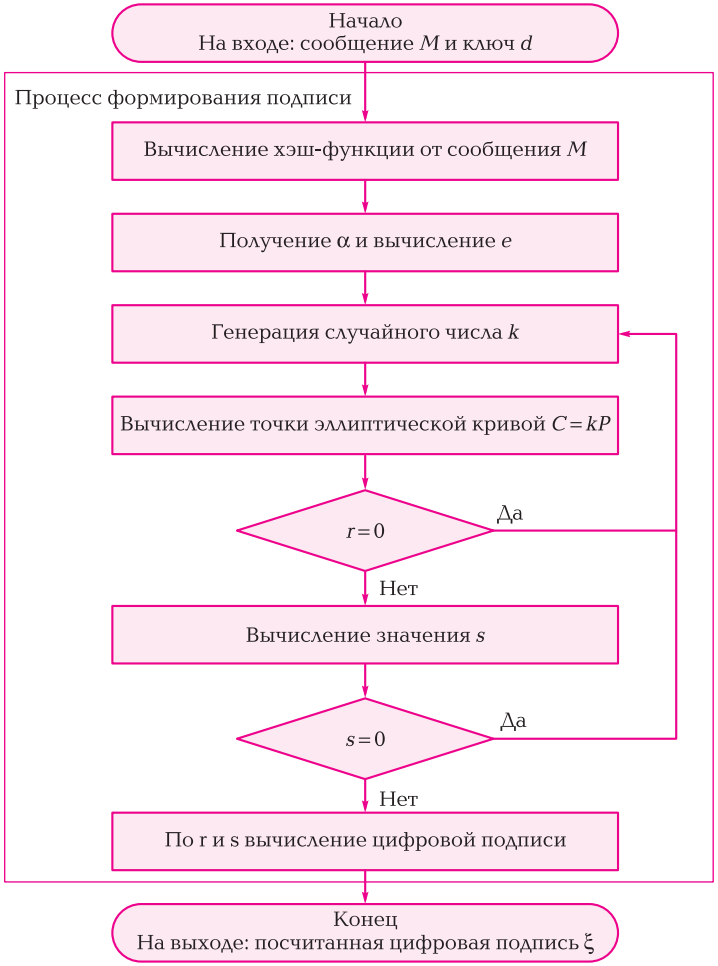

Рис. 5.1. Схема получения электронной подписи

электронную подпись. Механизм генерации параметров определяется на месте в зависимости от разрабатываемой системы.

Проверка подписи на приемной стороне осуществляется путем получения хеша от сообщения и расшифрование открытым ключом зашифрованного значения хеша, переданного вместе с сообщением: если эти хеши равны, то подпись верна. на рис. 5.2 показана схема проверки электронной подписи.

ГОСТ Р 34.10—2012. Стандарт ГОСТ Р 34.10—2012 использует ту же схему формирования электронной подписи, что и ГОСТ Р

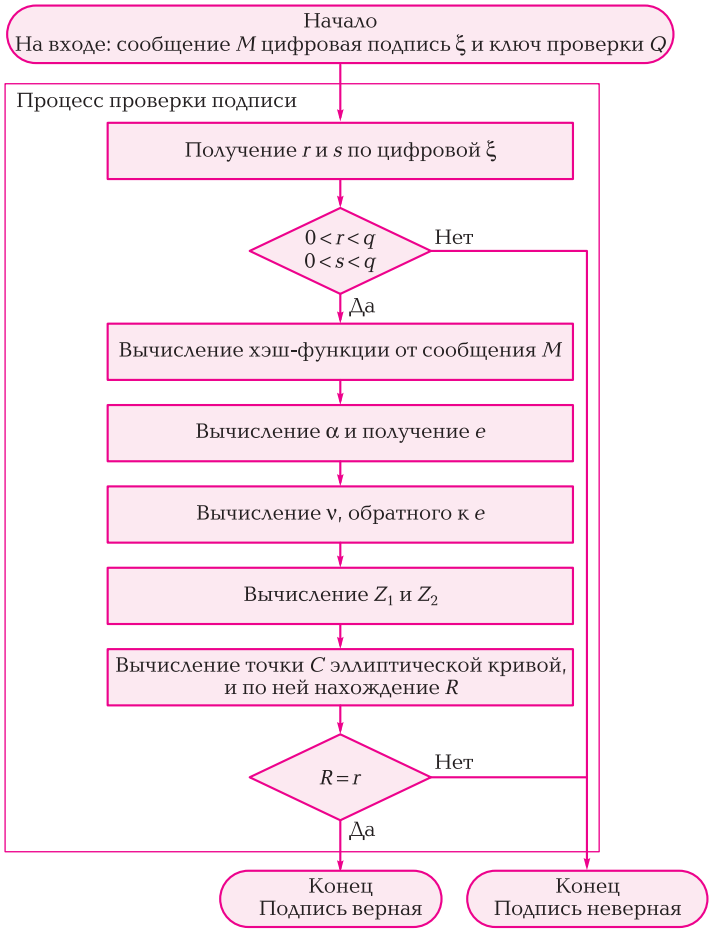

Рис. 5.2. Схема проверки цифровой подписи

34.10—2001. Новый стандарт отличается наличием дополнительного варианта параметров схем (соответствующего длине секретного ключа порядка 512 бит) и требованием использования функций хеширования ГОСТ Р 34.11—2012: первый вариант требований к параметрам (такой же, как в ГОСТ Р 34.10—2001, соответствующий длине секретного ключа порядка 256 бит) предусматривает использование хеш-функции с длиной хеш-кода 256 бит, дополнительный вариант требований к параметрам предусматривает использование хеш-функции с длиной хеш-кода 512 бит.

5.4. СИСТЕМА ШИФРОВАНИЯ RSA

Алгоритм шифрования с открытым ключом RSA был предложен одним из первых в конце 70-х гг. XX в. Его название составлено из первых букв фамилий авторов: Р. Райвеста (R. Rivest), А. Шамира (A. Shamir) и Л. Адлемана (L. Adleman). Алгоритм RSA является, наверное, наиболее популярным и широко применяемым асимметричным алгоритмом в криптографических системах. Криптографическая система *RSA* базируется на следующих двух фактах из теории чисел:

1) задача проверки числа на простоту является сравнительно легкой;

2) задача разложения чисел вида $n = pq$ (p и q — простые числа) на множители является очень трудной, если известно только n, а p и q — большие числа.

Алгоритм RSA представляет собой блочный алгоритм шифрования, где зашифрованные и незашифрованные данные должны быть представлены в виде целых чисел между 0 и n –1 для некоторого n.

Пусть абонент A хочет передать зашифрованное сообщение абоненту B. В этом случае абонент В должен подготовить пару (открытый ключ, закрытый ключ) и отправить свой открытый ключ пользователю A.

Для этого вначале выбираются два больших простых числа P и Q. Затем вычисляется произведение N:

$$N = PQ.$$

После этого определяется вспомогательное число f:

$$f = (P - 1)(Q - 1).$$

Затем случайным образом выбирается число $d < f$ и взаимно простое с f. Далее необходимо найти число e такое, что

$$ed \pmod f = 1.$$

Числа d и N будут открытым ключом пользователя, а значение e — закрытым ключом.

Таким образом, на этом этапе у пользователя должны быть сформированы открытый ключ (пара чисел N и d) и закрытый ключ (число e).

Так как пользователь В хочет получить зашифрованное сообщение от пользователя A, значит, пользователь В должен отправить свой открытый ключ (d, N) пользователю A. Числа P и Q больше не нужны, однако их нельзя никому сообщать; лучше всего их вообще забыть.

На этом этап подготовки ключей закончен и можно использовать основной протокол RSA для шифрования данных.

Второй этап — шифрование данных. Если абонент A хочет передать некоторые данные абоненту B, он должен представить свое сообщение в цифровом виде и разбить его на блоки m_1, m_2, m_3, ... , где $m_i < N$. Зашифрованное сообщение будет состоять из блоков c_i.

Абонент A шифрует каждый блок своего сообщения по формуле

$$c_i = m_i^d \pmod N,$$

используя открытые параметры пользователя B, и пересылает зашифрованное сообщение C = (c_1, c_2, c_3, ...) по открытой линии.

Абонент B, получивший зашифрованное сообщение, расшифровывает все блоки полученного сообщения по формуле

$$m_i = c^e \pmod N.$$

Все расшифрованные блоки будут точно такими же, как и исходящие от пользователя A.

Злоумышленник, перехватывающий все сообщения и знающий всю открытую информацию, не сможет найти исходное сообщение при больших значениях P и Q.

Для обеспечения высокой надежности шифрования необходимо, чтобы выступающее в качестве модуля число N было очень большим — несколько сотен или тысяч бит. Только в этом случае будет практически невозможно по открытым параметрам определить закрытый ключ.

5.5. СИСТЕМА ШИФРОВАНИЯ ЭЛЬ-ГАМАЛЯ

Асимметричный алгоритм, предложенный в 1985 г. Эль-Гамалем (T. ElGamal), универсален. Он может быть использован для решения всех трех основных задач: для шифрования данных; формирования электронной подписи и согласования общего ключа. Кроме того, возможны модификации алгоритма для схем проверки пароля, доказательства идентичности сообщения и другие варианты. Безопасность этого алгоритма так же, как и алгоритма Диффи — Хеллмана, основана на трудности вычисления дискретных логарифмов.

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

целое число P и большое целое число G, причем $G < P$. Отправитель и получатель подписанного документа используют при вычислениях одинаковые большие целые числа P (~10^{308} или ~2^{1024}) и G (~10^{154} или ~2^{512}), которые не являются секретными.

Отправитель выбирает случайное целое число X, $1 < X < (P - 1)$, и вычисляет

$$Y = G^x \pmod{P}.$$

Число Y является открытым ключом, используемым для проверки подписи отправителя. Число Y открыто передается всем потенциальным получателям документов.

Число X является секретным ключом отправителя для подписывания документов и должно храниться в секрете.

Для того чтобы подписать сообщение M, сначала отправитель хеширует его с помощью хеш-функции $h(M)$ в целое число m:

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

и генерирует случайное целое число K, $1 < K < (P - 1)$, такое, что K и $(P - 1)$ являются взаимно простыми. Затем отправитель вычисляет целое число a:

$$a = G^K \pmod{P}$$

и, применяя расширенный алгоритм Евклида, вычисляет с помощью секретного ключа X целое число b из уравнения

$$m = Xa + Kb \pmod{(P - 1)}.$$

Пара чисел (a, b) образует электронную подпись S:

$$S = (a, b),$$

проставляемую под документом M.

Тройка чисел (M, a, b) передается получателю, в то время как пара чисел (X, K) держится в секрете.

После приема подписанного сообщения (M, a, b) получатель должен проверить, соответствует ли подпись $S = (a, b)$ сообщению M. Для этого получатель сначала вычисляет по принятому сообщению M число

$$m = h(M),$$

т. е. хеширует принятое сообщение M.

Затем получатель вычисляет значение $A = Y^a a^b \pmod{P}$ и признает сообщение M подлинным, если и только если

$$A = G^m \pmod{P}.$$

Иначе говоря, получатель проверяет справедливость соотношения

$$Y^a a^b \pmod{P} = G^m \pmod{P}.$$

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

Следует отметить, что выполнение каждой подписи по методу Эль-Гамаля требует нового значения K, причем это значение должно выбираться случайным образом. Если злоумышленник раскроет когда-либо значение K, повторно используемое отправителем, то он сможет раскрыть секретный ключ X отправителя. Таким образом, K — сессионный ключ.

Схема цифровой подписи Эль-Гамаля имеет ряд преимуществ по сравнению со схемой цифровой подписи RSA. При заданном уровне стойкости алгоритма электронной подписи целые числа, участвующие в вычислениях, имеют запись на 25 % короче, что уменьшает сложность вычислений почти в два раза и позволяет заметно сократить объем используемой памяти.

При выборе модуля P достаточно проверить, что это число является простым и что у числа $(P - 1)$ имеется большой простой множитель (т. е. всего два достаточно просто проверяемых условия).

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

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

5.6. КРИПТОСИСТЕМЫ НА ОСНОВЕ МЕТОДА ЭЛЛИПТИЧЕСКИХ КРИВЫХ

В 1985 г. американские ученые Н. Коблиц (Neal Koblitz) и В. Миллер (Victor Miller) предложили использовать для криптосистем с открытым ключом теорию эллиптических кривых. Даль-

нейшие исследования подтвердили наличие подходящих свойств у этих математических функций и привели к созданию реальных криптографических систем, использующих математический аппарат эллиптических кривых. С 1998 г. использование эллиптических кривых для решения криптографических задач таких, как электронная подпись, было закреплено в стандартах США ANSI X9.62 и FIPS 186-2, а в 2001 г. аналогичный стандарт, ГОСТ Р 34.10—2001, был принят и в России.

Основное достоинство криптосистем на эллиптических кривых состоит в том, что по сравнению с другими асимметричными криптосистемами, рассмотренными ранее, они обеспечивают существенно более высокую криптостойкость при равных затратах на обработку и вычисления. Это объясняется тем, что вычисление обратных функций на эллиптических кривых значительно сложнее, чем, например, вычисление дискретных логарифмов (алгоритмы Диффи — Хеллмана и Эль-Гамаля) или решение задачи факторизации (алгоритм RSA). В результате тот уровень стойкости, который достигается, скажем, в RSA при использовании 1 024-битовых модулей, в системах на эллиптических кривых реализуется при размере модуля 160 бит, что обеспечивает более простую как программную, так и аппаратную реализацию.

В криптографии используются эллиптические кривые на плоскости, определяемые уравнениями вида

$$Y^2 = X^3 + aX + b \pmod{p},$$

где p — некоторое большое простое число, а a и b — константы.

Принцип использования эллиптических кривых следующий. Для группы пользователей выбирается общая эллиптическая кривая E и некоторая точка G на ней. Закрытым ключом пользователя выступает некоторое целое число c, а открытым — точка D на кривой E, полученная в результате специального преобразования композиции с использованием числа c. Параметры кривой и список открытых ключей абонентов, как и обычно, передаются всем пользователям сети. Открытые и закрытые ключи пользователей используются для выполнения операций шифрования и расшифрования в зависимости от назначения алгоритма.

С помощью эллиптических кривых могут быть реализованы многие известные протоколы с открытым ключом. Любая криптосистема, основанная на дискретном логарифмировании, легко может быть перенесена на эллиптические кривые.

Несмотря на сложность математического аппарата эллиптических кривых, существуют эффективные вычислительные методы,

позволяющие достаточно быстро реализовывать необходимые расчеты. За счет использования модуля меньшей длины операции генерации ключей и шифрования выполняются быстрее, чем, скажем, в алгоритме RSA или классическом алгоритме Диффи — Хеллмана. Криптографические методы на эллиптических кривых считаются перспективными и, закрепленные в различных стандартах, находят применение в современных системах защиты информации.

5.7. БЕЗОПАСНОСТЬ АСИММЕТРИЧНЫХ СИСТЕМ ШИФРОВАНИЯ

Как и в случае симметричных криптографических систем, с помощью асимметричных криптосистем обеспечивается не только конфиденциальность, но также подлинность и целостность передаваемой информации. Подлинность и целостность любого сообщения обеспечивается формированием электронной подписи этого сообщения и отправкой в зашифрованном виде сообщения вместе с электронной подписью. Проверка соответствия подписи полученному сообщению после его предварительного расшифровывания представляет собой проверку целостности и подлинности принятого сообщения.

Асимметричные криптосистемы обладают следующими характерными особенностями:

1) открытый ключ K_B и криптограмма C могут быть отправлены по незащищенным каналам, т. е. злоумышленнику известны K_B и C;

2) алгоритмы зашифрования и расшифрования являются открытыми.

Безопасность асимметричной криптосистемы обеспечивается выполнением следующих требований:

1) вычисление пары ключей (K_B, k_B) получателем B на основе начального условия должно быть простым;

2) отправитель A, зная открытый ключ K_B и сообщение M, может легко вычислить криптограмму

$$C = E_{KB}(M);$$

3) получатель B, используя секретный ключ k_B и криптограмму C, может легко восстановить исходное сообщение

$$M = D_{kB}(C);$$

4) злоумышленник, зная открытый ключ K_B, при попытке вычислить секретный ключ k_B наталкивается на непреодолимую вычислительную проблему;

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

Концепция асимметричных криптографических систем с открытым ключом основана на применении *однонаправленных функций*. Вторым важным классом функций, используемых при построении криптосистем с открытым ключом, являются *однонаправленные функции с секретом*.

ВОПРОСЫ И ЗАДАНИЯ ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ ЗНАНИЙ

1. Для решения каких проблем используют алгоритмы шифрования с открытым ключом?
2. Дайте определение односторонней функции.
3. Опишите алгоритм передачи сообщения с использованием схемы открытого шифрования.
4. Какие параметры использует ГОСТ Р 34.10—94?
5. Для чего используется ГОСТ Р 34.10—2001?
6. В чем отличие алгоритма ГОСТ Р 34.10—2001 от алгоритма ГОСТ Р 34.10—94?
7. Какова длина ключа у алгоритма ГОСТ Р 34.10—2012?
8. Кто создал алгоритм шифрования RSA?
9. На каких факторах основана криптостойкость алгоритма RSA?
10. Напишите формулу для зашифрования блока сообщения для алгоритма RSA.
11. Опишите параметры алгоритма Эль-Гамаля.
12. По какой формуле вычисляется открытый ключ для системы Эль-Гамаля?
13. Напишите формулы для вычисления электронной подписи системы Эль-Гамаля.
14. Как осуществляется проверка электронной подписи системы Эль-Гамаля?
15. Какие стандарты основаны на эллиптических кривых?
16. Опишите принцип использования эллиптических кривых в криптографии.
17. Перечислите требования безопасности асимметричных криптосистем.

Глава 6

КРИПТОАНАЛИЗ

6.1. ОСНОВНЫЕ МЕТОДЫ КРИПТОАНАЛИЗА

Криптоанализом (от греч. κρυπτός — скрытый и *analein* — ослаблять, избавлять) называют науку восстановления (дешифрования) открытого текста без доступа к ключу. Хотя история криптоанализа насчитывает многие века, особенно интенсивно эта область знаний начала развиваться с наступлением компьютерной эры.

Задача криптоанализа состоит в том, чтобы определить вероятность взлома шифра и, таким образом, оценить его применимость в той или иной области.

Попытка криптоанализа называется ***атакой***. Криптоанализ ставит своей задачей в разных условиях получить дополнительные сведения о ключе шифрования, чтобы значительно уменьшить диапазон вероятных ключей. Результаты криптоанализа могут варьироваться по степени практической применимости. Так, криптограф Ларс Кнудсен предлагает следующую *классификацию успешных исходов* криптоанализа блочных шифров в зависимости от объема и качества секретной информации, которую удалось получить:

- полный взлом — криптоаналитик извлекает секретный ключ;
- глобальная дедукция — криптоаналитик разрабатывает функциональный эквивалент исследуемого алгоритма, позволяющий зашифровывать и дешифровывать информацию без знания ключа;
- частичная дедукция — криптоаналитику удается дешифровать или зашифровать некоторые сообщения;
- информационная дедукция — криптоаналитик получает некоторую информацию об открытом тексте или ключе.

Взлом шифра не всегда подразумевает обнаружение способа, применимого на практике для восстановления открытого текста по перехваченному зашифрованному сообщению. Допустим, для дешифрования текста методом полного перебора требуется перебрать N возможных ключей. Тогда изобретение способа, требующего для дешифрования операций по подбору ключа, будет считаться взломом. В научной криптологии под ***взломом*** понимается лишь подтверждение наличия уязвимости криптоалгоритма, свидетельствующее, что свойства шифра не соответствуют заявленным характеристикам. Как правило, криптоанализ начинается с попыток взлома упрощенной модификации алгоритма, после чего результаты распространяются на полноценную версию.

В табл. 6.1 современные методы криптоанализа систематизированы по применимости для взлома различных категорий криптосистем. Знак «+» означает, что данный метод взлома применим к указанной криптосистеме.

6.2. КРИПТОГРАФИЧЕСКИЕ АТАКИ

Атака грубой силы. С появлением высокопроизводительной вычислительной техники у криптоаналитиков появилась возможность вскрывать шифры *методом перебора ключей*, или так называемой атакой грубой силы.

Пусть (x, y) означают пару открытого и зашифрованного текста и пусть $K = (k_1, ..., k_k)$ — множество всех возможных ключей. Атака на основе полного перебора проверяет для каждого $k_i \in K$ выполнение равенства: $D_{ki}(y) = x$. Если равенство выполняется — правильный ключ найден; если не выполняется — проверяется следующий ключ. на практике метод грубой силы может быть сложнее, так как неправильные ключи могут дать неверные положительные результаты.

При осуществлении попытки атаки на основе только шифротекста криптоаналитику требуется анализировать выходные данные алгоритма и проверять их «осмысленность». В случае, когда в качестве объекта шифрования выступает графический файл или программа, задача определения «осмысленности» выходных данных становится очень трудной.

Если известно, что открытый текст представляет собой предложение на естественном языке, проанализировать результат и опознать успешный исход дешифрования сравнительно несложно, тем более что криптоаналитик зачастую располагает некоторой априорной информацией о содержании сообщения.

Таблица 6.1. Методы криптоанализа

Вид криптосистем	Вид атаки									
	Частотный анализ	Полный перебор ключей	Анализ ключевого генератора	Факторизация / дискретное логарифмирование	Метод «встре-чи по середине»	Разностный анализ	Линейный анализ	Метод коллизий	Анализ по побочным каналам	Квантовый анализ
На основе хеш-функции		+			+	+		+		
Асимметричные		+	+	+	+				+	+
Симметричные	+	+	+			+	+		+	

Задачу выделения осмысленного текста, т. е. определения факта правильной дешифрации, решают с помощью ПК с использованием теоретических положений, разработанных в конце XIX в. петербургским математиком А. А. Марковым — *цепей Маркова*.

При проведении атаки на основе шифротекста, возможны следующие варианты:

- если открытый текст является осмысленным текстом на каком-либо языке, перехваченный шифротекст должен иметь достаточный размер для однозначного расшифрования в осмысленный текст (минимально достаточный для этого размер называется расстоянием единственности). В основополагающей для современных симметричных криптосистем работе размер единственности для английского языка теоретически определен как 27 букв. Если сообщение короче, то при переборе возможно появление нескольких различных осмысленных текстов, каждому из которых соответствует некий кандидат в искомые ключи. При невозможности перехвата дополнительных шифротекстов невозможно определить, какое из осмысленных сообщений является верным, если это не ясно из контекста;
- если открытый текст является бинарными данными, необходима какая-либо информация о том, что он из себя представляет. Если перехватывается архив, то при переборе ключей каждое значение M' должно рассматриваться как возможный заголовок архива. При другом потенциальном M это может быть заголовок исполняемого файла для Windows, заголовок графического файла и т. д.;

- многие средства шифрования информации внедряют в формат зашифрованного объекта контрольную сумму открытого текста для проверки его целостности после расшифрования. Это может быть, например, имитовставка, вычисленная согласно отечественному криптостандарту ГОСТ Р 28147—89, или просто код CRC32. Главное, что такая контрольная сумма может быть идеальным эталоном при криптоанализе, вполне подходящим для определения верного ключа.

Метод грубой силы часто используется совместно с другими методами. Например, после раскрытия части ключа с помощью дифференциального криптоанализа может стать возможен перебор оставшейся части.

Частотный анализ. Метод частотного анализа известен с еще IX в. и связан с именем Ал-Кинди. Но наиболее известным случаем применения такого анализа является дешифровка египетских иероглифов Ж.-Ф. Шампольоном в 1822 г.

Частотный анализ предполагает, что каждая буква алфавита того или иного языка в довольно длинном тексте встречается с определенной частотой; к примеру, для русского языка известно, что буквы «О», «П», «Р» встречаются очень часто, а вот «Й», «Ъ» — редко. Данные характеристики переносятся из открытого текста на зашифрованный.

Пусть имеется зашифрованный текст, полученный методом какой-либо перестановки букв по определенному алгоритму, и аналитикам требуется его расшифровать. Для этого берется открытый текст, желательно довольно длинный, затем подсчитывается в нем частота каждой буквы, причем чем больше будет текст, тем точнее получится расшифровка. Следующий шаг — то же самое проделывается с зашифрованным текстом, подсчитывается частота каждого символа. Собственно говоря, весь процесс расшифровки сводится к тому, что сопоставляются частоты двух текстов. Например, в открытом тексте буква «О» встречается с частотой 33 %, т. е. от общего количества букв текста, буква «О» составляет 33 %, а в зашифрованном тексте с частотой 33 % встречается буква «П», значит, с большей вероятностью под буквой «П» подразумевается «О». Далее в соответствии с частотами заменяем буквы в позициях текста и все повторяем снова. Так действуем до тех пор, пока не получим читаемый текст.

Поскольку метод частотного анализа основан на вероятностях, то не всегда возможное появление буквы в тексте будет соответствовать общепринятой частоте появления букв. Также успех зачастую зависит от размера самого текста: чем он больше — тем выше вероятность определить верную частоту букв.

Атака на основе знания открытого текста. Атака на основе открытых текстов (англ. *known-plaintext attack*) — вид криптоанализа, при котором в шифротексте присутствуют стандартные отрывки, смысл которых заранее известен аналитику. Во время Второй мировой войны английские криптоаналитики называли такие отрывки «подсказками» (англ. *crib* — подсказка, шпаргалка). Так, шифрованные тексты различных стран зачастую содержали специфические выражения: приветствия, подписи, различные формы выражения вежливости и т. д.

Другим примером использования этого метода является криптографическая атака на алгоритм простого гаммирования. Если известен хотя бы один открытый текст и соответствующий ему шифротекст длины больше или равной длине гаммы (ключа), то последняя однозначно находится.

Согласно принципу Керкгоффса (подробнее см. подразд. 6.4), криптоаналитик обладает всей информацией о криптосистеме, кроме некоторого набора параметров, называемого ключом. Задачей криптоаналитика является нахождение общего ключа шифрования или алгоритма дешифровки с целью дешифрования других шифротекстов с аналогичным ключом.

В общем виде можно сформулировать принцип атаки:

- P_1, P_2, ..., P_i — открытые тексты;
- $C_1 = E_k(P_1)$, $C_2 = E_k(P_2)$, ..., $C_i = E_k(P_i)$ — соответствующие им шифртексты.

Нужно найти:

- k — ключ шифрования;
- $D(C)$ — функцию дешифрования (не как функцию от ключа, но только от шифротекста).

Разновидностями атаки на основе знания открытого текста являются: атака на основе подобранного открытого текста и атака на основе адаптивно подобранного открытого текста.

Атака на основе подобранного открытого текста является улучшением метода, основанного на открытых текстах. Здесь криптоаналитик также имеет некоторое количество пар «открытый текст — шифрованный текст», известных заранее. Отличие данного метода от атаки на основе открытого текста в том, что в данном методе криптоаналитик может сам выбрать, какой текст он хочет зашифровать, а в методе, основанном только на открытом тексте, все пары «открытый текст — шифрованный текст» известны заранее.

Атака на основе адаптивно подобранного открытого текста является расширением атаки на основе подобранного открытого

текста. Атака предполагает, что криптоаналитик может несколько раз выбирать открытый текст и получать соответствующий ему шифротекст прямо во время атаки. Цель криптоаналитика — получить возможность извлекать информацию из перехваченных шифрованных текстов этой системы, а в идеале — подобрать ключ, сопоставляя открытый текст и полученный шифротекст. Для данного метода необходим непосредственный доступ к шифрующему устройству. Например, к смарт-карте, работающей по некоторому алгоритму шифрования по ключу, недоступному для считывания пользователем.

Также данная атака широко используется для попыток взлома криптографических систем с открытым ключом. Так как в такой криптосистеме услуги шифрования доступны каждому человеку, то любой вариант атаки, в которой не используется блок расшифровки, можно назвать атакой на основе адаптивно подобранного открытого текста.

6.3. КРИПТОГРАФИЧЕСКАЯ СТОЙКОСТЬ ШИФРОВ

Криптографическая стойкость шифра — это способность шифра противостоять возможным атакам на него. В криптографии различают три типа стойкости:

- *вычислительная стойкость* — когда имеется потенциальная возможность вскрыть шифр, но при выбранных параметрах шифрования и ключах на современном этапе развития криптоанализа у злоумышленника не хватит вычислительных ресурсов и времени для вскрытия. Если алгоритм вскрытия шифра на современных мощных компьютерах должен выполнить $\approx 2^{80}$ операций, то шифр называют вычислительно стойким;
- *информационно-теоретическая стойкость* (или абсолютная стойкость) — когда криптоаналитик не может раскрыть криптосистему ни теоретически, ни практически, даже имея бесконечно большие вычислительные ресурсы;
- *доказуемая стойкость*, при которой доказательство стойкости криптосистем сводят к решению определенной трудно решаемой математической проблемы, положенной в основу алгоритма.
- Вопрос о теоретической стойкости шифров впервые был исследован К. Шенноном в его фундаментальной публикации 1949 г., в которой, используя вероятностную модель шифра, он впервые сформулировал понятие *совершенно стойкого шифра* и показал, что он существует на примере шифра Вернама, называемого также одноразовым блокнотом.

6.4. АБСОЛЮТНО СТОЙКИЕ КРИПТОСИСТЕМЫ. ПРИНЦИП КЕРКГОФФСА

Абсолютно стойкие (или теоретически стойкие) ***криптографические системы*** — это криптосистемы, которые не могут быть взломаны ни теоретически, ни практически даже при наличии у атакующего бесконечно больших вычислительных ресурсов.

Стойкость этих систем не зависит от того, какими вычислительными возможностями обладает криптоаналитик.

Впервые требования к криптографическим системам, в основе работы которых лежат шифры, были сформулированы голландским ученым Жаном Вильгельмом Губертом Виктором Франсуа Александром Огюстом Керкгоффсом фон Ньювенгофом в его книге «Военная криптография», опубликованной в 1883 г.

Второе из этих требований получило название ***принцип Керкгоффса***. Это требование заключается в том, что стойкость шифра (а следовательно, и криптосистемы) определяется только секретностью ключа. Данное требование Керкгоффса легло в основу современных требований к шифрам и криптографическим системам.

Шенноном было доказано, что примером абсолютно стойкого алгоритма является шифр Вернама (одноразовый блокнот). Иными словами, корректное использование шифра Вернама не дает злоумышленнику никакой информации об открытом тексте (любой бит сообщения он может лишь угадать с вероятностью 1/2). Им же были определены требования к такого рода системам:

- ключ генерируется для каждого сообщения (каждый ключ используется один раз);
- ключ статистически надежен (т. е. вероятности появления каждого из возможных символов равны, символы в ключевой последовательности независимы и случайны);
- длина ключа равна или больше длины сообщения;
- исходный (открытый) текст обладает некоторой избыточностью (является критерием оценки правильности расшифровки).

В криптографических системах гражданского назначения в основном применяются практически стойкие или вычислительно стойкие системы. О вычислительной стойкости системы говорят в случае, если потенциальная возможность расшифровать суще-

ствует при определенных параметрах и ключах. Стойкость зависит от вычислительной мощности, имеющейся у криптоаналитика.

Стойкость таких систем базируется на теории сложности и оценивается в расчете на определенный момент времени и последовательно с двух позиций:

- какова вычислительная сложность полного перебора;
- каковы известные на данный момент слабости (уязвимости) и их влияние на вычислительную сложность.

В каждом конкретном случае могут существовать дополнительные критерии оценки стойкости.

О доказуемой стойкости говорят в случае, если доказательство стойкости криптосистемы сводится к решению определенной трудно решаемой математической проблемы, положенной в основу алгоритма.

6.5. ПЕРСПЕКТИВНЫЕ НАПРАВЛЕНИЯ КРИПТОАНАЛИЗА. КВАНТОВЫЙ КРИПТОАНАЛИЗ

С развитием вычислительной техники криптоанализ все больше приобретает новые черты. Так, при реализации метода полного перебора злоумышленники используют вычислительные сети и возможности распараллеливания вычислений в них, что существенно повышает скорость и объемы вычислений. Известно два направления в организации параллельного вычисления ключа. Первое направление — построение конвейера. Пусть алгоритм соотношения $E_k(x) = y$ представим в виде детерминированной цепочки простейших действий (операций): $O_1, O_2, \ldots, O_N$.

Возьмем N процессоров $A_1, A_2, \ldots, A_N$, зададим их порядок и положим, что i-й процессор выполняет три одинаковые по времени операции:

1) прием данных от $(i-1)$-го процессора;
2) выполнение операции O_i;
3) передачу данных следующему $(i+1)$-му процессору.

Тогда конвейер из N последовательно соединенных, параллельно и синхронно работающих процессоров работает со скоростью $v/3$, где v — скорость выполнения одной операции процессором.

Второе направление распараллеливания состоит в том, что множество возможных ключей K разбивается на непересекающиеся подмножества $K_1, K_2, \ldots, K_Q$. Система из Q машин перебирает ключи так, что i-я машина осуществляет перебор ключей из множества K_i, $i = \overline{1,Q}$. Система прекращает работу, если одна из машин нашла ключ. Самой большой сложностью в изложенном подходе является организация деления ключевого множества. Однако если организовать поиск ключа таким образом, что при каждом очередном опробовании каждый из N процессоров стартует со случайной точки, то время опробования увеличится, но схема значительно упростится. Среднее число шагов опробования N процессорами (машинами) ключей из множества K в этом случае составляет $|K|/N$. Реализация такого параллелизма предполагает различные решения. Самое очевидное решение — создание компьютерного вируса для распространения программы взломщика в глобальной сети. Вирус должен использовать периоды простоя компьютера (по данным исследований, компьютер простаивает $70-90\,\%$ времени) для осуществления перебора по множеству ключей. Рано или поздно один из зараженных компьютеров обнаружит искомый ключ (необходимо предусмотреть механизм оповещения злоумышленника). С ростом производительности компьютеров и скорости распространения вирусов угроза успешного исхода такой атаки растет.

В настоящее время появились новые направления криптоанализа, например, криптоанализ по побочным каналам. *Атаки по сторонним,* или *побочным, каналам* — это вид криптографических атак, использующих информацию, полученную по сторонним, или побочным, каналам. Под *информацией из побочных каналов* понимается информация, которая может быть получена с устройства шифрования и не является при этом ни открытым текстом, ни шифротекстом.

Например, злоумышленник может отслеживать энергию, потребляемую смарт-картой, когда она выполняет операции с закрытым ключом, такие как расшифрование или генерация подписи. Злоумышленник может также замерять время, затрачиваемое на выполнение криптографической операции, или анализировать поведение криптографического устройства при возникновении определенных ошибок. Побочную информацию на практике собрать порой несложно, поэтому нужно обязательно учитывать такую угрозу при оценке защищенности системы. Атаки по побочным каналам классифицируются по следующим трем типам:

- контролю над вычислительным процессом: *пассивные и активные*;
- способу доступа к модулю: *агрессивные* (invasive), *полуагрессивные* (semiinvasive) и *неагрессивные* (non-invasive);

- методу, применяемому в процессе анализа: *простые* (simple side channel attack, SSCA) и *разностные* (differential side channel attack, DSCA).

На сегодняшний день выделено более десяти побочных каналов. Атаки различаются по виду используемого побочного канала: времени исполнения (Timing Attacks), энергопотреблению (Power Analysis Attacks), ошибкам вычислений (Fault Attacks), электромагнитному излучению (ElectroMagnetic Analysis), ошибкам в канале связи (Error Message Attacks). Существуют и более изощренные виды атак: по кеш-памяти (Cache-based Attacks), акустические (Acoustic Attacks), по световому излучению (Visible Light Attacks).

Атака по времени — способ получения какой-либо скрытой информации путем точного измерения времени, которое требуется пользователю для выполнения криптографических операций.

Атака по анализу мощности пригодна в основном для аппаратной реализации криптографических средств и успешно применяется при взломе смарт-карт и других систем, в которых хранится секретный ключ.

Атака по мощности может быть разделена на простую (Simple Power Analysis, SPA) и разностную (Differential Power Analysis, DPA). Целью SPA является получение информации о конкретных выполняемых инструкциях в системе и о конкретных обрабатываемых данных. В общем случае SPA может дать как сведения о работе устройства, так и информацию о ключе. Для осуществления этой атаки криптоаналитик должен располагать точными данными о реализации устройства. Этот метод использует непосредственные данные измерений, собранные во время выполнения криптографических операций. Простая атака по мощности для смарт-карт обычно занимает несколько секунд, в то время как разностная атака по мощности может занять несколько часов.

В отличие от простых атак, разностные атаки, основанные на анализе потребляемой мощности, подразумевают не только визуальное представление потребляемой мощности, но также статистический анализ и статистические методы исправления ошибок для получения информации о ключах. Более того, DPA зачастую не нуждается в данных о конкретной реализации и в качестве альтернативы использует статистические методы анализа. Разностный анализ мощности — одно из самых мощных средств для проведения атак, использующих побочные каналы, причем эта атака требует очень маленьких затрат.

Ошибки аппаратного обеспечения, появляющиеся во время работы соответствующего криптографического модуля, или ошибочные выходные данные могут стать важными побочными каналами и иногда существенно увеличивают уязвимость шифра к криптоанализу. Криптоанализ на основе формирования случайных аппаратных ошибок — это вид нападения на шифры в случае, когда предполагаемый нарушитель имеет возможность оказать на шифратор внешнее физическое воздействие и вызвать одиночные ошибки в процессе шифрования одного блока данных.

Выполнение вычислительных операций на компьютере сопряжено с выделением электромагнитного излучения. Измеряя и анализируя это излучение, злоумышленник может получить значительную информацию о выполняющихся вычислениях и используемых данных. Атаки по электромагнитному анализу могут быть также разделены на две большие категории: простые (SEMA) и дифференциальные (DEMA).

Широкое распространение и развитие квантовой криптографии не могло не вызвать появление квантового криптоанализа, который в ряде случаев обладает, согласно теории, преимуществами перед обычным. С помощью квантового криптоанализа возможен «взлом» симметричных шифров с помощью алгоритма Гровера (аналог полного перебора вариантов). Из-за квадратичного выигрыша, который дает данный алгоритм относительно перебора на классическом компьютере, симметричные шифры для сохранения практической стойкости должны использовать ключи в два раза большей разрядности.

Квантовый компьютер можно использовать для решения задач разложения очень больших чисел на простые множители и задач дискретного логарифмирования. Для этого был разработан квантовый алгоритм Шора, позволяющий найти за конечное и приемлемое время все простые множители больших чисел или решить задачу дискретного логарифмирования, и, как следствие, взломать шифры, стойкость которых основана на данных проблемах.

В связи с этим многие криптографы в настоящее время ведут разработку алгоритмов, независимых от квантовых вычислений, т. е. устойчивых к подобным квантовым атакам.

ВОПРОСЫ И ЗАДАНИЯ ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ ЗНАНИЙ

1. Дайте определение понятия «криптоанализ».
2. Какова задача криптоанализа?
3. Перечислите основные методы криптоанализа.

4. Охарактеризуйте частотный криптоанализ.
5. Дайте определение атаки на основе знания открытого текста.
6. Что такое криптографическая стойкость шифров?
7. Перечислите виды криптографической стойкости шифров.
8. Какие криптографические системы являются абсолютно стойкими?
9. Перечислите перспективные направления криптоанализа.
10. Дайте определение атаки по времени.
11. Перечислите атаки на основе побочных каналов.
12. Какие бывают атаки по мощности?
13. Как использует криптоанализ аппаратные ошибки?
14. Как можно использовать квантовый криптоанализ?
15. Какие возможности дает квантовый криптоанализ при дешифрации?

Раздел III

ПРИМЕНЕНИЕ КРИПТОГРАФИИ

Глава 7

ЭЛЕКТРОННАЯ ПОДПИСЬ

7.1. ОДНОНАПРАВЛЕННЫЕ ХЕШ-ФУНКЦИИ

Хеш-функция (англ. *hash* — мелко измельчать и перемешивать) предназначена для сжатия произвольного сообщения или набора данных, записанного, как правило, в двоичном алфавите, в некоторую битовую комбинацию фиксированной длины, называемую сверткой.

Хеш-функция h принимает в качестве аргумента сообщение (документ) M произвольной длины и возвращает хеш-значение $h(M) = H$ фиксированной длины. Обычно хешированная информация является сжатым двоичным представлением основного сообщения произвольной длины. Следует отметить, что значение хеш-функции $h(M)$ сложным образом зависит от документа M и не позволяет восстановить сам документ M.

Среди хеш-функций выделяют *однонаправленные хеш-функции*. Большинство однонаправленных хеш-функций строится на основе однонаправленной функции $F(x)$, которая образует выходное значение длиной n при задании двух входных значений длиной n. Этими входами являются блок исходного текста M и хеш-значение H_{i-1} предыдущего блока текста. В начальный момент H_{i-1} принимается равной некоторому фиксированному начальному значению.

Функция $F: X \to Y$ называется ***однонаправленной***, если выполняются следующие два условия:

1) существует эффективный алгоритм, вычисляющий $F(x)$ для любого $x \in X$;

2) не существует эффективного алгоритма инвертирования функции $F(x)$, т. е. алгоритма, позволяющего определить значение x по значению $F(x)$.

Под эффективным алгоритмом будем понимать полиномиальный алгоритм, т. е. алгоритм, который для получения результата для текста длины n тратит не более $P(n)$ шагов, где P — некоторый полином.

В настоящее время теория алгоритмов не позволяет доказать отсутствие эффективных алгоритмов решения той или иной задачи, поэтому, строго говоря, не известна ни одна однонаправленная функция. Однако предложено несколько функций, которые могут оказаться однонаправленными: для этих функций на сегодняшний день, несмотря на интенсивные исследования, не известны эффективные алгоритмы инвертирования.

Наиболее часто используются следующие однонаправленные функции:

- функция $F(a, b) = ab$, т. е. произведение двух чисел. Если a и b — простые числа, то по известному $c = ab$ можно однозначно определить a и b. Эта задача называется *задачей факторизации числа*. До сих пор не известен ни один полиномиальный алгоритм для решения задачи факторизации, хотя для вычисления произведения чисел (т. е. самой функции F) такие алгоритмы известны;
- функция $F(a, n) = a^n \pmod M$, где a, n, M — целые числа. При известных a, n, M значение $b = a^n \pmod M$ может быть вычислено за полиномиальное количество шагов. Однако для обратной задачи — определить n по известным a, b, M (*задача дискретного логарифмирования*) — полиномиальные алгоритмы, в общем случае, пока не известны.

В криптографии можно использовать не любую однонаправленную функцию. Если преобразовать открытый текст M с помощью однонаправленной функции $C = F(M)$, то расшифровать полученный текст, т. е. по C восстановить M, не сможет никто, в том числе и законный получатель.

Для использования в криптографии необходимо, чтобы задача инвертирования шифрующего преобразования (т. е. вычисления M по $F(M)$) была разрешима за приемлемое время, но сделать это мог только тот, кто знает секретный ключ. Такие функции называются ***однонаправленными функциями с секретом***.

Будем считать однонаправленной функцией с секретом функцию F_k: $X \to Y$, зависящую от параметра $k \in K$ (этот параметр называется секретом), для которой выполняются следующие условия:

1) при любом $k \in K$ существует эффективный алгоритм, вычисляющий $F_k(x)$ для любого $x \in X$;

2) при неизвестном k не существует эффективного алгоритма инвертирования функции $F_k(x)$;

3) при известном k существует эффективный алгоритм инвертирования функции $F_k(x)$.

Существование однонаправленных функций с секретом, как и однонаправленных функций, пока не доказано. Однако известны функции, которые могут оказаться однонаправленными функциями с секретом, если будет доказано, что инвертирование этих функций действительно является сложной задачей.

Однонаправленную хеш-функцию можно построить, используя симметричный блочный алгоритм. Наиболее очевидный подход состоит в том, чтобы шифровать сообщение M посредством блочного алгоритма в специальных режимах CBC (сцепление блоков шифра) или CFB (обратная связь по шифротексту) с помощью фиксированного ключа и некоторого вектора инициализации. Последний блок шифротекста можно рассматривать в качестве хеш-значения сообщения M. При таком подходе не всегда возможно построить безопасную однонаправленную хеш-функцию, но всегда можно получить код аутентификации сообщения MAC (Message Authentication Code).

Более безопасный вариант хеш-функции можно получить, используя блок сообщения в качестве ключа, предыдущее хеш-значение — в качестве входа, а текущее хеш-значение — в качестве выхода. Реальные хеш-функции проектируются еще более сложными. Длина блока обычно определяется длиной ключа, а длина хеш-значения совпадает с длиной блока.

Поскольку большинство блочных алгоритмов являются 64-битовыми, некоторые схемы хеширования проектируют так, чтобы хеш-значение имело длину, равную двойной длине блока.

Если принять, что получаемая хеш-функция корректна, то безопасность схемы хеширования базируется на безопасности лежащего в ее основе блочного алгоритма.

7.2. КРИПТОГРАФИЧЕСКИЕ ХЕШ-ФУНКЦИИ

Хеш-функция MD5. Алгоритм MD5 разработан Роном Ривестом. Алгоритм получает на входе сообщение произвольной длины и создает в качестве выхода дайджест сообщения длиной 128 бит. Алгоритм состоит из нескольких шагов шагов.

Шаг 1. Добавление недостающих битов: сообщение дополняется таким образом, чтобы его длина стала равна 448 по модулю 512 (длина ≡ 448 (mod 512)). Это означает, что длина добавленного сообщения на 64 бита меньше, чем число, кратное 512. До-

бавление производится всегда, даже если сообщение имеет нужную длину. Например, если длина сообщения 448 битов, то оно дополняется 512 битами до 960 битов. Таким образом, число добавляемых битов находится в диапазоне от 1 до 512. Добавление состоит из единицы, за которой следует необходимое количество нулей.

Шаг 2. Добавление длины: 64-битное представление длины исходного (до добавления) сообщения в битах присоединяется к результату первого шага. Если первоначальная длина больше, чем 2^{64}, то используются только последние 64 бита. Таким образом, поле содержит длину исходного сообщения по модулю 2^{64}.

В результате первых двух шагов создается сообщение, длина которого кратна 512 битам. Это расширенное сообщение представляется как последовательность 512-битных блоков Y_0, Y_1, ..., Y_{L-1}, при этом общая длина расширенного сообщения равна L = 512 битам. Таким образом, длина полученного расширенного сообщения кратна 16 32-битным словам.

Шаг 3. Инициализация MD-буфера: используется 128-битный буфер для хранения промежуточных и окончательных результатов *хеш-функции*. Буфер может быть представлен как четыре 32-битных регистра (*A*, *B*, *C*, *D*). Эти регистры инициализируются следующими шестнадцатеричными числами: *A* = 01234567; B = = 89ABCDEF; *C* = FEDCBA98; *D* = 76543210.

Шаг 4. Обработка последовательности 512-битных (16-словных) блоков: основой алгоритма является модуль, состоящий из четырех циклических обработок, обозначенный как HMD5 (рис. 7.1; [17, с. 174]) . Четыре цикла имеют похожую структуру, но каждый цикл использует свою элементарную логическую функцию, обозначаемую f_F, f_G, f_H и f_I соответственно.

Каждый цикл принимает в качестве входа текущий 512-битный блок Y_q, обрабатывающийся в данный момент, и 128-битное значение буфера *ABCD*, которое является промежуточным значением дайджеста, и изменяет содержимое этого буфера. Каждый цикл также использует четвертую часть 64-элементной таблицы T [1 ... 64], построенной на основе функции sin: *i*-й элемент T, обозначаемый *T*[*i*], имеет значение, равное целой части от 2^{32} abs (sin (*i*)), *i* задано в радианах. Так как abs (sin (*i*)) является числом между 0 и 1, каждый элемент *T* является целым, которое может быть представлено 32 битами. Таблица обеспечивает «случайный» набор 32-битных значений, которые должны ликвидировать любую регулярность во входных данных.

++++++++++++++++++++++++++

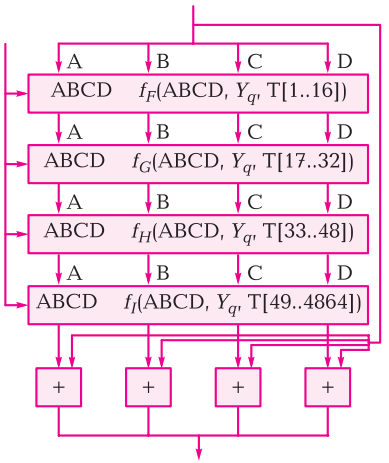

Рис. 7.1. Обработка очередного 512-битного блока

Для получения MD_{q+1} выход четырех циклов складывается по модулю 2^{32} с MD_q. Сложение выполняется независимо для каждого из четырех слов в буфере.

Шаг 5. Выход: после обработки всех *L* 512-битных блоков выходом *L*-й стадии является 128-битный дайджест сообщения.

Рассмотрим более детально логику каждого из четырех циклов выполнения одного 512-битного блока. Каждый цикл состоит из 16 шагов, оперирующих с буфером *ABCD*. Каждый шаг можно представить в виде, представленном на рис. 7.2 [17, с. 176]:

$$A \leftarrow B + CLS_s\ (A + f\ (B,\ C,\ D) + X\ [k] + T\ [i]),$$

где *A, B, C, D* — четыре слова буфера; после выполнения каждого отдельного шага происходит циклический сдвиг влево на одно слово; *f* — одна из элементарных функций f_F, f_G, f_H, f_I; CLS_s — циклический сдвиг влево на *s* битов 32-битного аргумента, *X* [*k*] – *M* [*q*·16 + *k*] — *k*-е 32-битное слово в *q*-ом 512 блоке сообщения, *T* [*i*] — *i*-е 32-битное слово в матрице T, «+» — сложение по модулю 2^{32}.

На каждом из четырех циклов алгоритма используется одна из четырех элементарных логических функций. Каждая элементарная функция получает три 32-битных слова на входе и на выходе создает одно 32-битное слово. Каждая функция является множеством побитовых логических операций, т. е. *n*-й бит выхода является функцией от *n*-го бита трех входов. Элементарные функции следующие:

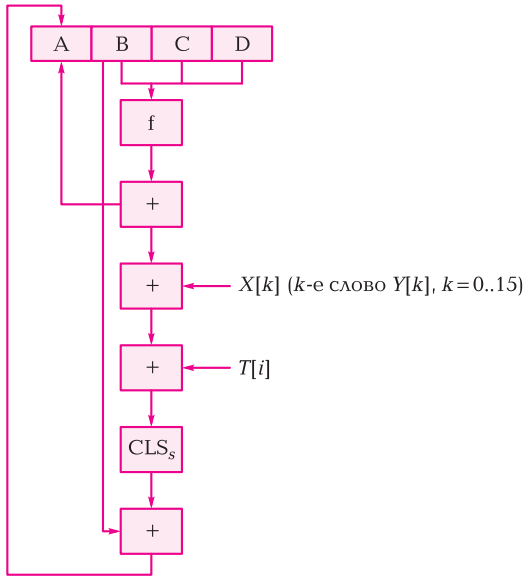

Рис. 7.2. Логика выполнения отдельного шага

$$f_F = (B \,\&\, C)(\text{not } B \,\&\, D),$$

$$f_G = (B \,\&\, D)(\text{C} \,\&\, \text{not } D),$$

$$f_H = B \oplus C \oplus D,$$

$$f_I = C \oplus (B \,\&\, \text{not } D).$$

Массив из 32-битных слов X [0...15] содержит значение текущего 512-битного входного блока, который обрабатывается в настоящий момент. Каждый цикл выполняется 16 раз. Если представить входной 512-битный блок в виде шестнадцати 32-битных слов, то каждое входное 32-битное слово используется четыре раза, по одному разу в каждом цикле, и каждый элемент таблицы T, состоящей из 64 32-битных слов, используется только один раз. После каждого шага цикла происходит циклический сдвиг влево четырех слов *A, B, C* и *D*. на каждом шаге изменяется только одно из четырех слов буфера *ABCD*. Следовательно, каждое слово буфера изменяется 16 раз и затем 17-й раз в конце — для получения окончательного выхода данного блока.

Хеш-функция SHA-1. Безопасный хеш-алгоритм SHA (Secure Hash Algorithm) был разработан Национальным институтом стандартов и технологии (NIST) США и опубликован в качестве федерального информационного стандарта (FIPS PUB 180) в 1993 г. SHA-1, как и MD5, основан на алгоритме MD4.

Алгоритм получает на входе сообщение максимальной длины 264 бит и создает в качестве выхода дайджест сообщения длиной 160 бит. Алгоритм состоит из ряда шагов.

Шаг 1. Добавление недостающих битов. К сообщению добавляются дополнительные биты таким образом, чтобы его длина была кратна 448 по модулю 512 (длина 448 mod 512). Добавление осуществляется всегда, даже если сообщение уже имеет нужную длину. Таким образом, число добавляемых битов находится в диапазоне от 1 до 512. Добавление состоит из единицы, за которой следует необходимое количество нулей.

Шаг 2. Добавление длины. К сообщению добавляется блок из 64 битов. Этот блок трактуется как беззнаковое 64-битное целое и содержит длину исходного сообщения до добавления.

Результатом первых двух шагов является сообщение, длина которого кратна 512 битам. Расширенное сообщение может быть представлено как последовательность 512-битных блоков Y_0, Y_1, ..., Y_{L-1}, так что общая длина расширенного сообщения есть $L \cdot 512$ бит. Таким образом, результат кратен шестнадцати 32-битным словам.

Шаг 3. Инициализация SHA-1 буфера. Используется 160-битный буфер для хранения промежуточных и окончательных результатов хеш-функции. Буфер может быть представлен как пять 32-битных регистров *A, B, C, D* и *E*. Эти регистры инициализируются следующими шестнадцатеричными числами:

A = 67452301; B = EFCDAB89; C = 98BADCFE; D = 10325476; E = C3D2E1F0

Шаг 4. Обработка сообщения в 512-битных блоках. Основой алгоритма является модуль, состоящий из 80 циклических обработок, обозначенный как HSHA. Все 80 циклических обработок имеют одинаковую структуру. Каждый цикл получает на входе текущий 512-битный обрабатываемый блок Y_q и 160-битное значение буфера *ABCDE* и изменяет содержимое этого буфера. В каждом цикле используется дополнительная константа K_t, которая принимает только четыре различных значения:

$0t19 K_t$ = 5A827999 (целая часть числа $[230 \times 21/2]$)

$20t39$ K_t = 6ED9EBA1 (целая часть числа $[230 \times 31/2]$)

$40t59$ K_t = 8F1BBCDC (целая часть числа $[230 \times 51/2]$)

$60t79$ K_t = CA62C1D6 (целая часть числа $[230 \times 101/2]$)

Для получения SHA_{q+1} выход 80-го цикла складывается со значением SHA_q. Сложение по модулю 2^{32} выполняется независимо

для каждого из пяти слов в буфере с каждым из соответствующих слов в SHA_q.

Шаг 5. Выход*:* после обработки всех 512-битных блоков выходом *L*-й стадии является 160-битный дайджест сообщения.

Сравним оба алгоритма в соответствии с теми целями, которые были определены для каждого алгоритма.

Безопасность*:* наиболее очевидное и наиболее важное различие состоит в том, что дайджест SHA-1 на 32 бита длиннее, чем дайджест MD5. Если предположить, что оба алгоритма не содержат каких-либо структурированных данных, которые уязвимы для криптоаналитических атак, то SHA-1 является более стойким алгоритмом. Используя лобовую атаку, труднее создать произвольное сообщение, имеющее данный дайджест, если требуется порядка 2^{160} операций, как в случае алгоритма SHA-1, чем порядка 2^{128} операций, как в случае алгоритма MD5. Используя лобовую атаку, труднее создать два сообщения, имеющие одинаковый дайджест, если требуется порядка 2^{80}, как в случае алгоритма SHA-1, чем порядка 2^{64} операций, как в случае алгоритма MD5.

Скорость: так как оба алгоритма выполняют сложение по модулю 2^{32}, они рассчитаны на 32-битную архитектуру. SHA-1 содержит больше шагов (80 вместо 64) и выполняется на 160-битном буфере по сравнению со 128-битным буфером MD5. Таким образом, SHA-1 должен выполняться приблизительно на 25 % медленнее, чем MD5 на той же аппаратуре.

Простота и компактность: оба алгоритма просты и в описании, и в реализации, не требуют больших программ или подстановочных таблиц. Тем не менее SHA-1 применяет одношаговую структуру по сравнению с четырьмя структурами, используемыми в MD5. Более того, обработка слов в буфере одинаковая для всех шагов SHA-1, в то время как в MD5 структура слов специфична для каждого шага.

Хеш-функция SHA-512. Хеш-функция SHA-512 относится к семейству хеш-функций SHA-2. SHA-512 построена на основе структуры Меркла — Дамгарда. Непосредственно перед обработкой входного блока данных исходное сообщение дополняется до полного блока и разбивается на блоки. Каждый блок разбивается, в свою очередь, на 16 слов. Алгоритм пропускает каждый блок сообщения через цикл с 80-ю раундами. на каждой итерации преобразуются два слова, при этом функцию преобразования задают остальные слова. Результаты обработки каждого блока складываются. Итоговая сумма является значением хеш-функции.

Алгоритм использует следующие битовые операции:

-- — конкатенация;

+ — сложение;

and — побитовое «И»;

or — побитовое «ИЛИ»;

xor — исключающее «ИЛИ»;

shr (*Shift Right*) — логический сдвиг вправо;

rotr (*Rotate Right*) — циклический сдвиг вправо.

Проведенные исследования по криптостойкости данных хеш-функций каких-либо критических уязвимостей не обнаружили, что позволяет рекомендовать широкое применение SHA-512.

Хеш-функция ГОСТ Р 34.11—94. Российский стандарт ГОСТ Р 34.11—94 определяет алгоритм и процедуру вычисления хеш-функции. В основе данной хеш-функции лежит блочный алгоритм шифрования ГОСТ Р 28147—89. Данная функция формирует 256-битное значение.

Основой описываемой хеш-функции является функция хеширования

$$H_{out} = f\,(H_{in},\, m),$$

где H_{out}, H_{in}, m — блоки длины 256 бит.

Входное сообщение M разделяется на блоки m_n, m_{n-1}, m_{n-2}, ..., m_1 по 256 бит. В случае если размер последнего блока m_n меньше 256 бит, то к нему приписываются слева нули для достижения заданной длины блока.

Каждый блок сообщения, начиная с первого, подается на шаговую функцию для вычисления промежуточного значения хеш-функции:

$$H_{i+1} = f\,(H_i,\, m_i).$$

Значение H_i можно выбрать произвольным. После вычисления H_{i+1} конечное значение хеш-функции получают следующим образом:

- $H_{n+2} = f\,(H_{i+1},\, L)$, где L — длина сообщения M в битах по модулю 2^{256};
- $h = f\,(H_{i+2},\, K)$, где K — контрольная сумма сообщения M: $m_1 + m_2 + m_3 + \ldots + m_n$;
- h — значение хеш-функции сообщения M.

Алгоритм работы хеш-функции включает в себя инициализацию, выполнение функции сжатия внутренних итераций, выполнение функции сжатия финальной итерации, получение окончательного значения хеш-функции.

Особенностями работы алгоритма ГОСТ Р 34.11—94 являются следующие моменты:

- обрабатывается блок длиной 256 бит, и выходное значение тоже имеет длину 256 бит;
- определяется контрольная сумма, рассчитанная по всем блокам исходного сообщения, которая является частью финального вычисления хеша, что несколько затрудняет коллизионную атаку;
- применены меры борьбы против поиска коллизий, в основанном на неполноте последнего блока;
- обработка блоков происходит по алгоритму шифрования ГОСТ 28147—89, который содержит преобразования на *S*-блоках, что существенно осложняет применение метода дифференциального криптоанализа к поиску коллизий.

Хеш-функция ГОСТ Р 34.11—2012. ГОСТ Р 34.11—2012 определяет алгоритм и процедуру вычисления хеш-функции для любой последовательности двоичных символов, которые применяются в криптографических методах обработки и защиты информации, в том числе для реализации процедур обеспечения целостности, аутентичности, электронной подписи (ЭП) при передаче, обработке и хранении информации в автоматизированных системах. Функция хеширования ГОСТ Р 34.11—2012 используется при реализации систем электронной подписи на базе асимметричного криптографического алгоритма по ГОСТ Р 34.10—2012.

В ГОСТ Р 34.11—2012 размер блоков сообщения и внутреннего состояния хеш-функции составляет 512 бит. Новый стандарт определяет две функции хеширования с длинами хеш-кода 256 и 512 бит.

Основное отличие современной хеш-функции от старой — использование функции сжатия. В ГОСТ Р 34.11—2012 используется функции сжатия, в основе которой лежат три преобразования: нелинейное биективное (обозначается *S*), перестановка байт (обозначается *P*), линейное преобразование (обозначается *L*). В ГОСТ Р 34.11—94 используется функция сжатия, основанная на симметричном блочном шифре ГОСТ Р 28147—89, также эта функция использует операции перемешивания.

При вычислении новой хеш-функции, если размер сообщения не кратен размеру обрабатываемого блока (для современного стандарта — 512 бит, для старого стандарта — 256 бит), то такой блок дополняется вектором (00 ... 01). При вычислении старой хеш-функции неполный блок дополняется значением (00 ... 0). Считается, что дополнение (00 ... 01) лучше, чем (00 ... 0), с криптографической точки зрения, так как дополнение значением (00 ... 0) приводит к атакам на алгоритм.

Еще одно отличие состоит в том, что стандарт ГОСТ Р 34.11—94 не определял значение инициализационного вектора, в то время как в стандарте ГОСТ Р 34.11—2012 значение инициализационного вектора фиксировано и определено в стандарте: для хеш-функции с размером выходного хеша 512 бит — это вектор (00 ... 0), для хеш-функции с размером выходного хеш-кода 256 бит — (000000010 ... 100000001) (все байты равны 1).

В основу хеш-функции положена итерационная конструкция Меркла – Дамгора с использованием MD-усиления. Под MD-усилением понимается дополнение неполного блока при вычислении хеш-функции до полного путем добавления вектора (0 ... 01) такой длины, чтобы получился полный блок. Из дополнительных элементов нужно отметить следующие:

- завершающее преобразование, которое заключается в том, что функция сжатия применяется к контрольной сумме всех блоков сообщения по модулю 2^{512};
- при вычислении хеш-кода на каждой итерации применяются разные функции сжатия, т. е. можно сказать, что функция сжатия зависит от номера итерации.

Описанные выше решения позволяют противостоять многим известным атакам.

Кратко описание хеш-функции ГОСТ Р 34.11—2012 можно представить следующим образом. на вход хеш-функции подается сообщение произвольного размера. Далее сообщение разбивается на блоки по 512 бит; если размер сообщения не кратен 512, то оно дополняется необходимым количеством бит. Потом итерационно используется функция сжатия, в результате действия которой обновляется внутреннее состояние хеш-функции. Также вычисляется контрольная сумма блоков и число обработанных бит. Когда обработаны все блоки исходного сообщения, производятся еще два вычисления, которые завершают вычисление хеш-функции:

- обработка функцией сжатия блока с общей длиной сообщения;
- обработка функцией сжатия блока с контрольной суммой.

Анализ безопасности хеш-функций. Основным требованием по безопасности к криптографическим хеш-функциям является их криптографическая стойкость. Для того чтобы хеш-функция H считалась криптографически стойкой, она должна удовлетворять трем основным требованиям, на которых основано большинство применений хеш-функций в криптографии:

- необратимость или стойкость к восстановлению прообраза: для заданного значения хеш-функции m должно быть вычислительно невозможно найти блок данных X, для которого $H(X) = m$;

- стойкость к коллизиям первого рода или восстановлению вторых прообразов: для заданного сообщения M должно быть вычислительно невозможно подобрать другое сообщение N, для которого $H(N) = H(M)$;
- стойкость к коллизиям второго рода: должно быть вычислительно невозможно подобрать пару сообщений (M, M'), имеющих одинаковый хеш.

Данные требования не являются независимыми:

- обратимая функция нестойка к коллизиям первого и второго рода;
- функция, нестойкая к коллизиям первого рода, нестойка к коллизиям второго рода; обратное неверно.

Важную роль при использовании хеш-функций играет выбор длины выходного хеш-значения. Чем меньше длина выходного значения, тем меньше множество значений хеш-функции и тем проще злоумышленнику обнаружить коллизию. Для хеш-функций, однако, не существует такой универсальной «лобовой» атаки, как полный перебор всего ключевого пространства для криптосистем.

Одной из стандартных атак на функцию хеширования является атака, основанная на парадоксе о днях рождения. Атака, основанная на рассмотренном парадоксе, направлена на обнаружение коллизии. В чем она заключается? Пусть есть некоторая хеш-функция, которая может принимать N значений. Если вычислять хеш-значения для сообщений из некоторого потока с помощью данной функции, то появления первой коллизии следует ожидать через N вычислений. Таким образом, если длина хеш-значения составляет 64 бита, то потребуется всего 2^{32} вычислений (или чуть больше) чтобы обнаружить коллизию.

Длина выходного значения существующих хеш-функций составляет от 128 до 512 бит. С учетом того, что нижний предел безопасности в современном мире составляет 80 бит, минимальная длина хеш-значения должна составлять 160 бит, что обеспечит защиту от атак, направленных на обнаружение коллизий.

В последние годы были затрачены значительные усилия и достигнуты определенные успехи в деле разработки криптографических методов анализа функции хеширования. Чтобы понять их, рассмотрим структуру типичной защищенной функции хеширования, показанную на рис. 7.3. Эту структуру, называемую ***итерированной функцией хеширования***, предложил Меркл (Merkle), и именно такую структуру имеет большинство используемых сегодня функций хеширования, включая MD5, SHA-1 и RIPEMD-160.

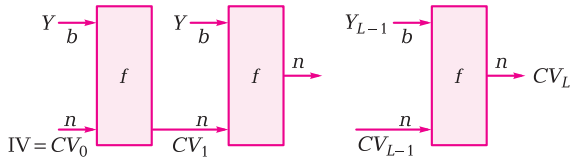

Рис. 7.3. Общая структура защищенного хеш-кода

Функция хеширования получает на вход сообщение и делит его на L-1 блоков равной фиксированной длины по b битов каждый. Если необходимо, последний блок дополняется до b битов. В последний блок также включается значение суммарной длины ввода функции хеширования. Это делает задачу злоумышленника еще более сложной. Злоумышленник должен найти либо два сообщения равной длины, имеющие одинаковые значения функции хеширования, либо два сообщения разной длины, которые вместе с соответствующими им значениями длины будут иметь одинаковые значения функции хеширования.

Алгоритм хеширования предполагает многократное применение функции сжатия f, получающей на вход два значения: CV_i — n-битовое значение, полученное на предыдущем этапе и называемое переменной сцепления; Y— b-битовый блок сообщения и порождающее n-битовое выходное значение.

В начале хеширования переменная сцепления получает начальное значение, являющееся частью алгоритма. Конечное значение переменной сцепления и будет значением функции хеширования. Обычно $b > n$, поэтому и говорят о сжатии.

Функция хеширования может быть формально описана следующим образом:

$$CV_0 = \text{IV} = \text{начальное } n\text{-битовое значение}$$

$$CV_1 = f\,(CV_{i-1},\ Y_{i-1}),\ 1 \le i \le L,$$

где вводимыми данными функции хеширования является сообщение M, складывающееся из блоков Y_i.

Толчком к созданию такого рода итерационных структур послужили результаты работ Меркла и Дамгарда (Damgard), свидетельствующие о том, что если функция сжатия обладает сопротивляемостью коллизиям, то такой же будет и итерированная функция хеширования. Поэтому такая структура может использоваться для создания защищенной функции хеширования, работающей с сообщениями любой длины. Проблема создания защищенной функ-

ции хеширования сводится к проблеме поиска функции сжатия, обладающей сопротивляемостью коллизиям и работающей с вводимыми данными некоторой фиксированной длины.

Криптоанализ функций хеширования обычно сосредоточен на исследовании внутренней структуры f и опирается на попытки найти эффективные методы обнаружения коллизий при однократном выполнении f.

Если эта проблема решена, то атакующему остается рассмотреть фиксированное значение IV. Конкретный вид атаки на f зависит от внутренней структуры этой функции. Обычно, например, когда речь идет о симметричных блочных шифрах, f предполагает несколько раундов обработки данных, так что лучше всего выполнять анализ изменения побитовой структуры данных от раунда к раунду.

Атаки на хеш-функции. *Грубый взлом хеш-функций (метод простого перебора)* основан на обычном переборе всех комбинаций определенного набора символов. Предположим, злоумышленнику известен алгоритм построения хеш-функции h, первоначальное сообщение $M \in \{0, 1\}$ и хеш-значение $h(M)$. Требуется найти $N \in \{0,1\}$ такое, что $h(M) = h(N)$. Для произвольных $M, N \in \{0, 1\}$ вероятность того, что $h(M) = h(N)$, равна 2^{-n}, где n — порядок хеш-функции h. Обозначим, через $P_1(k, n)$ вероятность того, что для фиксированного $M \in \{0, 1\}$ и $N_1, ..., N_k \in \{0, 1\}$ существует номер $i = 1, ..., k$: $h(M) = h(N_i)$. Тогда получим:

$$P_1(k, n) = 1 - (1 - 2^{-n})^k \approx k\ 2^{-n};$$

значения вероятности в зависимости от k представлены в табл. 7.1.

Как следует из табл. 7.1, злоумышленнику для подделки хеш-значения порядка n фиксированного текста методом подбора потребуется перебрать не менее 2^{n-7} текстов. При этом вероятность успеха будет не выше 1 %. Данное свойство иллюстрирует, что для взлома фиксированного значения перебор не годится. Действительно, уже для 128 битных хеш-значений (MD2, MD4, MD5, RIPEMD128, HAVAL3—4) потребуется перебрать около 2^{120} текстов. Осуществить это за обозримое время невозможно.

Взлом методом поиска по таблице — чрезвычайно эффективный метод для взлома множества хешей одного типа. Используется при подборе пароля пользователя. Суть метода заключается в построении таблицы с заранее вычисленными значениями хешей и соответствующими значениями пароля. При этом просто производится поиск хеша подвергаемого взлому и затем из таблицы берется исходное значение пароля.

Таблица 7.1. Вероятность коллизий первого рода $P_1(k, n)$ в зависимости от длины перебора текстов k

$P_1(k, n)$	k
0,01	$\approx 2^{n-7}$
0,5	$\approx 2^{n-1}$
0,99	$\approx 2^n$

Похожий метод взлома называется *обратным поиском по таблице*. Это очень эффективный метод при больших объемах пользовательской базы данных (БД). Эта атака позволяет злоумышленнику применять метод перебора паролей ко множеству хешей одновременно. Сначала создается поисковая таблица, где в соответствие каждому возможному значению хеша ставится один или более пользователей, имеющих данный хеш. Затем злоумышленник перебирает все возможные варианты пароля, хеширует их и делает поиск пользователей с таким же значением хеша.

Последний, наиболее изощренный и прогрессивный, способ взлома — это *использование* так называемых *радужных таблиц*.

Радужная таблица строится путем вычисления хеш-значения наиболее часто используемых слов и словосочетаний. Такие таблицы могут содержать миллионы, а то и миллиарды строк. К примеру, для создания такой таблицы можно пройтись по словарю и создать хеш для каждого слова. Также можно создавать хеши для комбинации слов. Но и это не все; можно тоже вставлять цифры перед (после, между) слов и записывать значение таких хешей в таблицу. Вот такие огромные радужные таблицы могут быть составлены и использованы. Далее путем поиска в такой таблице подбирают слова, входящие, например, в пароль. Чтобы затруднить атаки описанного вида, применяется так называемая соль. Это стандартное средство, но в условиях современных вычислительных мощностей его уже недостаточно, особенно если длина соли невелика. В общем виде функцию с использованием соли можно представить так:

$$f\ (password, salt) = hash\ (password + salt).$$

Для затруднения атаки соль должна быть длиной не менее 64 символов. Но проблема в том, что для дальнейшей аутентификации пользователей соль должна храниться в БД в виде простого текста.

$$if\ (hash([\text{введенный пароль}] + [\text{соль}]) == [\text{хеш}]),$$

тогда пользователь аутентифицирован.

Благодаря уникальности соли для каждого пользователя можно решить проблему коллизий простых хешей. Теперь все хеши будут разными. Также уже не сработают подходы с поиском хешей и подбором паролей. Но если злоумышленник через внедрение SQL-кода получит доступ к соли или БД, в которой хранятся хеши паролей, то сможет успешно атаковать подбором паролей или перебором по словарю, особенно если пользователи выбирают распространенные пароли (типа 123456).

Тем не менее взлом любого из паролей уже не позволит автоматически вычислить пользователей, у которых тот же пароль: ведь теперь все хеши разные.

Взлом хеш-функций на основе парадокса дней рождений (другое название «Атака дня рождения») направлен на поиск коллизий хеш-функций. Для функции хеширования с n-битовыми выходными данными возникновения коллизий следует ожидать через $2^{n/2}$ шагов.

Взлом хеш-функций за линейное время (*туннельный эффект*) основывается на специфике построения большинства хеш-функций. Действительно, большинство хеш-функций основано на функции сжатия (или сдвиговой функции). Текст M разбивается на блоки M_i (в большинстве случаев размером в 512 бит) и включается итерационный процесс подсчета хеш-значения $h_i = f\ (h_{i-1}, M_i)$. Так как длина раунда небольшая, то, применяя дифференциальный анализ, возможно найти такой текст ΔC: $h(M_i + \Delta C) = h(M_i)$.

П р и м е р. Рассмотрим широко распространенную однонаправленную хеш-функцию MD4, предложенную Роном Ривестом. MD4 обладает 128-битным выходным значением и является самой быстрой хеш-функцией.

Пусть $\Delta C = (0, 2^{31}, 2^{31} - 2^{28}, 0, 0, 0, 0, 0, 0, 0, 0, 0, -2^{16}, 0, 0, 0)$. Тогда для любого текста M: $\mathrm{MD4}(M + \Delta C) = \mathrm{MD4}(M)$. Соответствующие результаты приведены в табл. 7.2.

Таблица 7.2. Результаты вычисления функции MD4

M_1	4d7a9c83 56cb927a b9d5a578 57a7a5ee de748a3c dcc366b3 b683a020 3b2a5d9f c69d71b3 f9e99198 d79f805e a63bb2e8 45dd8e31 97e31fe5 2794bf08 b9e8c3e9
M_2	4d7a9c83 d6cb927a 29d5a578 57a7a5ee de748a3c dcc366b3 b683a020 3b2a5d9f c69d71b3 f9e99198 d79f805e a63bb2e8 45dc8e31 97e31fe5 2794bf08 b9e8c3e9
$\mathrm{MD4}(M_1) = \mathrm{MD4}(M_2)$	**5f5c1a0d 71b360461b5435da9b0d807a**

Окончание табл. 7.2

M_1	4d7a9c83 56cb927a b9d5a578 57a7a5ee de748a3c dcc366b3 b683a020 3b2a5d9f c69d71b3 f9e99198 d79f805e a63bb2e8 45dd8e31 97e31fe5 f713c240 a7b8cf69
M_2	4d7a9c83 d6cb927a 29d5a578 57a7a5ee de748a3c dcc366b3 b683a020 3b2a5d9f c69d71b3 f9e99198 d79f805e a63bb2e8 45dc8e31 97e31fe5 f713c240 a7b8cf69
$\mathrm{MD4}(M_1) = \mathrm{MD4}(M_2)$	**e0f76122c429c56cebb5e256 b809793**

Данная атака является самой эффективной, ведь ее трудоемкость минимальна и сводится к простым вычислениям.

Во многих случаях хеш-функции строятся на основе одношаговых сжимающих функций, поэтому имеется тесная связь атак на хеш-функцию с атаками на соответствующую одношаговую сжимающую функцию. В частности, последняя должна обладать практически всеми теми же свойствами, которыми обладает и сама хеш-функция.

Итеративный способ построения хеш-функции позволяет иногда при ее обращении или построении коллизий использовать метод «встречи посередине». Для защиты от этой атаки в конце сообщения обычно дописывают блоки с контрольной суммой и длинным сообщением.

Возможны атаки, использующие слабости тех схем, на базе которых построены хеш-функции. Например, для построения коллизий хеш-функций, основанных на алгоритмах блочного шифрования, можно использовать наличие слабых ключей или свойство дополнения (как это имеет место у алгоритма DES), наличие неподвижных точек (для которых $E_k(x) = x$), коллизии ключей (то есть пар различных ключей, для которых выполняется равенство $E_k(x) = E_{K'}(x)$) и т. д.

7.3. СВОЙСТВА, ОБЕСПЕЧИВАЕМЫЕ ЭЛЕКТРОННОЙ ПОДПИСЬЮ

Электронная подпись (ЭП) для сообщения является числом, зависящим от самого сообщения и от некоторого секретного, известного только подписываемому субъекту ключа. При этом предполагается, что она должна быть легко проверяемой и осуществить про-

Таблица 7.3. Свойства электронной и обычной подписей

Свойства собственноручной подписи	Свойства электронной подписи
Не зависит от подписываемого текста, всегда одинакова	Зависит от подписываемого текста, практически всегда разная
Неразрывно связана с подписывающим лицом, однозначно определяется его психофизическими свойствами; не может быть утеряна	Определяется секретным ключом, принадлежащим подписывающему лицу; может быть утеряна владельцем
Неотделима от носителя (бумаги), поэтому отдельно подписывается каждый экземпляр документа	Легко отделима от документа, поэтому верна для всех его копий
Не требует для реализации дополнительных механизмов	Требует дополнительных механизмов, реализующих алгоритмы ее вычисления и проверки
Не требует создания поддерживающей инфраструктуры	Требует создания доверенной инфраструктуры сертификатов открытых ключей

верку подписи должен иметь возможность каждый пользователь без получения доступа к секретному ключу. При возникновении спорной ситуации, связанной с отказом подписывающего от факта подписи им некоторого сообщения либо попыткой подделки подписи, третья сторона должна иметь возможность разрешить спор.

Электронная подпись позволяет решить следующие три задачи:

- осуществить аутентификацию источника сообщения;
- установить целостность сообщения;
- обеспечить невозможность отказа от факта подписи конкретного сообщения.

Использование термина «подпись» в данном контексте оправдано тем, что электронная подпись имеет много общего с обычной собственноручной подписью на бумажном документе. Собственноручная подпись также решает три перечисленные задачи, однако между ней и электронной подписью имеются существенные различия, представленные в табл. 7.3 .

7.4. АЛГОРИТМ ФОРМИРОВАНИЯ ЭЛЕКТРОННОЙ ПОДПИСИ

Алгоритм формирования ЭП состоит из двух частей. Первая часть алгоритма формирования ЭП осуществляет ее вычисление;

вторая часть алгоритма формирования электронной подписи выполняет ее проверку.

Главные требования к этим частям заключаются в исключении возможности получения подписи без использования секретного ключа и гарантировании возможности проверки подписи без знания какой-либо секретной информации.

Надежность схемы электронной подписи определяется сложностью следующих трех задач:

- подделки подписи, т. е. нахождения значения подписи под заданным документом лицом, не являющимся владельцем секретного ключа;
- создания подписанного сообщения, т. е. нахождения хотя бы одного сообщения с правильным значением подписи;
- подмены сообщения, т. е. подбора двух различных сообщений с одинаковыми значениями подписи.

Существует несколько методов построения схем ЭП.

1. Шифрование электронного документа (ЭД) на основе *симметричных алгоритмов*. Данная схема предусматривает наличие в системе третьего лица (арбитра), пользующегося доверием участников обмена. Взаимодействие пользователей данной системой производится по следующей схеме:

а) участник A зашифровывает сообщение на своем секретном ключе K_A, знание которого разделено с арбитром. Затем шифрованное сообщение передается арбитру с указанием адресата данного сообщения (информация, идентифицирующая адресата, передается также в зашифрованном виде);

б) арбитр расшифровывает полученное сообщение на ключе K_A, производит необходимые проверки и затем зашифровывает на секретном ключе участника B (K_B). Далее зашифрованное сообщение посылается участнику B вместе с информацией, что оно пришло от участника A;

в) участник B расшифровывает данное сообщение и убеждается в том, что отправителем является участник A;

г) авторизацией документа в данной схеме будет считаться сам факт зашифрования ЭД секретным ключом и передача зашифрованного ЭД арбитру. Основным преимуществом этой схемы является наличие третьей стороны, исключающей какие-либо спорные вопросы между участниками информационного обмена, т. е. в данном случае не требуется дополнительной системы арбитража ЭП. Недостатком схемы является наличие третьей стороны и использование симметричных алгоритмов шифрования. на практике эта схема не получила широкого распространения.

2. Использование *асимметричных алгоритмов* шифрования. Фактом подписания документа в данной схеме является зашифрование документа на секретном ключе его отправителя. Эта схема тоже используется довольно редко вследствие того, что длина ЭД может оказаться критичной. Одной из основных проблем, возникающих при использовании асимметричных алгоритмов, является низкая скорость проведения операций зашифрования / расшифрования. Этот факт особенно характерен для реализаций алгоритмов на устройствах с небольшими вычислительными возможностями (реализация ЭП на интеллектуальных карточках). Один из выходов — уменьшение размерности параметров системы, что чревато уменьшением стойкости алгоритма. В этом случае не требуется наличия третьей стороны, хотя она может выступать в роли сертификационного органа открытых ключей пользователей.

3. Развитием предыдущей идеи стала наиболее распространенная схема ЭП — зашифрование окончательного результата обработки ЭД *хеш-функцией при помощи асимметричного алгоритма*.

Генерация подписи происходит следующим образом:

- участник *A* вычисляет хеш-код от ЭД. Полученный хеш-код проходит процедуру преобразования с использованием секретного ключа участника *A*. После чего полученное значение (которое и является ЭП) вместе с ЭД отправляется участнику *B*;
- участник B должен получить ЭД с ЭП и сертифицированный открытый ключ участника *A*, а затем произвести расшифрование на нем ЭП; сам ЭД подвергается операции хеширования, после чего результаты сравниваются, и если они совпадают, то ЭП признается истинной, в противном случае — ложной.

Эффективность реализации данной схемы по сравнению с предыдущей состоит в применении процедур асимметричного шифрования к хеш-коду ЭД, который значительно короче самого ЭД.

Стойкость данного типа ЭП основана на стойкости асимметричных алгоритмов шифрования и применяемых хеш-функций.

Кроме вышеперечисленных существуют другие варианты построения схем ЭП (групповая подпись, неоспариваемая подпись, доверенная подпись и т. д.). Появление этих разновидностей обусловлено многообразием задач, решаемых с помощью электронных технологий передачи и обработки ЭД.

В общем случае подписанный ЭД выглядит как пара, состоящая из бинарных строк (M, S), где M представляет собой ЭД, а S — решение уравнения $F_K(S) = M$, где F_K является функцией с секретом. В связи с вышеизложенным определением ЭП можно выделить следующие ее свойства:

- является не подделываемой, поскольку решить уравнение $F_K(S) = M$ может только обладатель секрета K;
- однозначно идентифицирует автора, т. е. человека, подписавшего данный документ;
- верификация (проверка) подписи производится на основе знания функции F_K;
- является непереносимой на другой ЭД (исключение составляет случай, когда для используемой хеш-функции найдены коллизии);
- ЭД с ЭП может передаваться по открытым каналам, поскольку любое изменение ЭД приведет к тому, что процедура проверки ЭП обнаружит данный факт.

7.5. СХЕМЫ ЭЛЕКТРОННОЙ ПОДПИСИ

Алгоритм электронной подписи RSA. Первой и наиболее известной во всем мире конкретной системой ЭП стала система RSA, математическая схема которой была разработана в 1977 г. в Массачусетском технологическом институте США.

Для реализации данной ЭП сначала необходимо вычислить пару ключей (секретный ключ и открытый ключ). Для этого отправитель (автор) электронных документов вычисляет два больших простых числа P и Q, затем находит их произведение

$$N = P \cdot Q$$

и значение функции Эйлера от N

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

Далее отправитель вычисляет число E из условий:

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

Обычно в качестве E берут простые числа, содержащие небольшое количество единичных битов в двоичной записи, например простые числа Ферма 17, 257 или 65 537. Слишком малые значения E, например 3, и число D, мультипликативно обратное к числу E по модулю $\varphi(N)$, потенциально могут ослабить безопасность схемы RSA. Затем вычисляет число D из условий

$$D < N,\ ED \equiv 1 (\text{mod}\ \varphi(N)).$$

Чаще всего D вычисляется с помощью расширенного алгоритма Евклида.

Пара чисел (E, N) является открытым ключом. Эту пару чисел автор передает партнерам по переписке для проверки его цифровых подписей. Число D сохраняется автором как секретный ключ для подписывания.

Допустим, что отправитель хочет подписать сообщение M перед его отправкой. Сначала сообщение M (блок информации, файл, таблица) сжимают с помощью хеш-функции $h(M)$ в целое число m:

$$m = h(M).$$

Затем вычисляют электронную подпись S под электронным документом M, используя хеш-значение m и секретный ключ D:

$$S = m^D \pmod{N}.$$

Пара (M, S) передается партнеру-получателю как электронный документ M, подписанный цифровой подписью S, причем подпись S сформирована обладателем секретного ключа D.

После приема пары (M, S) получатель вычисляет хеш-значение сообщения M двумя разными способами. Прежде всего он восстанавливает хеш-значение m', применяя криптографическое преобразование подписи S с использованием открытого ключа E:

$$m' = S^E \pmod{N}.$$

Кроме того, он находит результат хеширования принятого сообщения M с помощью такой же хеш-функции $h(M)$:

$$m = h(M).$$

Если соблюдается равенство вычисленных значений, т. е.

$$S^E \pmod{N} = h(M),$$

то получатель признает пару (M, S) подлинной. Доказано, что только обладатель секретного ключа D может сформировать электронную подпись S по документу M, а определить секретное число D по открытому числу E не легче, чем разложить модуль N на множители.

Кроме того, можно строго математически доказать, что результат проверки электронной подписи S будет положительным только в том случае, если при вычислении S был использован секретный ключ D, соответствующий открытому ключу E. Поэтому открытый ключ E иногда называют «идентификатором» подписавшего.

Недостатками алгоритма электронной подписи RSA являются:

- при вычислении модуля N ключей E и D для системы электронной подписи RSA необходимо проверять большое количество дополнительных условий, что сделать практически трудно. Невыполнение любого из этих условий делает возможным фальсификацию электронной подписи со стороны того, кто обнаружит такое невыполнение. При подписании важных документов нельзя допускать такую возможность даже теоретически;
- для обеспечения криптостойкости электронной подписи RSA по отношению к попыткам фальсификации на уровне, например, национального стандарта США на шифрование информации, необходимо использовать при вычислениях N, D и E целые числа не менее 2^{512} (или около 10^{154}) каждое, что требует больших вычислительных затрат, превышающих на 20 — 30 % вычислительные затраты других алгоритмов электронной подписи при сохранении того же уровня криптостойкости;
- электронная подпись RSA уязвима к так называемой мультипликативной атаке. Иначе говоря, алгоритм электронной подписи RSA позволяет злоумышленнику без знания секретного ключа D сформировать подписи под теми документами, у которых результат хеширования можно вычислить как произведение результатов хеширования уже подписанных документов.

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

Рассмотрим подробнее алгоритм электронной подписи Эль-Гамаля. Для того чтобы генерировать пару ключей (открытый ключ — секретный ключ), сначала выбирают некоторое большое простое целое число P и большое целое число G, причем $G < P$. Отправитель и получатель подписанного документа используют при вычислениях одинаковые большие целые числа P (~10^{308} или ~2^{1024}) и G (~10^{154} или ~2^{512}), которые не являются секретными.

Отправитель выбирает случайное целое число X, $1 < X < (P - 1)$, и вычисляет

$$Y = G^X \bmod P.$$

Число Y является открытым ключом, используемым для проверки подписи отправителя. Число Y открыто передается всем потенциальным получателям документов. Число X является секретным ключом отправителя для подписывания документов и должно храниться в секрете. Для того чтобы подписать сообщение M, сначала отправитель хеширует его с помощью хеш-функции $h(M)$ в целое число m:

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

и генерирует случайное целое число K, $1 < K < (P - 1)$, такое, что K и $(P - 1)$ являются взаимно простыми. Затем отправитель вычисляет целое число a:

$$a = G^K \pmod{P}$$

и, применяя расширенный алгоритм Евклида, вычисляет с помощью секретного ключа X целое число b из уравнения

$$m = Xa + Kb \pmod{(P - 1)}.$$

Пара чисел (a, b) образует электронную подпись S:

$$S = (a, b),$$

проставляемую под документом M. Тройка чисел (M, a, b) передается получателю, в то время как пара чисел (X, K) держится в секрете.

После приема подписанного сообщения (M, a, b) получатель должен проверить, соответствует ли подпись $S = (a, b)$ сообщению M. Для этого получатель сначала вычисляет по принятому сообщению M число

$$m = h(M),$$

т. е. хеширует принятое сообщение M.

Затем получатель вычисляет значение $A = Y^a a^b \pmod{P}$ и признает сообщение M подлинным, если, и только если

$$A = G^m \pmod{P}.$$

Иначе говоря, получатель проверяет справедливость соотношения

$$Y^a a^b \pmod{P} = G^m \pmod{P}.$$

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

Следует отметить, что выполнение каждой подписи по методу Эль-Гамаля требует нового значения K, причем это значение должно выбираться случайным образом. Если нарушитель раскроет когда-либо значение K, повторно используемое отправителем, то он сможет раскрыть секретный ключ X отправителя. Таким образом K — сессионный ключ.

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

- при заданном уровне стойкости алгоритма цифровой подписи целые числа, участвующие в вычислениях, имеют запись на 25 % короче, что уменьшает сложность вычислений почти в два раза и позволяет заметно сократить объем используемой памяти;
- при выборе модуля P достаточно проверить, что это число является простым и что у числа $(P–1)$ имеется большой простой множитель (т. е. всего два достаточно просто проверяемых условия);
- процедура формирования подписи по схеме Эль-Гамаля не позволяет вычислять электронные подписи под новыми сообщениями без знания секретного ключа (как было в RSA).

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

Алгоритм электронной подписи Шнорра. Безопасность схемы подписи К. Шнорра (K. Schnorr) также опирается на трудность вычисления дискретных логарифмов. Для генерации пары ключей сначала выбираются два простых числа p и q так, чтобы q было сомножителем $p–1$. Затем выбирается такое a, не равное 1, что $a^q = 1 \pmod{p}$. Все эти числа могут быть свободно опубликованы и использоваться группой пользователей.

Для генерации конкретной пары ключей выбирается случайное число, меньшее q: оно служит закрытым ключом s. Затем вычисляется открытый ключ

$$v = a^{-s} \pmod{p}.$$

Алгоритм Шнорра можно использовать в качестве протокола электронной подписи сообщения M, при этом используется однонаправленная хеш-функция $h(M)$:

- отправитель выбирает случайное число r, меньшее q, и вычисляет $x = a^r \pmod p$. Это стадия предварительных вычислений;
- отправитель объединяет M и x и хеширует результат: $e = h(M, x)$;
- отправитель вычисляет $y = (r + se) \pmod q$. Подписью являются значения e и y, он посылает их получателю;
- получатель вычисляет $x' = a^y v^e \pmod p$. Затем он проверяет, что хеш-значение для объединения M и x' равно e, т. е. $e = h(M, x')$. Если это так, то получатель считает подпись верной.

В своей работе Шнорр приводит следующие новые свойства своего алгоритма: бóльшая часть вычислений, нужных для генерации подписи и не зависящих от подписываемого сообщения, может быть выполнена на стадии предварительных вычислений. Следовательно, эти вычисления могут быть выполнены во время простоя и не влияют на скорость подписания. Вскрытие, направленное против стадии предварительных вычислений, не имеет практической ценности.

При одинаковом уровне безопасности длина подписей алгоритма Шнорра короче, чем для RSA. Например, при 140-битовом q длина подписей равна всего лишь 212 битам, меньше половины длины подписей RSA. Подписи алгоритма Шнорра также намного короче подписи алгоритма Эль-Гамаля.

Алгоритм электронной подписи DSA. Алгоритм электронной подписи DSA (Digital Signature Algorithm) предложен в 1991 г. в национальном институте стандартов США для использования в стандарте электронной подписи DSS (Digital Signature Standard). Алгоритм DSA является развитием алгоритмов электронной подписи Эль-Гамаля и К. Шнорра.

Отправитель и получатель электронного документа используют при вычислении большие целые числа G и P — простые числа, длиной L бит каждое ($512 < L < 1\,024$), и q — простое число длиной 160 бит (делитель числа $(P - 1)$). Числа G, P, q являются открытыми и могут быть общими для всех пользователей сети.

Отправитель выбирает случайное целое число X, $1 < X < q$. Число X является секретным ключом отправителя для формирования электронной подписи. Затем отправитель вычисляет значение

$$Y = G^X \pmod P.$$

Число Y является открытым ключом для проверки подписи отправителя и передается всем получателям документов.

Данный алгоритм также предусматривает использование односторонней функции хеширования $h(M)$. В стандарте DSS для этого

определен алгоритм безопасного хеширования SHA (Secure Hash Algorithm).

Для того чтобы подписать документ M, отправитель хеширует его в целое хеш-значение m:

$$m = h(M),\ 1 < m < q,$$

затем генерирует случайное целое число K, $1 < K < q$, и вычисляет число r:

$$r = (G^K \pmod P) \pmod q.$$

Далее отправитель вычисляет с помощью секретного ключа X целое число s:

$$s = ((m + rX)/K) \pmod q.$$

Пара чисел r и s образует электронную подпись $S = (r, s)$ под документом M. Таким образом, подписанное сообщение представляет собой тройку чисел (M, r, s). Получатель подписанного сообщения (M, r, s) проверяет выполнение условий

$$0 < r < q,\ 0 < s < q$$

и отвергает подпись, если хотя бы одно из этих условий не выполнено.

Затем получатель вычисляет значение $w = (1/s) \pmod q$, хеш-значение

$$m = h(M) \text{ и числа}$$

$$u_1 = (mw) \pmod q,$$

$$u_2 = (rw) \pmod q.$$

Далее получатель с помощью открытого ключа Y вычисляет значение

$$v = ((G^{u_1} Y^{u_2}) \pmod P) \pmod q$$

и проверяет выполнение условия $v = r$.

Если условие $v = r$ выполняется, тогда подпись $S = (r, s)$ под документом M признается получателем подлинной.

Можно строго математически доказать, что последнее равенство будет выполняться тогда и только тогда, когда подпись $S = (r, s)$ под документом M получена с помощью именно того секретного ключа X, из которого был получен открытый ключ Y. Таким образом, можно надежно удостовериться, что отправитель сообщения владеет именно данным секретным ключом X (не раскрывая при

этом значения ключа X) и что отправитель подписал именно данный документ M.

По сравнению с алгоритмом цифровой подписи Эль-Гамаля алгоритм DSA имеет следующие основные преимущества:

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

Недостатком алгоритма DSA является то, что при подписывании и проверке подписи приходится выполнять сложные операции деления по модулю q:

$$s = ((m + rX)/K) \pmod q.$$
$$w = (1/s) \pmod q,$$

что не позволяет получать максимальное быстродействие.

Следует отметить, что реальное исполнение алгоритма DSA может быть ускорено с помощью выполнения предварительных вычислений. Заметим, что значение r не зависит от сообщения M и его хеш-значения m. Можно заранее создать строку случайных значений K u затем для каждого из этих значений вычислить значения r. Можно также заранее вычислить обратные значения $K - 1$ для каждого из значений K. Затем, при поступлении сообщения M, вычисляют значение s для данных значений r и $K - 1$. Эти предварительные вычисления значительно ускоряют работу алгоритма DSA.

Схема подписи Фиата — Шамира (Fiat — Shamir). Эта схема подписи предполагает наличие арбитра. Перед выдачей любых закрытых ключей арбитр выбирает случайный модуль n, который является произведением двух больших простых чисел p и q. В реальной жизни длина n должна быть не меньше 512 битов и, лучше, как можно ближе к 1 024 битам; N может быть общим для группы контролеров. Главным преимуществом схемы электронной подписи Fiat — Shamir по сравнению с RSA является ее скорость: для Fiat — Shamir нужно всего лишь от 1 до 4 % модульных умножений, используемых в RSA.

Для генерации открытого и закрытого ключей отправителю доверенный арбитр выбирает k различных чисел v_1, ..., v_k каждое из

которых является квадратичным остатком по модулю n. Другими словами, выбираются v_i так, чтобы уравнение $x^2 \approx v_i \pmod n$ имело решение и существовало $v_i^{-1} \pmod n$. Строка v_1, ..., v_k будет открытым ключом отправителя. Затем вычисляются наименьшие s_i, для которых

$$s_i \equiv \text{sqrt } (v_i^{-1}) \pmod n.$$

Строка s_1, ..., s_k служит закрытым ключом отправителя.

Отправитель выбирает t случайных целых чисел в диапазоне от 1 до $n - r_1$, ..., r_t и вычисляет x_1, ..., x_t такие, что $x_i = r_i^2 \pmod n$.

Отправитель хеширует объединение сообщения и строки x, создавая битовый поток: $H\{m, x_1, x_2, \ldots x_t\}$. Он использует первые kt битов этой строки в качестве значений b_{ij}, где i пробегает от 1 до t, а j от 1 до k. Отправитель вычисляет $y_1, y_2, \ldots y_t$, где $y_i = r_i (s_1^{bi}{}_1 \ldots s_k^{bi}{}_k) \pmod n$. (Для каждого i он перемножает вместе значения s_i в зависимости от случайных значений b_{ij}: если $b_{ij} = 1$, то s_i участвует в вычислениях, если $b_{ij} = 0$, то нет.)

Отправитель посылает получателю m, все биты b_{ij} и все значения y_i. У получателя уже есть открытый ключ отправителя: v_1, ..., v_k.

Получатель вычисляет $z_1, z_2, \ldots z_t$, где $z_i = v^2(v_1^{bi}{}_1 \ldots v_k^{bi}{}_k) \pmod n$. (И снова получатель выполняет умножение в зависимости от значений b_{ij}.) Также обратите внимание, что z_i должно быть равно x_i. Получатель проверяет, что первые kt битов $h\{m, z_1, z_2, \ldots z_t\}$ — это значения b_{ij}, которые прислал ему отправитель.

Безопасность схемы подписи пропорциональна $1/2^{kt}$. Она также зависит от сложности разложения n на множители. А. Фиат и А. Шамир показали, что подделка подписи облегчается, если сложность разложения n на множители заметно меньше 2^{kt}. Кроме того, из-за вскрытия «методом дня рождения», они рекомендуют повысить kt от 20 по крайней мере до 72, предлагая $k = 9$ и $t = 8$.

Сильвия Микали и Ади Шамир улучшили протокол Fiat — Shamir. Они выбирали $v_1, v_2, \ldots v_k$ так, чтобы они были первыми k простыми числами, т. е.

$$v_1 = 1,\ v_2 = 3,\ v_3 = 5 \text{ и т. д.}$$

Это — открытый ключ. Закрытым ключом $s_1, s_2, \ldots, s_k$ служат случайные квадратные корни, определяемые как $s_i = \text{sqrt } (v_i^{-1}) \pmod n$.

В этой версии у каждого участника должен быть свой n. Такая модификация облегчает проверку подписей, не влияя на время генерации подписей и их безопасность.

7.6. СЛЕПАЯ ЭЛЕКТРОННАЯ ПОДПИСЬ

В отличие от обычных схем ЭП, описанных ранее, *схемы слепой подписи* (иногда называемые *схемами подписи вслепую*) являются двусторонними протоколами между отправителем A и стороной B, подписывающей документ. Основная идея этих схем заключается в следующем. Отправитель A посылает порцию информации стороне B, которую B подписывает и возвращает A. Используя полученную подпись, сторона A может вычислить подпись стороны B на более важном для себя (для A) сообщении m. По завершении этого протокола сторона B ничего не знает ни о сообщении m, ни о подписи под этим сообщением.

Цель слепой подписи состоит в том, чтобы воспрепятствовать подписывающему лицу B ознакомиться с сообщением стороны A, которое он (B) подписывает, и с соответствующей подписью (B) под этим сообщением. Поэтому в дальнейшем подписанное сообщение невозможно связать со стороной A (подписывает-то B).

В частности, это может быть важно при организации анонимных безналичных расчетов с использованием так называемой электронной наличности (цифровых денег), когда сообщение m могло бы представлять денежную сумму, которую A хочет потратить. Сообщение m с подписью $s_B(m)$ предъявляется банку B для оплаты, банк B не может проследить, кто именно из клиентов предъявляет подписанный документ. Это позволяет пользователю A остаться анонимным.

Для построения протокола слепой подписи необходимы следующие компоненты:

1) механизм обычной электронной подписи для подписывающей строны B: пусть $s_B(X)$ обозначает подпись стороны B на документе X;

2) функции $f(m)$ и $g(m)$ (известные только отправителю) такие, что $g(s_B(f(m))) = s_B(m)$, где $f\,(m)$ — маскирующая функция; $g\,(m)$ — демаскирующая функция; $f\,(m)$ — замаскированное сообщение m.

При выборе s_B, f и g существует ряд ограничений. Выберем в качестве алгоритма подписи s_B для стороны B схему электронной подписи RSA с открытым ключом N и секретным ключом D, причем $N = PQ$, т. е. равно произведению двух больших случайных простых чисел. Пусть k — некоторое фиксированное целое число, взаимно простое с N, т. е. НОД $(N, k) = 1$. Маскирующая функция f: $Z_n \to Z_n$ определяется как

$$f\,(m) = mk^E \pmod N,$$

а демаскирующая функция g: $Z_n \to Z_n$ определяется как

$$g(m) = k^{-1}m \pmod N.$$

При таком выборе f, g и s получаем

$$g\,(sB\,(f\,(m))) = g(sB\,(mkE \pmod N)) = g\,(mDk \pmod N) = \\ = mD \pmod N = sB\,(m),$$

что соответствует требованию 2.

Согласно протоколу слепой подписи, который предложил Д. Чом, отправитель A сначала получает подпись стороны B на замаскированном сообщении m^*. Используя эту подпись, сторона A вычисляет подпись B на заранее выбранном сообщении m, где $0 < m < N–1$; при этом стороне B ничего неизвестно ни о значении m, ни о подписи, связанной с m.

Пусть сторона B имеет для подписи по схеме RSA открытый ключ N и секретный ключ D. Пусть k — случайное секретное целое число, выбранное стороной A и удовлетворяющее условиям $0 < k < N - 1$ и НОД $(N, k) = 1$.

Протокол слепой подписи Д. Чома. Протокол включает в себя ряд шагов.

Шаг 1. Отправитель A вычисляет замаскированное сообщение $m^* = mk^E \pmod N$ и посылает его стороне B.

Шаг 2. Подписывающая сторона B вычисляет подпись

$$s^* = (m^*)D \pmod N$$

и отправляет эту подпись стороне A.

Шаг 3. Сторона A вычисляет подпись $s = k^{-1}\,s^* \pmod N$, которая является подписью B на сообщении m.

Нетрудно видеть, что

$$(m^*)^D = (mk^E)^D \equiv m^Dk \pmod N,$$

поэтому

$$k^{-1}\,s^* \equiv m^Dkk^{-1} \equiv m^D \pmod N.$$

7.7. НЕОСПОРИМАЯ ЭЛЕКТРОННАЯ ПОДПИСЬ

Неоспоримая подпись, как и обычная электронная подпись, зависит от подписанного документа и секретного ключа. Однако, в отличие от обычных электронных подписей, неоспоримая подпись не может быть верифицирована без участия лица, поставившего эту подпись.

Рассмотрим два возможных сценария применения неоспоримой подписи.

Сценарий 1. Сторона *A* (клиент) хочет получить доступ в защищенную зону, контролируемую стороной B (банком). Этой защищенной зоной может быть, например депозитарий (хранилище ценностей клиентов). Сторона B требует от *A* поставить до предоставления клиенту доступа на заявке о допуске в защищенную зону подпись, время и дату. Если *A* применит неоспоримую подпись, тогда сторона B не сможет впоследствии доказать кому-либо, что *A* получил допуск без непосредственного участия *A* в процессе верификации подписи.

Сценарий 2. Предположим, что известная корпорация *A* разработала пакет программного обеспечения. Чтобы гарантировать подлинность пакета и отсутствие в нем вирусов, сторона *A* подписывает этот пакет неоспоримой подписью и продает его стороне *B*. Сторона B решает сделать копии этого пакета программного обеспечения и перепродать его третьей стороне *C*. При использовании стороной *A* неоспоримой подписи сторона C не сможет убедиться в подлинности этого пакета программного обеспечения и отсутствии в нем вирусов без участия стороны *A*.

Конечно, этот сценарий не препятствует стороне *B* поставить на пакете свою подпись, но тогда для стороны *B* будут утрачены все маркетинговые преимущества, связанные с использованием торговой марки корпорации *A*. Кроме того, будет легче раскрыть мошенническую деятельность стороны *B*.

Алгоритм неоспоримой электронной подписи. Рассмотрим алгоритм неоспоримой ЭП, разработанный Д. Чомом. Сначала опишем алгоритм генерации ключей, с помощью которого сторона *A*, подписывающая документ, выбирает секретный ключ и соответствующий открытый ключ.

Сторона *A* должна выполнить следующее:

а) выбрать случайное простое число $p = 2q + 1$, где q — также простое число;

б) выбрать генераторное число α для подгруппы порядка q в циклической группе Z_p^*:

1) выбрать случайный элемент $\beta \in Z_p^*$ и вычислить $\alpha = \beta^{(p-1)/q} \pmod p$;

2) если $\alpha = 1$, тогда возвратиться к шагу б) 1);

в) выбрать случайное целое $x \in \{1, 2, ..., q - 1\}$ и вычислить $y = \alpha^x \pmod p$;

г) для стороны *A* открытый ключ равен (p, α, y), секретный ключ равен x.

Согласно алгоритму неоспоримой подписи Д. Чома, сторона *A* подписывает сообщение m, принадлежащее подгруппе порядка q в Z_p^*. Любая сторона B может проверить эту подпись при участии *A*.

В работе алгоритма неоспоримой подписи можно выделить два этапа:

- генерация подписи;
- верификация подписи.

На *этапе генерации* подписи сторона *A* вычисляет $s = m^x \pmod p$, где s — подпись стороны *A* на сообщении m. Сообщение m с подписью s отсылается стороне *B*.

Этап верификации подписи выполняется стороной B с участием стороны *A* и включает следующие шаги:

1) *B* получает подлинный открытый ключ (p, α, y) стороны *A*;

2) *B* выбирает два случайных секретных целых числа a и b: a, $b \in \{1, 2, ..., q - 1\}$;

3) *B* вычисляет $z = s^a y^b \pmod p$ и отправляет значение z стороне *A*;

4) *A* вычисляет $w = (z)^{1/x} \pmod p$, где $w^{-1} \equiv 1 \pmod q$, и отправляет значение w стороне *B*;

5) *B* вычисляет $w' = m^a \alpha^b \pmod p$ и признает подпись s подлинной, если только $w = w'$.

Убедимся, что проверка подписи s работает:

$$w \equiv (z)^{1/x} \equiv (s^a y^b)^{1/x} \equiv (m^{xa} \alpha^{xb})^{1/x} \equiv m^a \alpha^b \equiv w' \pmod p.$$

Можно показать, что с высокой степенью вероятности злоумышленник не сможет заставить B принять фальшивую подпись. Предположим, что s представляет собой подделку подписи стороны *A* на сообщении m, т. е. $s \neq m^x \pmod p$. Тогда вероятность принятия стороной *B* этой подписи в данном алгоритме составляет только $1/q$, причем эта вероятность не зависит от вычислительных ресурсов злоумышленника. Подписавшая сторона *A* при некоторых обстоятельствах могла бы попытаться отказаться от своей подлинной подписи одним из трех способов:

- отказаться от участия в протоколе верификации;
- некорректно выполнить протокол верификации;
- объявить подпись фальшивой, даже если протокол верификации оказался успешным.

Отречение от подписи первым способом рассматривалось бы как очевидная попытка неправомерного отказа. Против второго и третьего способов бороться труднее: здесь требуется специальный протокол дезавуирования. Этот протокол определяет, пытается ли подписавшая сторона *A* дезавуировать правильную подпись

s или эта подпись является фальшивой. В этом протоколе, по существу, дважды применяется протокол верификации и затем производится проверка с целью убедиться, что сторона A выполняет этот протокол корректно.

Протокол дезавуирования для схемы неоспоримой подписи Д. Чома включает следующие шаги:

1) B принимает от стороны A сообщение m с подписью s и получает подлинный открытый ключ (p, α, y) стороны A;

2) B выбирает случайные секретные целые числа $a, b \in \{1, 2, ..., q-1\}$, вычисляет $z = s^a y^b \pmod p$ и отправляет значение z стороне A;

3) A вычисляет $w = (z)^{1/x} \pmod p$, где $w^{-1} \equiv 1 \pmod q$, и отправляет значение w стороне B;

4) если $w = m^a a^b \pmod p$, тогда B признает подпись s подлинной и выполнение протокола прекращается;

5) B выбирает случайные секретные целые числа $a', b' \in \{1, 2, ..., q-1\}$, вычисляет $z' = s^{a'} y^{b'} \pmod p$ и отправляет значение z' стороне A;

6) A вычисляет $w' = (z')^{1/x} \pmod p$ и отправляет значение w' стороне B;

7) если $w' = m^{a'}\alpha^{b'} \pmod p$, тогда B принимает подпись s и выполнение протокола останавливается;

8) B вычисляет $c = (w\alpha^{-b})^{a'} \pmod p$, $c' = (w'\alpha^{-b'})^{a} \pmod p$. Если $c = c'$, тогда B заключает, что подпись s фальшивая; в противном случае B делает вывод, что подпись s подлинная, а сторона A пытается дезавуировать подпись s.

Нетрудно убедиться в том, что этот протокол достигает поставленной цели. Пусть m — сообщение и предположим, что s — подпись стороны A под сообщением m. Если подпись s фальшивая, т. е. $s \neq m^x \pmod p$, и если стороны A и B следуют протоколу должным образом, тогда $w = w'$ (и поэтому справедливо заключение B, что подпись s фальшивая). Пусть s на самом деле является подписью стороны A под сообщением m, т. е. $s = m^x \pmod p$. Предположим, что B точно следует протоколу, а A не следует. Тогда вероятность того, что $w = w'$ (и A преуспевает в дезавуировании подписи), составляет только $1/q$.

Следует отметить, что третья сторона С никогда не должна принимать в качестве доказательства подлинности подписи s запись стороной B протокола верификации, поскольку сторона B может выдумать успешную запись шага 2 и последующих шагов протокола верификации без участия подписывающей стороны A.

Неоспоримая подпись может быть верифицирована только путем непосредственного взаимодействия с подписывающей стороной A.

7.8. ЭЛЕКТРОННАЯ ПОДПИСЬ НА ОСНОВЕ ГОСТ Р 34.10–2012

ГОСТ Р 34.10—2012 основан на эллиптических кривых. Стойкость алгоритма основывается на сложности вычисления дискретного логарифма в группе точек эллиптической кривой, а также на стойкости хеш-функции. Для ГОСТ Р 34.10—2012 используется хеш-функция по ГОСТ Р 34.11—2012.

Стандарт ГОСТ Р 34.10—2012 использует ту же схему формирования электронной подписи, что и ГОСТ Р 34.10—2001. Новый стандарт отличается наличием дополнительного варианта параметров схем (соответствующего длине секретного ключа порядка 512 бит) и требованием использования функций хеширования ГОСТ Р 34.11—2012: первый вариант требований к параметрам (такой же, как в ГОСТ Р 34.10—2001, соответствующий длине секретного ключа порядка 256 бит) предусматривает использование хеш-функции с длиной хеш-кода 256 бит, дополнительный вариант требований к параметрам предусматривает использование хеш-функции с длиной хеш-кода 512 бит.

После подписывания сообщения M к нему дописывается электронная подпись размером 512 или 1 024 бит и текстовое поле. B текстовом поле могут содержаться, например, дата и время отправки или различные данные об отправителе.

Данный алгоритм не описывает механизм генерации параметров, необходимых для формирования подписи, а только определяет, каким образом на основании таких параметров можно получить электронную подпись. Механизм генерации параметров определяется на месте в зависимости от разрабатываемой системы.

Параметрами схемы электронной подписи являются:

- простое число p-модуль эллиптической кривой;
- эллиптическая кривая E, задаваемая своим инвариантом $J(E)$ или коэффициентами a, b;
- целое число m — порядок группы точек эллиптической кривой E;
- простое число q — порядок циклической подгруппы группы точек эллиптической кривой E, для которого выполнены следующие условия:

$$\begin{cases} m = nq, n \in Z, n \geq 1 \\ 2^{254} < q < 2^{256} \text{ или } 2^{508} < q < 2^{512} \end{cases};$$

- точка $P \neq 0$ эллиптической кривой E, с координатами (x_P, y_P), удовлетворяющая равенству $qP = 0$;

- хеш-функция $h(M)$: $V^* \to V_L$, отображающая сообщения, представленные в виде двоичных векторов произвольной конечной длины, в двоичные вектора длины L бит. Хеш-функция определена в ГОСТ Р 34.11-2012. Если $2^{254} < q < 2^{256}$, то $L = 256$. Если $2^{508} < q < 2^{512}$, то $L = 512$.

Каждый пользователь схемы электронной подписи должен обладать личными ключами:

- ключом подписи — целым числом d, удовлетворяющим неравенству $0 < d < q$;
- ключом проверки подписи — точкой эллиптической кривой Q с координатами (x_q, y_q), удовлетворяющими равенству $dP = Q$.

Алгоритм 1. ***Для получения электронной подписи*** под сообщением $M \in V^*$ необходимо выполнить ряд действий (шагов) по алгоритму 1.

Шаг 1. Вычислить хеш-код сообщения M: $\bar{h} = h(M)$.

Шаг 2. Вычислить целое число α, двоичным представлением которого является вектор $\bar{h}$, и определить $e \equiv a \pmod q$. Если $e = 0$, то определить $e = 1$.

Шаг 3. Сгенерировать случайное (псевдослучайное) целое k, удовлетворяющее неравенству $0 < k < q$.

Шаг 4. Вычислить точку эллиптической кривой $C = kP$ и определить

$r \equiv x_c \pmod q$, где x_c — x-координата точки C. Если $r = 0$, то вернуться к шагу 3.

Шаг 5. Вычислить значение $s \equiv (rd + ke) \pmod q$. Если $s = 0$, то вернуться к шагу 3.

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

Исходными данными этого процесса являются ключ подписи d и подписываемое сообщение M, а выходным результатом — электронная подпись ζ.

Алгоритм 2. ***Для проверки электронной подписи*** ζ под полученным сообщением M необходимо выполнить некоторые действия (шаги) по алгоритму 2.

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

Шаг 2. Вычислить хеш-код полученного сообщения M: $\bar{h} = h(M)$.

Шаг 3. Вычислить целое число α, двоичным представлением которого является вектор $\bar{h}$ и определить $e \equiv \alpha \pmod q$. Если $e = 0$, то определить $e = 1$.

Шаг 4. Вычислить значение $v \equiv e^{-1} \pmod q$.

Шаг 5. Вычислить значение $z_1 \equiv sv \pmod q$, $z_2 \equiv -rv \pmod q$.

Шаг 6. Вычислить точку эллиптической кривой $C = z_1P + z_2Q$ и определить $R \equiv x_c \pmod q$, где x_c — x-координата точки C.

Шаг 7. Если выполнено равенство $R = r$, то подпись принимается, в противном случае — подпись не верна.

Исходными данными для этого процесса являются подписанное сообщение M, электронная подпись ζ и ключ проверки подписи Q, а выходным результатом — свидетельство о достоверности или ошибочности данной подписи.

7.9. АТАКИ НА ЭЛЕКТРОННУЮ ПОДПИСЬ

Стойкость большинства схем ЭП зависит от стойкости алгоритмов шифрования и хеш-функций, использованных в ЭП. Поэтому атаки на ЭП в ряде случаев аналогичны атакам на хеш-функции и алгоритмы шифрования, которые в ней используются.

Можно выделить следующие ***атаки на схемы ЭП***:

1) атака с *использованием открытого ключа*, которая включает в себя:

- атаку на основе известных сообщений. Злоумышленник обладает допустимыми подписями набора электронных документов, которые ему известны;
- адаптивную атаку на основе выбранных сообщений. Криптоаналитик может получить подписи электронных документов, которые он выбирает сам. При безошибочной реализации современных алгоритмов ЭП получение закрытого ключа алгоритма является практически невозможной задачей (из-за вычислительной сложности задач, на которых ЭП построена). Гораздо более вероятен поиск криптоаналитиком коллизий первого и второго рода;

2) атака с *использованием коллизий первого и второго рода*. Коллизией первого рода является такая возможность подбора злоумышленником документа к данной подписи, чтобы подпись к нему подходила. Однако в подавляющем большинстве случаев такой документ может быть только один. Причина в том, что документ представляет из себя осмысленный текст, и подобрать какой-то другой подходящий текст почти невозможно. Злоумышленники, подделывая документы, подбирают под текст произвольный набор данных и вставляют его в служебные поля. Это позволяет повысить вероятность подбора документа под ЭП, хотя она и остается очень малой.

Коллизией второго рода называется получение двух документов с одинаковой подписью. При этом вычислительно сложная атака возможна из-за ошибок реализации алгоритмов или слабостей алгоритмов хеширования. В этом случае злоумышленник фабрикует два документа с одинаковой подписью и в нужный момент подменяет один другим. В частности, таким образом можно провести атаку на SSL-сертификаты и алгоритм хеширования MD5;

3) *социальные атаки*. Социальные атаки направлены не на взлом алгоритмов цифровой подписи, а на манипуляции с открытым и закрытым ключами:

- злоумышленник, укравший закрытый ключ, может подписать любой документ от имени владельца ключа;
- злоумышленник может обманом заставить владельца подписать какой-либо документ, например, используя протокол слепой подписи;
- злоумышленник может подменить открытый ключ владельца на свой собственный, выдавая себя за него.

Использование протоколов обмена ключами и защита закрытого ключа от несанкционированного доступа позволяет снизить опасность социальных атак.

Возможные результаты атак на ЭП:

- полный взлом цифровой подписи. Получение закрытого ключа, что означает полный взлом алгоритма;
- универсальная подделка цифровой подписи. Нахождение алгоритма, аналогичного алгоритму подписи, что позволяет подделывать подписи для любого ЭД;
- выборочная подделка ЭП. Возможность подделывать подписи для документов, выбранных криптоаналитиком;
- экзистенциальная подделка электронной подписи. Возможность получения допустимой подписи для какого-то документа, не выбираемого криптоаналитиком.

Адаптивная атака на основе выбранных сообщений является одной из самых опасных атак, и при анализе алгоритмов ЭП на криптостойкость нужно рассматривать именно ее (если нет каких-либо особых условий).

Главными условиями защищенности электронных документов с использованием электронных подписей являются: использование криптостойких закрытых ключей; их надежная сохранность; проверка ЭП на подлинность; подписание проверенных документов; снижение вероятности влияния человеческого фактора.

ВОПРОСЫ И ЗАДАНИЯ ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ ЗНАНИЙ

1. Дайте определение однонаправленной хеш-функции.
2. Каким требованиям (условиям) должна удовлетворять однонаправленная хеш-функция?
3. Что такое электронная подпись?
4. Какие задачи позволяет решить электронная подпись?
5. Что такое хеш-функция?
6. Чем отличается электронная подпись от хеш-функции?
7. Какая хеш-функция считается криптографически стойкой?
8. Перечислите виды атак на хеш-функцию.
9. Какие существуют виды атак на электронную подпись?
10. Что такое слепая электронная подпись? Каково ее назначение?
11. Что такое электронная подпись с временной меткой? Назовите возможности штампа времени.
12. Что такое неоспоримая электронная подпись?
13. Опишите алгоритм неоспоримой электронной подписи, разработанный Д. Чомом.
14. Что такое парадокс о днях рождениях? Как он используется?
15. Какие бывают методы построения электронной подписи?
16. Что такое коллизия первого рода?
17. Что такое коллизия второго рода?
18. Расскажите об электронной подписи на основе ГОСТ Р 34.10—2012. Чем обеспечивается стойкость алгоритма?

Глава 8

АУТЕНТИФИКАЦИЯ

8.1. ПРОТОКОЛЫ АУТЕНТИФИКАЦИИ

Все протоколы идентификации и аутентификации включают по крайней мере двух абонентов. Обозначим их следующим образом:

- A — доказывающий участник (проходящий идентификацию);
- B — проверяющий участник.

Целью протокола является проверка того, что проверяемый действительно является A (т. е. тем, за кого себя выдает). С точки зрения проверяющего B в ходе работы протокола либо принимается решение об аутентичности доказывающего A, либо протокол завершается без принятия такого решения.

Существуют три большие группы протоколов идентификации и аутентификации:

а) протоколы, *основанные на известной обеим сторонам информации*. Такой информацией может быть пароль, личные идентификационные номера (*personal identification number* — PIN), секретные или открытые ключи, знание которых демонстрируется во время выполнения протокола;

б) протоколы, *использующие физические приборы* (*объекты*), с помощью которых производится идентификация (объекты, идентифицирующие пользователя). Такими приборами могут быть внешние носители (не принадлежащие системе): флеш-накопители, магнитные диски, пластиковые карты и т. д.;

в) протоколы, *использующие физические* (*биометрические*) *параметры*, являющиеся неотъемлемой принадлежностью доказывающего. Здесь используют подписи, отпечатки пальцев, характеристики голоса, геометрию руки и т. д.

Целями идентификации и аутентификации является обеспечение доступа к ресурсам компьютерной (или иной) системы (банковские счета, телекоммуникационные каналы, компьютерные программы, базы данных, сооружения, здания и т. д.).

Протоколы идентификации и аутентификации связаны с протоколами электронной подписи, но проще их. Протоколы электронной подписи определяются содержанием сообщений и включают элементы, обеспечивающие невозможность отказа от подписанного сообщения. В протоколах идентификации и аутентификации содержание сообщения фиксировано (заявление о личности доказывающего A в текущий момент времени).

Рассмотрим структуры данных и протоколы идентификации и аутентификации пользователя. Допустим, что в компьютерной системе зарегистрировано n пользователей. Пусть i-й аутентифицирующий объект i-го пользователя содержит два информационных поля:

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

K_i _ аутентифицирующая информация пользователя, которая может изменяться и служит для аутентификации (для случая парольной аутентификации: пароль $P_i = K_i$).

Описанная структура соответствует практически любому ключевому носителю информации, используемому для опознания пользователя. Например, для носителей типа пластиковых карт выделяется неизменяемая информация ID_i первичной персонализации пользователя и объект в файловой структуре карты, содержащий K_i.

Совокупную информацию в ключевом носителе можно назвать первичной аутентифицирующей информацией i-го пользователя. Очевидно, что внутренний аутентифицирующий объект не должен существовать в системе длительное время (больше времени работы конкретного пользователя). Для длительного хранения следует использовать данные в защищенной форме.

Рассмотрим две типовые схемы идентификации и аутентификации.

Схема 1. В компьютерной системе выделяется объект-эталон для идентификации и аутентификации пользователей. Пусть $E_i = F(ID_i, K_i)$, где F — функция, которая обладает свойством «невосстановимости» (однонаправленности) значения K_i по E_i и ID_i. «Невосстановимость» K_i оценивается некоторой пороговой трудоемкостью T_0 решения задачи восстановления аутентифицирующей информации K_i по E_i и ID_i. Кроме того, для пары K_i и K_j возможно совпадение соответствующих значений E. В связи с этим

вероятность ложной аутентификации пользователя не должна быть больше некоторого порогового значения P_0. на практике задают $T_0 = 10^{20} \ldots 10^{30}$, $P_0 = 10^{-7} \ldots 10^{-9}$.

Протокол идентификации и аутентификации (для схемы 1):

а) пользователь предъявляет свой идентификатор *ID*;

б) если *ID* не совпадает ни с одним ID_i, зарегистрированным в компьютерной системе, то идентификация отвергается (пользователь не допускается к работе), иначе (существует $ID_i = ID$) устанавливается, что пользователь, назвавшийся пользователем *i*, прошел идентификацию;

в) субъект (подсистема) аутентификации запрашивает у пользователя его аутентификатор K (пароль);

г) субъект (подсистема) аутентификации вычисляет значение $Y = F(ID_i, K)$;

д) субъект (подсистема) аутентификации производит сравнение значений Y и E_i. При совпадении этих значений устанавливается, что данный пользователь успешно аутентифицирован в системе. Информация об этом пользователе передается в программные модули, использующие ключи пользователей (т. е. в систему шифрования, разграничения доступа и т. д.). В противном случае аутентификация отвергается, и пользователь не допускается к работе.

Данная схема идентификации и аутентификации пользователя может быть модифицирована. Модифицированная схема (схема 2) обладает лучшими характеристиками по сравнению со схемой 1.

Схема 2. В компьютерной системе выделяется модифицированный объект-эталон. В отличие от схемы 1, в схеме 2 значение E_i равно $F(S_i, K_i)$, где S_i _ случайный вектор, задаваемый при создании идентификатора пользователя, т. е. при создании строки, необходимой для идентификации и аутентификации пользователя; F — функция, которая обладает свойством «невосстановимости» значения K_i по E_i и S_i.

Протокол идентификации и аутентификации (для схемы 2):

а) пользователь предъявляет свой идентификатор *ID*;

б) если *ID* не совпадает ни с одним ID_i, зарегистрированным в компьютерной системе, то идентификация отвергается (пользователь не допускается к работе), иначе (существует $ID_i = ID$) устанавливается, что пользователь, называвшийся пользователем *i*, прошел идентификацию;

в) по идентификатору ID_i выделяется вектор S_i;

г) субъект (подсистема) аутентификации запрашивает у пользователя аутентификатор K;

д) субъект (подсистема) аутентификации вычисляет значение $Y = F(S_i, K)$;

е) субъект (подсистема) аутентификации производит сравнение значений Y и E_i. При совпадении этих значений устанавливается, что данный пользователь успешно аутентифицирован в системе. В противном случае аутентификация отвергается, т. е. пользователь не допускается к работе.

Следует отметить, что необходимым требованием устойчивости схем аутентификации к восстановлению информации K_i является случайный равновероятный выбор K_i из множества возможных значений.

8.2. ОДНОСТОРОННЯЯ АУТЕНТИФИКАЦИЯ

Статическая аутентификация. Простой пароль. Аутентификация заключается в проверке: является ли подключающийся субъект тем, за кого он себя выдает. Если в процессе аутентификации подлинность субъекта установлена, то система защиты информации должна определить его полномочия (совокупность прав). Это необходимо для последующего контроля и разграничения доступа к ресурсам. В общем случае этот термин может относиться ко всем аспектам информационного взаимодействия: сеансу связи, сторонам, участвующим во взаимодействии, передаваемым сообщениям и т. д.

Установление подлинности (т. е. проверка и подтверждение) всех аспектов информационного взаимодействия является важной составной частью проблемы обеспечения достоверности получаемой информации.

Статическая аутентификация основана на использовании простого пароля. Пароль формируется пользователем либо администратором системы на основании ограничений на содержание пароля по алфавиту, сложности, длине, времени действия.

Достоинством статической процедуры аутентификации на основе простого пароля является ее простота. Для того чтобы получить доступ к ресурсу, пользователь представляет свой идентификатор и пароль и прямо или косвенно определяет необходимый ресурс. При этом идентификатор пользователя выступает как заявка на идентификацию, а пароль — как подтверждение этой заявки.

Недостатком такой процедуры аутентификации является ее *нестойкость к атакам подбора пароля*. Для того чтобы увеличить,

повысить стойкость паролей и вместе с тем не нарушить такое их важное качество, как возможность запоминания человеком, часто используются парольные фразы. В этом случае в качестве пароля используется целое предложение вместо короткого слова. Парольные фразы хешируются до фиксированной длины и играют ту же роль, что и обычные пароли.

Широкое использование такой процедуры также связано с простотой изменения пароля. Пароль может быть легко изменен при наличии подозрения его компрометации. Во многих информационных системах регулярное изменение паролей является обязательным требованием.

Устойчивая аутентификация. Динамический пароль. Этот класс аутентификации использует динамические данные аутентификации, меняющиеся с каждым сеансом аутентификации. Часто такая аутентификация используется как дополнение к статической аутентификации.

Атакующий, который может перехватить информацию, передаваемую между аутентифицируемым и аутентифицирующим, может попытаться инициировать новый сеанс аутентификации с аутентифицирующим и повторить записанные им данные аутентификации в надежде замаскироваться под легального пользователя. Однако устойчивая аутентификация защищает от таких атак, так как данные аутентификации, записанные в ходе предыдущего сеанса аутентификации, не смогут быть использованы для аутентификации в последующих сеансах.

Тем не менее устойчивая аутентификация не защищает от активных атак, в ходе которых атакующий может изменить данные или команды, передаваемые пользователем серверу после аутентификации. Так как сервер связывает на время сеанса данного аутентифицировавшегося пользователя с данным логическим соединением, он полагает, что именно он является источником всех принятых им команд по этому соединению.

Традиционные пароли не смогут обеспечить устойчивую аутентификацию, так как пароль пользователя можно перехватить и использовать в дальнейшем, а одноразовые пароли и электронные подписи могут обеспечить такой уровень защиты.

PIN-код. Разновидностью фиксированных паролей являются ***PIN-коды*** *— числовые пароли длиной от 4 до 8 десятичных цифр.* Чаще всего они используются в соединении с методом «обладания чем-либо»: обычно микропроцессорной пластиковой картой или картой с магнитной полосой. PIN-код обеспечивает второй уровень защиты на случай, если карта потеряна или украдена. Для защиты от полного перебора такого маленького ключевого пространства необходимы дополнительные меры: организационная и физическая защита. Например, банкомат может забрать у пользователя пластиковую карту или блокировать ее после нескольких подряд неудачных попыток ввода пароля.

Биометрические средства идентификации. Биометрическая идентификация — это предъявление пользователем своего уникального биометрического параметра и процесс сравнения его со всей базой имеющихся подобных данных. Для извлечения такого рода персональных данных используются биометрические считыватели.

Биометрические системы контроля доступа удобны для пользователей тем, что носители информации находятся всегда при них, не могут быть утеряны либо украдены. Биометрический контроль доступа считается более надежным, так как идентификаторы не могут быть переданы третьим лицам, скопированы.

Различают следующие ***методы биометрической идентификации*:**

1) *статические*, основанные на физиологических признаках человека, присутствующих с ним на протяжении всей его жизни:

- по отпечатку пальца;
- по лицу;
- по радужной оболочке глаза;
- по геометрии руки;
- по термограмме лица;
- по ДНК;
- на основе акустических характеристик уха;
- по рисунку вен;

) *динамические*, которые берут за основу поведенческие характеристики людей, а именно: подсознательные движения в процессе повторения какого-либо обыденного действия (почерк, голос, походка). Например:

- по голосу;
- рукописному почерку;
- клавиатурному почерку и другие.

Существуют также комбинированные системы идентификации, использующие несколько биометрических характеристик, что позволяет удовлетворить самые строгие требования к надежности и безопасности систем контроля доступа.

Для определения *эффективности средств контроля* и управления доступом на основе биометрической идентификации используют следующие *показатели*:

- FAR — коэффициент ложного пропуска;
- FMR — вероятность, что система неверно сравнивает входной образец с несоответствующим шаблоном в базе данных;
- FRR — коэффициент ложного отказа;
- FNMR — вероятность того, что система ошибется в определении совпадений между входным образцом и соответствующим шаблоном из базы данных;
- график ROC — визуализация компромисса между характеристиками FAR и FRR;
- коэффициент отказа в регистрации (FTE или FER) — коэффициент безуспешных попыток создать шаблон из входных данных (при низком качестве последних);
- коэффициент ошибочного удержания (FTC) — вероятность того, что автоматизированная система не способна определить биометрические входные данные, когда они представлены корректно;
- емкость шаблона — максимальное количество наборов данных, которые могут храниться в системе.

В России использование биометрических данных регулируются Статьей 11 Федерального закона № 152-ФЗ от 27 июля 2006 г. «О персональных данных».

Электронные ключи и карты. Электронные идентификаторы (электронные ключи и карты) представляют собой различного вида электронные изделия, предназначенные для выполнения идентификации и аутентификации пользователя при доступе к различным информационным и физическим ресурсам. Они позволяют заменить привычную схему авторизации, когда пользователю для получения доступа необходимо ввести имя, пароль или набрать код доступа на клавиатуре, на ее более защищенный вариант — авторизацию с помощью электронного ключа или смарт-карты.

В этом случае исключается как минимум один фактор компрометации ключа доступа — визуальный, т. е. когда злоумышленник может подсмотреть вводимые пользователем данные, необходимые для его авторизации.

Кроме того, пароли, хранимые в памяти электронного ключа, можно делать сколь угодно сложными — большой длины, состоящих из несвязанного логически набора букв и цифр, которые просто невозможно запомнить.

Возможность хранения в электронном идентификаторе не только паролей, но и ключей доступа, ключевых контейнеров, электронных сертификатов и профилей пользователя, еще больше расширяет границы их применения.

В настоящее время электронные ключи и смарт-карты широко используются для строгой аутентификации пользователя при доступе к компьютеру, в сеть предприятия, к интернет-сайтам, электронной подписи в различных системах электронного документооборота, сдачи отчетности, доступа в помещения и на охраняемую территорию, контроля охраны периметра, доступа к управлению транспортным средством, оплаты услуг и товаров.

Выделяют следующие электронные идентификаторы:

- *электронные USB-ключи* и *смарт-карты eToken* — компактные устройства персонального использования, выполненные в виде пластиковой карты, USB-ключа или брелока и предназначенные для защищенного хранения данных, необходимых для авторизации пользователя при доступе к защищенным ресурсам;
- *смарт-карты и USB-ключи JaCarta PKI* — новое поколение электронных идентификаторов, предназначенных для выполнения строгой двух- и трехфакторной аутентификации пользователей при доступе к корпоративной информации и персональным данным, хранящимся и обрабатываемым на рабочих станциях, персональных компьютерах, ноутбуках, корпоративных и web-серверах. Выпускаются в виде смарт-карт, USB-токенов, SD-карт и различных комбинированных решений;
- *электронные ключи iButton* (Dallas Touch Memory) — представляют собой электронные идентификаторы в виде стального герметичного цилиндрического изделия, имеющего уникальный регистрационный номер (*ID*). Выпускаются модели, как только с *ID*, так и модели с дополнительно встроенной памятью. Идентификаторы iButton широко используются в охранных системах, системах контроля доступа, при идентификации пользователя на компьютере, в различных системах контроля и измерений;
- *бесконтактные проксимити-идентификаторы* (Proximity-идентификаторы) — выпускаются в виде бесконтактных пластиковых карт, брелоков, браслетов, компактных радиометок и предназначены для бесконтактной радиочастотной (RFID) идентификации пользователей при доступе в помещение, контроля и допуска автомобильного и железнодорожного транспорта, расчета в транспорте и развлекательных комплексах, учета на складах и производстве.

Токены. Электронные ключи eToken представляют собой защищенные устройства, предназначенные для безопасного хранения секретных данных, необходимых для авторизации пользователей при доступе к различным информационным ресурсам.

В защищенной ПИН-кодом (PIN) памяти eToken хранятся пользовательские пароли, профили, электронные сертификаты и ключи, с помощью которых обеспечивается безопасный доступ к компьютерам и в сеть предприятия, интернет-сайтам, почтовым программам, рабочим приложениям и данным на дисках, а также осуществляется цифровая подпись электронных документов для обеспечения их юридической значимости.

Ключи eToken выпускается в различных форм-факторах, наиболее востребованными из которых являются USB-ключи, смарт-карты, брелоки, не требующие подключения к каким-либо устройствам, а также комбинированные электронные идентификаторы.

В настоящее время наиболее популярны следующие виды ключей eToken:

- USB-ключ eToken PRO (Java) 72K — персональное средство аутентификации и защищенного хранения данных, необходимых для авторизации пользователей, аппаратно поддерживающее работу с цифровыми сертификатами и электронно-цифровой подписью;
- USB-ключ eToken NG-FLASH (Java) 72K — комбинированное изделие, сочетающее в себе возможности ключа eToken PRO (Java) 72K и дополнительного модуля Flash-памяти (флешки) объемом до 16 Гб. Используется для авторизации пользователей и хранения различной информации в Flas*h*-модуле;
- USB-ключ eToken NG-OTP (Java) 72K — состоит из двух независимых изделий: электронного ключа eToken PRO (Java) 72K и генератора одноразовых паролей eToken PASS. Применяется для авторизации пользователей с помощью цифровых сертификатов, ключей доступа, обычных и одноразовых паролей;
- брелок eToken PASS — автономный генератор одноразовых паролей, не требующий подключения к компьютеру или установки программного обеспечения на клиентской машине. Используется в любых операционных системах и при доступе к защищенным ресурсам с мобильных устройств и терминалов.

8.3. ВЗАИМНАЯ АУТЕНТИФИКАЦИЯ

Метод «запрос — ответ». Взаимная аутентификация — вариант аутентификации сторон, при котором каждая из сторон проверяет, что взаимодействующая с ней сторона — именно та, за которую себя выдает. Взаимная аутентификация реализуется таким протоколом идентификации, в котором каждый из участников является одновременно и доказывающим, и проверяющим. Это позволяет за один сеанс выполнения протокола каждым из участников доказать другому участнику свою идентичность.

Идея, заложенная в основе протоколов аутентификации типа «запрос — ответ», заключается в том, что одна сторона (претендент) доказывает свою идентичность другой стороне (проверяющему) посредством демонстрации знания секрета проверяющему (в некоторых протоколах секрет известен проверяющему и используется для проверки ответа, в других — вообще нет необходимости, чтобы секрет был известен проверяющему).

Претендент должен ответить на запрос, меняющийся во времени, причем ответ должен зависеть и от его секрета, и от запроса. Запрос — это обычно число, выбираемое одной стороной в начале протокола. Если линия связи между участниками протокола прослушивается злоумышленником, ответ претендента не должен снабжать злоумышленника полезной информацией, которая могла бы быть использована в последующих сеансах протокола, так как последующие запросы проверяющего изменятся.

Параметры, меняющиеся во времени, используются для противодействия атакам на протокол методами повтора сеанса и включения в канал. Они обеспечивают гарантии уникальности и актуальности каждого сеанса протокола. В криптографических протоколах широко используется такое понятие, как свежесть (freshness) той или иной величины. Оно означает, что значение этой величины было сгенерировано в начале или в ходе выполнения текущего сеанса протокола.

В качестве таких параметров могут использоваться три типа величин, изменяющихся от сеанса к сеансу: это случайные числа, числовые последовательности и метки времени. При этом важно, чтобы была гарантирована целостность этих одноразовых величин, что обычно реализуется путем криптографической привязки их к другим данным, используемым в протоколе.

Наиболее просто реализуется выработка *числовых последовательностей* (члены последовательности обычно обозначают символом n с соответствующими индексами). В простейшем случае можно применить последовательность номеров сеансов выполняемого протокола с увеличением на единицу в каждом последующем сеансе. Недостатком этого способа является необходимость ведения участниками своих внутренних счетчиков последовательностей чисел.

Случайные числа (обычно обозначают символом r с соответствующими индексами) легко генерировать всем сторонам про-

токола, при этом от участников не требуется поддержания какой-либо дополнительной информации, как в предыдущем случае.

В третьем случае, когда используются *метки времени* (*timestamps*), сторона, генерирующая сообщение в протоколе, получает метку времени со своих локальных часов (системного таймера) и криптографически привязывает ее к сообщению. Получатель берет текущее время со своих локальных часов (системного таймера) и сравнивает с величиной метки времени, полученной от партнера. Для использования этого метода необходимо выполнение следующих условий:

- разница между метками времени отправителя и получателя должна укладываться в определенный интервал времени фиксированного размера — окно принятия (acceptance window);
- ранее не должно было быть получено сообщений с идентичной меткой времени от того же отправителя. Это подтверждается проверкой по списку всех меток времени, полученных от каждого источника за период текущего окна принятия;
- часы должны быть синхронизированы и защищены от модификации.

Преимущество метода в том, что он не требует поддержания внутренней информации (ни долговременной, ни кратковременной) и наличия каких-либо дополнительных устройств, так как часы (системный таймер) есть практически на любой вычислительной платформе.

Процедура «рукопожатие». Для взаимной проверки подлинности, помимо метода «запрос — ответ», часто используют процедуру «рукопожатие». Эта процедура заключается во взаимной проверке ключей, используемых сторонами. Иначе говоря, стороны признают друг друга законными партнерами, если докажут друг другу, что обладают правильными ключами. Процедуру «рукопожатие» обычно применяют в компьютерных сетях при организации сеанса связи между пользователями, пользователем и хост-компьютером, между хост-компьютерами и т. д.

Рассмотрим в качестве примера процедуру «рукопожатие» для двух пользователей *A* и *B*. Пусть применяется симметричная криптосистема, а пользователи *A* и *B* разделяют один и тот же секретный ключ K_{AB}. Протокол (процедура) «рукопожатия» состоит из следующих шагов:

а) пусть пользователь *A* инициирует процедуру рукопожатия, отправляя пользователю *B* свой идентификатор *id* (*A*) в открытой форме;

б) пользователь *B*, получив идентификатор *id* (*A*), находит (в некоей своей базе данных) общий с *A* секретный ключ K и вводит его в свою криптосистему;

в) тем временем пользователь *A* генерирует случайную последовательность *S* и отправляет ее пользователю *B в* зашифрованном виде;

г) пользователь *B* расшифровывает эту криптограмму и раскрывает исходный вид последовательности *S*;

д) оба пользователя *A* и *B* преобразуют последовательность *S*, используя открытую одностороннюю хеш-функцию $\alpha(S)$, т. е. *A* и *B* вычисляют $\alpha(S)$;

е) пользователь *B* шифрует сообщение $\alpha(S)$ и отправляет результат шифрования пользователю *A*;

ж) пользователь *A* расшифровывает это зашифрованное сообщение и сравнивает полученное сообщение $\alpha'(S)$ с исходным (самим вычисленным) значением $\alpha(S)$. Если эти значения равны, то пользователь *A* признает подлинность пользователя B;

з) пользователь *A* шифрует сообщение $\alpha(S)$ и отправляет результат шифрования пользователю *B*;

и) пользователь *B* расшифровывает это зашифрованное сообщение и сравнивает полученное сообщение $\alpha'(S)$ с исходным (самим вычисленным) значением $\alpha(S)$. Если эти значения равны, то пользователь *B* признает подлинность пользователя A.

Шаги е)—ж) и з)—и) образуют две процедуры проверки одной стороной другой стороны. Обе эти процедуры образуют единую процедуру «рукопожатия», которая обычно выполняется в самом начале любого сеанса связи между любыми двумя сторонами в компьютерных сетях.

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

8.4. НАЗНАЧЕНИЕ И СТРУКТУРА PKI

Рассмотренные метод взаимной аутентификации «запрос — ответ» и процедура «рукопожатие» используются в том случае, когда абоненты доверяют друг другу. В случае, когда абоненты не доверяют друг другу, использование общего секретного ключа невозможно. Для взаимодействия абонентов, не доверяющих друг другу, используют аутентификацию на основе открытого и закрытого (секретного) ключей. Для реализации такой аутентификации

разработана инфраструктура открытых ключей (PKI — public key infrastruction).

Инфраструктура открытых ключей — это инфраструктура безопасности для распространения открытых ключей, управления электронными сертификатами и ключами пользователей.

В основе PKI лежит использование криптографической системы с открытым ключом и несколько основных принципов:

- закрытый ключ известен только его владельцу;
- удостоверяющий центр (УЦ) создает сертификат открытого ключа, таким образом удостоверяя этот ключ;
- никто не доверяет друг другу, но все доверяют УЦ;
- УЦ подтверждает или опровергает принадлежность открытого ключа заданному лицу, которое владеет соответствующим закрытым ключом.

Фактически, PKI представляет собой систему, основным компонентом которой является удостоверяющий центр и пользователи, взаимодействующие между собой посредством УЦ.

Система PKI реализуется в модели «клиент — сервер», т. е. проверка какой-либо информации, предоставляемой инфраструктурой, может происходить только по инициативе клиента.

Перечислим *основные компоненты PKI*.

1. УЦ является основной структурой, формирующей цифровые сертификаты подчиненных центров сертификации и конечных пользователей. УЦ является главным управляющим компонентом PKI:
   - является доверенной третьей стороной;
   - осуществляет управление сертификатами.
2. Сертификат открытого ключа (чаще всего просто сертификат) — это данные пользователя и его открытый ключ, скрепленные подписью УЦ. Выпуская сертификат открытого ключа, УЦ тем самым подтверждает, что лицо, поименованное в сертификате, владеет секретным ключом, который соответствует этому открытому ключу.
3. Регистрационный центр (РЦ) — необязательный компонент системы, предназначенный для регистрации пользователей. Для этих целей РЦ обычно предоставляет web-интерфейс. УЦ доверяет РЦ проверку информации о субъекте. РЦ, проверив правильность информации, подписывает ее своим ключом и передает УЦ, который, проверив ключ регистрационного центра, выписывает сертификат. Один РЦ может работать с несколькими УЦ (т. е. состоять в нескольких PKI), один УЦ может работать с несколькими регистрационными центрами. Иногда УЦ выполняет функции РЦ.

4. Репозиторий — хранилище, содержащее сертификаты и списки отозванных сертификатов (СОС) и служащее для распространения этих объектов среди пользователей. В Законе РФ «Об электронной подписи» он называется реестр сертификатов ключей подписей.
5. Архив сертификатов — хранилище всех изданных когда-либо сертификатов (включая сертификаты с закончившимся сроком действия). Архив используется для проверки подлинности электронной подписи, которой заверялись документы.
6. Центр запросов — необязательный компонент системы, где конечные пользователи могут запросить или отозвать сертификат.
7. Конечные пользователи — пользователи, приложения или системы, являющиеся владельцами сертификата и использующие инфраструктуру управления открытыми ключами.

8.5. СЕРТИФИКАТЫ ОТКРЫТОГО КЛЮЧА

При использовании открытых ключей возникает проблема установления их подлинности. Наиболее распространенный подход к решению этой задачи — сертификация открытых ключей.

Сертификат — это электронный или бумажный документ, содержащий открытый ключ, информацию о владельце ключа, области применения ключа, подписанный выдавшим его УЦ и подтверждающий принадлежность открытого ключа владельцу.

С другой стороны, сертификат — это структура данных, применяющаяся для связывания определенного модуля с определенным открытым ключом. Сертификаты используются для подтверждения подлинности пользователей, приложений и сервисов и для контроля доступа (авторизации).

Сертификация заключается в том, что некоторый УЦ, которому доверяют все пользователи, гарантирует подлинность открытых ключей. Для этого УЦ для каждого открытого ключа выпускает сертификат. Сертификат открытого ключа состоит из следующих полей:

- сам открытый ключ владельца сертификата;
- срок действия;
- имя эмитента (центра сертификации);
- имя владельца сертификата и самой важной части электронной подписи.

Электронная подпись гарантирует невозможность подделки сертификата. Она является результатом криптографической хеш-

функции от данных сертификата, зашифрованным закрытым ключом УЦ. Открытый ключ УЦ является общеизвестным, поэтому любой может расшифровать им электронную подпись сертификата, затем вычислить хеш самостоятельно и сравнить, совпадают ли хеши. Если хеши совпадают, значит, сертификат действительный и можно не сомневаться, что открытый ключ принадлежит именно тому, с кем будет установлено соединение.

Открытый ключ может быть использован для организации защищенного канала связи с владельцем двумя способами:

- для проверки подписи владельца (аутентификация);
- шифрования посылаемых ему данных (конфиденциальность).

8.6. СТАНДАРТ X.509

Существует довольно большое количество форматов сертификатов, но наибольшее распространение в настоящее время получил формат X.509 v3. Эта базовая технология, используемая в инфраструктуре открытых ключей операционной системы Windows. Все сертификаты X.509 согласованы с международным стандартом ITU-T X.509.Таким образом (теоретически), сертификат X.509, созданный для одного приложения, может быть использован в любом другом, поддерживающим этот стандарт. на деле, однако, выясняется, что разные компании создали собственные расширения X.509, многие из которых друг с другом никак не совместимы.

Сертификат X.509 — это набор стандартных полей, содержащих сведения о пользователе или устройстве, и их соответствующий открытый ключ. Стандарт X.509 определяет, какие сведения входят в сертификат и как они кодируются (формат данных).

Сертификат X.509 содержит следующие сведения:

- версия X.509 — указывает, на основе какой версии стандарта X.509 построен данный сертификат, что определяет, какая информация может в нем содержаться;
- открытый ключ обладателя сертификата — открытый ключ совместно с идентификатором используемого алгоритма (указывающим криптосистему, к которой принадлежит данный ключ) и прочая аналогичная информация о параметрах ключа;
- серийный номер сертификата — сообщество (программа или лицо), создавшее сертификат, обязано снабдить его уникальным серийным номером для его выделения среди прочих сертификатов, выданных данным сообществом. Эта информация применяется в ряде случаев; например, при отзыве сертификата его серийный номер помещается в реестр аннулированных сертификатов;
- уникальный опознаватель обладателя ключа (*distinguished name*, DN — уникальное имя) — это имя должно быть уникальным и единственным во всем Интернете. DN состоит из нескольких подсекций и может выглядеть примерно так:

CN=Bob Davis, EMAIL=bdavis@pgp.com, OU=PGP Security Division, O=Network Associates, Inc., C=US

(что обозначает имя субъекта, электронную почту, подразделение организации, организацию и страну соответственно);

- срок действия сертификата — дата / время начала действия сертификата и дата / время окончания его действия; указывает на то, когда сертификат станет просрочен;
- уникальное имя выдавшего сертификат — уникальное имя сообщества, подписавшего сертификат. Обычно это наименование УЦ. Использование сертификата подразумевает доверие сообществу, его подписавшему;
- ЭП выдавшего сертификат — электронная подпись, сообщества, выдавшего сертификат;
- идентификатор алгоритма подписи — указывает алгоритм, использованный УЦ для подписания сертификата.

8.7. УДОСТОВЕРЯЮЩИЕ ЦЕНТРЫ

Понятие «Удостоверяющий центр» регламентируется Федеральным законом от 6 апреля 2011 г. № 63-ФЗ «Об электронной подписи». В соответствии с данным законом удостоверяющий центр — юридическое лицо, индивидуальный предприниматель либо государственный орган или орган местного самоуправления, осуществляющие функции по созданию и выдаче сертификатов ключей проверки электронных подписей, а также иные функции, предусмотренные указанным Федеральным законом.

Основными задачами УЦ являются:

1) изготовление сертификата ключа проверки электронной подписи. УЦ выдает такие сертификаты лицам, обратившимся за их получением (заявителям);

2) установление сроков действия сертификатов ключей проверки электронных подписей;

3) аннулирование выданных этим УЦ сертификаты ключей проверки электронных подписей;

4) выдача по обращению заявителя средства электронной подписи, содержащие ключ электронной подписи и ключ проверки электронной подписи или выдача средств электронной подписи, обеспечивающие возможность создания ключа электронной подписи и ключа проверки электронной подписи заявителем;

5) ведение реестра выданных и аннулированных этим удостоверяющим центром сертификатов ключей, в т.ч. включающего в себя информацию о сертификатах ключей проверки ЭП, и информацию о датах прекращения действия или аннулирования сертификатов и об основаниях таких прекращений или аннулирований;

6) установление порядка ведения реестра сертификатов, не являющихся квалифицированными, и порядка доступа к нему, а также обеспечение доступа лиц к информации, содержащейся в реестре сертификатов, в т.ч. и через интернет;

7) создание по обращениям заявителей ключей электронных подписей и ключей проверки электронных подписей;

8) проверка уникальности ключей проверки электронных подписей в реестре сертификатов;

9) осуществление по обращениям участников электронного взаимодействия проверки электронных подписей;

10) осуществление иной связанной с использованием электронной подписи деятельности.

Выпуская сертификат открытого ключа некоторого пользователя, УЦ подтверждает и берет на себя ответственность за то, что он установил принадлежность данного открытого ключа именно указанному в сертификате пользователю.

В подлинности сертификата и, как следствие, в подлинности открытого ключа пользователи убеждаются, проверяя подпись УЦ. Для такой проверки необходим открытый ключ подписи самого центра. При этом сертификат открытого ключа УЦ выпускает сам УЦ — это так называемый корневой сертификат. Корневой сертификат передается каждому пользователю, доверяющему УЦ, надежным и защищенным способом, исключающим возможность подмены сертификата. Доверие к сертификатам открытых ключей пользователей полностью основывается на доверии к корневому сертификату УЦ.

Деятельность УЦ сводится не только к выпуску сертификатов. Сертификат открытого ключа всегда выпускается на определенный срок, после которого открытый ключ считается недействительным. В реальности же срок действия открытого ключа может быть уменьшен, а сертификат отозван, например, по причине компрометации или утери парного закрытого ключа. Отозванные сертификаты помещаются в специальный общедоступный список — список отозванных сертификатов. Любой пользователь должен иметь возможность проверить текущий статус используемого им сертификата, обратившись к этому списку. Также любой владелец сертификата должен иметь возможность изменить его статус по своему желанию: приостановить, возобновить или отозвать. Открытый ключ может использоваться и после формального прекращения своего действия. Таким образом, УЦ должен сопровождать весь жизненный цикл выпускаемых им сертификатов открытых ключей, включая поддержку в актуальном состоянии информации о статусе сертификатов.

8.8. АРХИТЕКТУРА PKI

В настоящее время выделяют 5 основных видов архитектур PKI:

1) простая PKI (одиночный удостоверяющий центр);
2) иерархическая PKI;
3) сетевая PKI;
4) кросс-сертифицированные корпоративные PKI;
5) архитектура мостового УЦ.

Любой тип архитектуры PKI имеет свои слабые и сильные стороны. Выбор оптимальной архитектуры осуществляется с учетом специфики деятельности, потребностей и возможностей организации.

Одиночный УЦ: все сертификаты выпускаются одним УЦ; исчезают проблемы, связанные с построением и проверкой пути сертификации. Компрометация одиночного УЦ является катастрофой и делает недействительными все выпущенные им сертификаты. В результате сообщество полностью теряет доступ к сервисам безопасности. С другой стороны, такое сплоченное сообщество способно эффективно распространить уведомление о компрометации и быстро восстановить PKI.

Иерархическая PKI соответствует структуре организации, поэтому центр определяется естественным образом. Построение и проверка пути сертификации — просты. Компрометация головного УЦ выводит из строя всю инфраструктуру, но восстановить работу PKI после компрометации любого другого УЦ достаточно просто. Правильно выбранные политика, регламент функционирования и средства физической защиты УЦ могут сделать его компрометацию маловероятной. Различают модель строгой иерархии

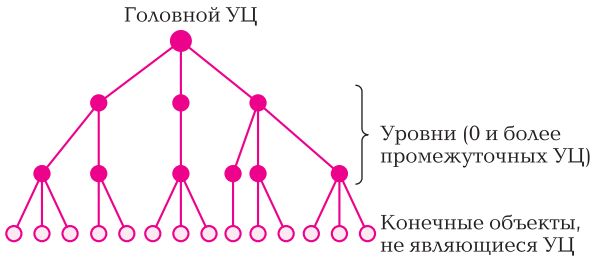

Рис. 8.1. Строгая иерархия удостоверяющих центров

удостоверяющих центров и нестрогую иерархию удостоверяющих центров.

Модель строгой иерархии УЦ обычно представляется в виде древовидной структуры, где корень находится наверху, а ветви спускаются вниз и заканчиваются листами. В таком дереве корень представляет определенный УЦ, который является корневым (головным) и действует как главный пункт доверия целого домена подчиненных ему субъектов PKI.

Промежуточные УЦ располагаются под корнем и представлены древовидными узлами. От них, в свою очередь, отходят следующие ветви. Листья, конечные вершины дерева, — это субъекты PKI, которые не являются УЦ. Также их называют конечными пользователями, они представлены на рис. 8.1.

Корень является начальным пунктом доверия. У всех субъектов PKI есть открытый ключ корневого УЦ, и все субъекты полагаются на него как на начальный и конечный пункт доверия при проверке всех сертификатов. Помимо этого, если в конфигурации PKI отсутствуют промежуточные удостоверяющие центры, или она выглядит иначе, — этот ключ также является корневым. В некоторых иерархиях УЦ верхнего уровня может сертифицировать не только удостоверяющие центры, но и конечные субъекты.

В модели строгой иерархии удостоверяющих центров все субъекты иерархии доверяют одному головному УЦ. Иерархия строится следующим образом:

- создается головной УЦ, который генерирует и подписывает сертификат для самого себя. Сертификат головного УЦ называют самоизданным (или самоподписанным) корневым сертификатом; именно он составляет основу доверия всех субъектов данной строгой иерархии;
- головной УЦ сертифицирует, т. е. выпускает и подписывает сертификаты для удостоверяющих центров, непосредственно ему подчиненных. Количество подчиненных удостоверяющих центров может быть и нулевым (подчиненные центры отсутствуют);
- каждый из этих удостоверяющих центров сертифицирует другие удостоверяющие центры, непосредственно ему подчиненные;
- на уровнях иерархии от второго до последнего удостоверяющие центры сертифицируют конечных субъектов.

Каждый субъект иерархии должен обладать копией открытого ключа головного УЦ. От процесса инсталляции открытого ключа зависит обработка сертификатов для всей цепочки связей в этой модели, поэтому он должен быть защищен надлежащим образом. Ключ может быть получен субъектом по физическому каналу (например, обычная почтовая или телефонная связь) либо доставлен электронным способом и подтвержден через внешний механизм (например, хеш-код ключа может быть отправлен по почте, сообщен по телефону). Также в многоуровневой строгой иерархии конечные субъекты сертифицируются находящимся непосредственно над ними УЦ, но головной УЦ является главным пунктом доверия. В тех иерархиях, где отсутствуют подчиненные удостоверяющие центры, головной УЦ одновременно является издателем сертификатов всех конечных субъектов.

Не для всех областей PKI подходит модель строгой иерархии. Поэтому в остальных случаях используется модель нестрогой иерархии удостоверяющих центров, которая позволяет строить доверенный путь доверяющим сторонам, сертифицированным одним УЦ, не вовлекая в этот процесс головной УЦ и любые вышестоящие УЦ. Таким образом, при проверке сертификатов друг друга доверяющие стороны могут полагаться на общий для них УЦ.

Если сертификаты пользователей *A* и *B* выпущены одним и тем же УЦ2, то пользователи могут проверять сертификаты друг друга без построения пути сертификации к головному УЦ строгой иерархии тогда и только тогда, когда пользователи относятся к иерархии одного и того же доверенного издателя. Однако если сертификат пользователя *C* издан не УЦ2, а другим УЦ, то для проверки сертификата *C* пользователи *A* и *B* должны строить полный путь сертификации через головной УЦ, заверивший сертификат пользователя *C*. Таким образом, если доверяющая сторона оценивает сертификат, выпущенный сторонним УЦ, то должна выполнять валидацию пути сертификации в соответствии с требованиями строгой иерархии.

Сетевая PKI — это решение для тех организаций, которые не имеют четкой иерархической структуры. В таких организациях проще развернуть сетевую PKI, поскольку взаимодействующие

стороны предпочитают устанавливать равноправные, а не подчиненные отношения доверия. В сетевой конфигурации все головные удостоверяющие центры потенциально кросс-сертифицированы друг с другом. Два головных удостоверяющих центра устанавливают отношения кросс-сертификации, если их сообществам необходимо иметь защищенные коммуникации друг с другом. В полностью связанном случае, иногда называемом полной сетью, кросс-сертифицированные отношения устанавливаются для всех головных удостоверяющих центров, хотя на практике чаще встречаются неполные сети.

Кросс-сертификация — это механизм связывания вместе удостоверяющих центров, ранее не имевших связей друг с другом, таким образом, что становятся возможными защищенные коммуникации между соответствующими сообществами субъектов. Фактически механизм кросс-сертификации аналогичен механизму обычной сертификации, за исключением того, что и субъект, и издатель кросс-сертификата являются удостоверяющими центрами, в то время как субъектом обычного сертификата является конечный субъект.

Сетевая PKI — лучший вариант, если организация нуждается в инфраструктуре, способной легко преодолевать последствия компрометации любого УЦ. Компрометация одного УЦ будет катастрофой только для его пользователей, но серьезно не отразится на транзакциях между пользователями других удостоверяющих центров сетевой PKI. Главные недостатки сетевой архитектуры PKI — сложность построения и проверка пути сертификации.

Мостовые удостоверяющие центры — эффективное решение для связывания большого числа разнородных корпоративных PKI. Эта модель наиболее соответствует динамичным деловым отношениям, когда компании работают с ограниченным набором деловых партнеров, но связи между ними быстро устанавливаются и легко разрываются. Отношения между компаниями одной отрасли меняются не столь быстро. Мостовой УЦ позволяет установить отдельную связь с каждой корпоративной PKI. Обеспечивая мост доверия со всеми корпоративными PKI, даже если они никогда не работали вместе, мостовой УЦ поддерживает динамичные деловые связи, необходимые современной экономике.

ВОПРОСЫ И ЗАДАНИЯ ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ ЗНАНИЙ

1. Какие существуют виды протоколов идентификации и аутентификации?
2. Чем отличаются протоколы электронной подписи от протоколов идентификации и аутентификации?
3. Опишите схему 1 идентификации и аутентификации.
4. Опишите схему 2 идентификации и аутентификации.
5. Что такое односторонняя аутентификация?
6. Назовите виды односторонней аутентификации.
7. Какая аутентификация называется устойчивой?
8. Когда используется модель строгой иерархии удостоверяющих центров?
9. Когда используется модель нестрогой иерархии удостоверяющих центров?
10. Назовите основные задачи удостоверяющих центров.
11. Что такое сертификат открытого ключа? Для чего он используется?
12. Что содержит в себе сертификат открытого ключа?
13. Что содержит в себе сертификат X.509?
14. Что такое «рукопожатие»?
15. Что такое динамический пароль?
16. Что такое PIN-код?
17. На чем основан статический метод биометрической аутентификации?
18. На чем основан динамический метод биометрической аутентификации?
19. Какие показатели характеризуют биометрическую идентификацию?
20. Чем отличаются токены от электронных ключей?

Глава 9

ЗАЩИТА ИНФОРМАЦИИ В СЕТЯХ ПЕРЕДАЧИ ДАННЫХ

9.1. ЦЕЛОСТНОСТЬ СООБЩЕНИЯ

Одной из задач при передаче сообщений в вычислительных сетях является обеспечение целостности сообщения. Будем понимать под целостностью информации состояние информации, при котором ее изменение осуществляется только преднамеренно субъектами, имеющими на него право. Поскольку информация при общении людей представляется в том числе и в виде сообщений, то определение целостности информации можно применить и к целостности сообщения. В соответствии со сказанным будем понимать под ***целостностью сообщения*** состояние сообщения, при котором его изменение осуществляется только преднамеренно субъектами, имеющими на него право.

Одним из путей обеспечения целостности сообщения является передача пользователем получателю краткого представления передаваемого сообщения. Подобное краткое представление называют ***контрольной суммой*** или ***дайджестом сообщения***. Алгоритмы расчета контрольных сумм разработаны так, чтобы они были по возможности уникальны для каждого сообщения. Таким образом, устраняется возможность подмены одного сообщения другим с сохранением того же самого значения контрольной суммы. Однако при использовании контрольных сумм возникает проблема передачи их получателю.

Одним из возможных путей ее решения является *включение контрольной суммы в электронную подпись*. При помощи электронной подписи получатель может убедиться в том, что полученное им сообщение послано не сторонним лицом, а имеющим определенные права отправителем. Электронные подписи создаются шифрованием контрольной суммы и дополнительной информации с помощью

личного ключа отправителя. Таким образом, кто угодно может расшифровать электронную подпись, используя открытый ключ, но корректно создать подпись может только владелец личного ключа. Для защиты от перехвата и повторного использования электронная подпись включает в себя уникальное число — *порядковый номер*.

9.2. СЛУЧАЙНАЯ МОДЕЛЬ ORACLE

Случайная модель Oracle была предложена в 1993 г. М. Белларом (M. Bellare) и Ф. Роджеем (Ph. Rogaway). Это идеальная математическая модель для хеш-функции. Функция, которая основана на этой модели, обладает следующими свойствами:

- когда поступает новое сообщение любой длины, Oracle порождает и вырабатывает на выходе дайджест сообщения фиксированной длины, которые состоят из случайных строк нулей и единиц. Это oracle-запись сообщения и дайджест сообщения;
- когда передается сообщение, для которого существует дайджест, Oracle просто вставляет дайджест в запись;
- дайджест для нового сообщения должен быть выбран независимо от всех предыдущих дайджестов. Это подразумевает, что модель Oracle не может использовать формулу или алгоритм для вычисления дайджеста.

Рассмотрим следующий пример. Возьмем модель Oracle с таблицей и правильной монетой. Таблица имеет два столбца: левый столбец — сообщения, дайджесты которых были выработаны; второй столбец перечисляет дайджесты, созданные для этих сообщений. Примем, что дайджест — всегда 16 битов независимо от размера сообщения. В табл. 9.1 приведены примеры сообщений и дайджестов сообщений в шестнадцатеричном исчислении.

Теперь предположим, что возникает три ситуации.

1. Поступает сообщение AB1234CDS765BDAD для вычисления дайджеста. Модель Oracle проверяет свою таблицу. Этого сообще-

Таблица 9.1. Таблица Oracle после создания первых трех дайджестов

Сообщение	Дайджест сообщения
4523AB1352CDEF45126	13AB
723BAE38F2AB3457AC	02CA
AB45CD1048765412AAAB6662BE	A38B

Таблица 9.2. Таблица Oracle после создания четвертого дайджеста

Сообщение	Дайджест сообщения
4523AB1352CDEF45126	13AB
723BAE38F2AB3457AC	02CA
AB1234CD8765BDAD	DCB1
AB45CD1048765412AAAB6662BE	A38B

ния нет в таблице, так что сотрудник, использующий модель Oracle, подбрасывает в воздух свою монету 16 раз. Предположим, что результат — OOPOOOPPOPOOPPPO, в котором буква «O» представляет положение «Орел», буква «P» — положение «Решка». Oracle интерпретирует «O» как 1 бит и «P» как бит 0 и выдает 1101 1100101 10001 в двоичном коде либо DCB1 в шестнадцатеричном. Далее модель использует полученное значение как дайджест для этого сообщения и складывает сообщение и дайджест (показано в табл. 9.2).

2. Сообщение 4523AB1352CDEF45126 поступает для вычисления дайджеста. Модель Oracle проверяет свою таблицу и находит, что есть дайджест для этого сообщения в таблице (первая строка). Oracle просто выдает соответствующий дайджест (13AB).

3. Модель Oracle не может использовать формулу или алгоритм, чтобы создать дайджест для сообщения. Проанализируем это событие методом от противного. Пусть модель Oracle использует формулу $h(M) = M \pmod n$. Теперь предположим, что Oracle уже выдал $h(M_1)$ и $h(M_2)$. Если новое сообщение представлено как $M_3 = M_1 + M_2$,то модель Oracle не сможет вычислить $h\ (M_3)$, так как новый дайджест будет равен $[h(M_1) + h(M_2)] \pmod n$, получим:

$$(M_3) = (M_1 + M_2) \pmod n = M_1 \pmod n + M_2 \pmod n = \\ = [h(M_1) + h(M_2)] \pmod n.$$

Это нарушает третье требование: каждый дайджест должен быть выбран беспорядочно на основе сообщения данного Oracle.

9.3. УСТАНОВЛЕНИЕ ПОДЛИННОСТИ СООБЩЕНИЯ

Для обеспечения подлинности сообщения используется аутентификация сообщения. Целью аутентификации сообщения является подтверждение или отрицание следующих предположений:

- сообщение исходит от законного абонента;
- сообщение при передаче не изменилось;
- сообщение доставлено по требуемому адресу;

последовательность принятых сообщений соответствует последовательности отправленных.

Проверка подлинности особенно важна для криптограмм, поскольку у получателя они вызывают больше недоверия, чем открытый текст. Методы аутентификации разрабатываются в предположении, что установление подлинности производится исключительно по самому сообщению, без привлечения каких-либо внешних средств. Для этого на передающей стороне в сообщение x вводится дополнительно код хеш-функции, который также называют ***сигнатурой, контрольной комбинацией, имитовставкой, дайджестом*** или ***профилем сообщения***. Сообщение x предварительно преобразуется в число или набор чисел.

Хеш-функция отображает сообщение произвольной длины L в последовательность символов фиксированной длины m. Чтобы снижение скорости передачи при таком дополнении не было значительным, естественно требование: проверка подлинности сообщения заключается в *сравнении дайджеста*, вычисленного по принятому x, со значением дайджеста, выделенного из самого сообщения. Если они равны, то принимается решение, что при передаче сообщение не изменилось; в противном случае фиксируется искажение переданного сообщения.

Так как $m << L$, то возможно, что нескольким сообщениям x_1, x_2, ..., x_i, называемым ***коллизиями***, соответствует одно значение хеш-функции $H(x_1) = H(x_2) = ... = H(x_i)$. Замена переданного сообщения на любую коллизию не будет обнаружена при проверке подлинности. Вероятность успешной подделки сообщения определяется разрядностью значения $H(x)$. на практике используются длины в 64—150 бит.

9.4. АБОНЕНТСКОЕ ШИФРОВАНИЕ

Абонентское шифрование реализуется на уровне пользователей. ***Абонентское шифрование —*** защита информации, передаваемой средствами телекоммуникаций, криптографическими методами, непосредственно между отправителем и получателем. Для реализации процесса шифрования пользователи должны выбрать алгоритмы шифрования и их параметры; сгенерировать ключи — открытый и закрытый; обменяться открытыми ключами.

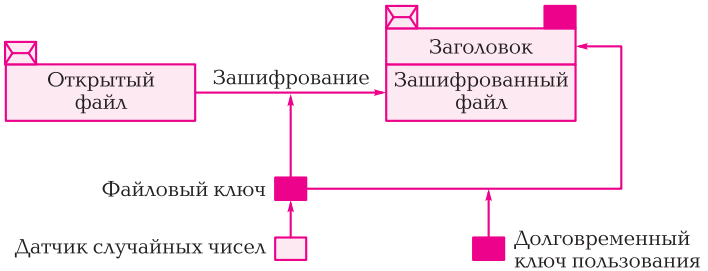

Рис. 9.1. Пример абонентского шифрования

Представленная на рис. 9.1 схема — пример абонентского шифрования.

Ясно, что это не единственно возможная ключевая схема: вариантов таких схем — великое множество, конкретный же выбирается, исходя из требований функциональности и безопасности конкретной системы защиты. В общем случае существует два вида ключевых элементов: файловые и долговременные.

Файловые ключи предназначены для шифрования собственно информации. Часто их название соответствует конкретному виду шифруемых данных: пакетные, сеансовые или дисковые. Обычно такие ключи генерируются случайным образом для каждого шифруемого объекта, например, для каждого файла, сообщения электронной почты, а иногда и IP-пакета.

Долговременные ключи используются для шифрования и передачи файловых. Конечно, это не означает, что для шифрования любого объекта обязательно используется ровно два ключа: файловый и долговременный. Между файловым и долговременным ключами могут располагаться промежуточные, получаемые, например, с помощью датчика случайных чисел (файловый ключ шифруется на промежуточном, который, в свою очередь, шифруется на долговременном).

Независимо от количества ключей, используемых для шифрования конкретного объекта, существует некий особый долговременный ключ, на котором построена вся защита. Если такой ключ попадет в руки злоумышленника, защищенный объект будет им расшифрован, независимо от количества других используемых при шифровании ключей. Именно этот ключ необходимо хранить, не допуская его компрометации (т. е. утери или хищения).

Обычно для хранения подобных ключей используется некий персональный ключевой носитель (дискета, смарт-карта, USB-

токен, iButton и т. д.), который всегда должен находиться у пользователя — владельца ключа.

9.5. ПАКЕТНОЕ ШИФРОВАНИЕ

Для защиты информации при передаче в вычислительных сетях используется пакетное шифрование.

Пакетное шифрование — это шифрование IP-пакетов, осуществляемое на сетевом уровне в соответствии с семиуровневой моделью открытых систем OSI/ISQ непосредственно перед передачей пакетов сетевому интерфейсу (канальному уровню).

Одно из достоинств пакетного шифрования заключается в том, что шифрование невидимо для конечного пользователя и работает независимо от любых других процессов. Данные зашифрованы только во время передачи и существуют в виде открытого текста на исходном и принимающем хостах.

Шифрование пакетов осуществляется коммуникационными программами, которые могут располагаться как на стороне клиента, так и на стороне сервера. Ключевым моментом организации пакетного шифрования является защита целостности коммуникационных программ. Кроме того, шифрование пакетов может быть реализовано в виде отдельного устройства — шифратора IP-пакетов.

Дополнительной мерой защиты при пакетном шифровании может являться использование пакетных ключей. В этом случае при создании нового IP-пакета к нему добавляется пакетный ключ. Генерация пакетного ключа происходит при помощи разделяемого секрета, известного только непосредственным участникам защищенного обмена, поэтому обеспечивается защита от несанкционированного доступа (НСД), а также аутентификация информации на уровне узла сети. При получении IP-пакета с помощью пакетного ключа рассчитывается и проверяется контрольная сумма. Если пакетный ключ будет скомпрометирован, то нарушится безопасность только тех данных, которые были защищены этим ключом.

При пакетном шифровании каждый отдельно взятый IP-пакет не отличается от стандартного незашифрованного, поэтому его использование не влияет на процесс маршрутизации и, следовательно, на выбор сетевого оборудования.

9.6. ОБЕСПЕЧЕНИЕ БЕЗОПАСНОСТИ СЕТИ С ПРИМЕНЕНИЕМ КРИПТОГРАФИЧЕСКИХ ПРОТОКОЛОВ НА ТРАНСПОРТНОМ И СЕТЕВОМ УРОВНЯХ

Назначение сетевого уровня модели взаимодействия открытых систем OSI (Open System Interconnection) — *передача дейтаграмм между хостами*; реализуется передача с помощью двух протоколов. Первый протокол определяет поля дейтаграмм и интерпретацию их содержимого маршрутизаторами и оконечными системами. Для этого используется IP-протокол, входящий в стек протоколов TCP/IP. Второй протокол предназначен для определения оптимального пути передачи дейтаграмм между хостами и называется ***протоколом маршрутизации***. В настоящее время таких протоколов разработано достаточно много.

Для обеспечения безопасности передачи информации в открытых телекоммуникационных сетях на базе существующих протоколов был разработан протокол IPSec (Internet Protocol Security), позволяющий осуществлять шифрование и аутентификацию всего потока данных на уровне IP.

Механизмы защиты протокола IPSec обеспечивают целостность, конфиденциальность и достоверность данных, передаваемых через открытые IP-сети.

Протокол IPSec позволяет объединить в единую защищенную сеть компьютеры центрального офиса и его филиалов. Такая сеть, реализованная на основе общедоступной сети Internet, является виртуальной частной сетью VPN (Virtual Private Network). При формировании защищенных виртуальных каналов на сетевом уровне модели OSI достигается оптимальное соотношение между прозрачностью и качеством защиты. Размещение средств защиты на сетевом уровне делает их прозрачными для приложений, так как между сетевым уровнем и приложением функционирует реализация протокола транспортного уровня. Для пользователей процедуры защиты, как и сам протокол IP, являются абсолютно прозрачными. на сетевом уровне возможна достаточно полная реализация функций защиты трафика и управления ключами, поскольку именно на данном уровне осуществляется маршрутизация пакетов сообщений.

В стеке протоколов IPSec применяются следующие *технологии криптографических преобразований*:

- обмен ключами по алгоритму Диффи — Хеллмана для обмена ключами между пользователями в незащищенной (открытой) сети;
- подтверждение подлинности двух сторон с использованием технологии открытых ключей для подписывания передаваемых данных по алгоритму Диффи — Хеллмана;
- использование алгоритмов аутентификации сообщений на базе функций хеширования;
- использование цифровых сертификатов для подтверждения подлинности открытых ключей;
- применение блочных симметричных алгоритмов шифрования данных.

Протокол IPSec включает в себя следующие компоненты:

- ядро протокола IPSec, которое реализует протоколы ESP и AH, обрабатывает заголовки, взаимодействует с базами данных SPD и SAD для определения политики безопасности пакета;
- протокол управления обменом ключевой информацией IKE;
- база данных политик безопасности SPD (Security Policy Database), в соответствии с которой определяется политика безопасности, применяемая ко всем входящим или исходящим пакетам;
- база данных безопасных ассоциаций SAD (Security Association Database), которая используется для обработки входящей и исходящей информации (данная база данных может заполняться безопасными ассоциациями (SA) вручную или с помощью протокола IKE);
- средства управления политиками и безопасными ассоциациями.

Протокол аутентификации (Authentication Header, AH) обеспечивает аутентификацию сторон в процессе взаимодействия, подписывает данные в составе пакета с использованием электронной подписи, позволяющей удостовериться как в подлинности отправителя, так и в целостности принятых от него данных.

Протокол шифрования (Encapsulated Security Payload, ESP) отвечает за шифрование содержимого отдельных пакетов (и даже некоторых IP-адресов, передаваемых в их составе), а также их аутентификацию и целостность; обеспечивает конфиденциальность. В качестве дополнительной возможности ESP может обеспечивать также аутентификацию. Спецификации на IPSec требуют, чтобы протокол ESP поддерживал использование алгоритмов шифрования: DES, тройной DEA, IDEA и другие.

Протокол IPSec может работать в транспортном и туннельном режимах. В транспортном режиме протокол ESP шифрует все, кроме IP-заголовка. В туннельном режиме шифруется весь IP-пакет и к зашифрованным данным добавляется дополнительный заголовок.

Транспортный режим обеспечивает конфиденциальность для любого использующего этот режим приложения, что позволяет избежать необходимости реализации функций защиты в каждом отдельном приложении. Этот режим достаточно эффективен, а объем добавляемых к пакету IP данных при этом невелик. Недостатком этого режима является то, что IP-адреса пользователей являются открытыми, и поэтому не исключается возможность анализа трафика пересылаемых пакетов.

Туннельный режим ESP в отношении возможности анализа трафика имеет преимущество перед транспортным режимом, так как туннельный режим ESP предлагает шифрование всего пакета IP. Данный режим можно использовать, когда требуется исключить возможность проведения атак, построенных на анализе трафика.

Протокол обмена ключами (Internet Key Exchange, IKE) предназначен для согласования используемых алгоритмов аутентификации и шифрования, ключей и продолжительности их действия, а также для защищенного обмена ключами.

База данных политики безопасности (SPD) определяет, какие сервисы обрабатывают IP-дейтаграммы и каким образом.

Каждая запись в базе данных безопасных ассоциаций (SAD) определяет параметры, связанные с конкретной безопасной ассоциацией (SA). Основой функционирования IPSec являются защищенные виртуальные соединения или безопасные ассоциации SA. Безопасная ассоциация SA представляет собой соглашение о защите обмена данными между двумя взаимодействующими партнерами; соответственно каждая SA имеет запись в SAD.

Протокол транспортного уровня обеспечивает логическое соединение между прикладными процессами, выполняющимися на разных хостах. Существует несколько протоколов транспортного уровня, например, в локальных сетях протоколами транспортного уровня являются TCP и UDP.

Основная задача протокола транспортного уровня на передающей стороне — прием данных от верхнего уровня (например, прикладного) и подготовка их к передаче сетевому уровню. Данная подготовка заключается в разбиении полученной информации на порции (пакеты), удобные для обработки их на сетевом уровне.

На принимающей стороне протокол транспортного уровня из полученных пакетов от сетевого уровня собирает данные для передачи их на верхний уровень (прикладной).

Работа протокола транспортного уровня осуществляется на хосте (или локальном компьютере), поэтому для защиты информации, обрабатываемой протоколом транспортного уровня, в первую очередь, подойдут средства и методы, используемые для защиты локального компьютера.

Затем можно использовать специальные протоколы, работающие на транспортном уровне модели OSI. В качестве примера такого протокола можно рассмотреть *протокол PPTP* (Point-to-Point-Tunneling Protocol, RFC 2637), разработанный компанией Microsoft совместно с компаниями Ascend Communications, 3Com/Primary Access, ECI-Telematics и US Robotics.

Основное предназначение протокола — использование при создании VPN-соединений. Протокол PPTP основывается на протоколе Point-to-Point Protocol (PPP) и является его расширением. Данные верхних уровней модели OSI сначала инкапсулируются в PPP, а затем в PPTP для туннельной передачи через сети общего доступа. PPTP инкапсулирует пакеты IP для передачи по IP-сети. Клиенты PPTP используют порт назначения 1723 для создания управляющего туннелем соединения. Этот процесс происходит на транспортном уровне модели OSI.

После создания туннеля компьютер-клиент и сервер начинают обмен служебными пакетами. В дополнение к управляющему соединению PPTP, обеспечивающему работоспособность канала, создается соединение для пересылки по туннелю данных. Инкапсуляция данных перед отправкой в туннель включает два этапа: сначала создается информационная часть PPP (данные проходят сверху вниз, от прикладного уровня OSI до канального); затем полученные данные отправляются вверх по модели OSI и инкапсулируются протоколами верхних уровней. Таким образом, во время второго прохода данные достигают транспортного уровня.

Однако информация не может быть отправлена по назначению, так как за это отвечает канальный уровень OSI. Поэтому PPTP шифрует поле полезной нагрузки пакета и берет на себя функции второго уровня, обычно принадлежащие PPP, т. е. добавляет к PPTP-пакету PPP-заголовок (header) и окончание (trailer). на этом создание кадра канального уровня заканчивается. Далее, PPTP инкапсулирует PPP-кадр в пакет Generic Routing Encapsulation (GRE), который принадлежит сетевому уровню. GRE инкапсулирует протоколы сетевого уровня, например IPX, AppleTalk, DECnet, чтобы обеспечить возможность их передачи по IP-сетям. Однако GRE не имеет возможности устанавливать сессии и обеспечивать защиту данных от злоумышленников. Для этого используется способность PPTP создавать соединение для управления туннелем. Применение GRE в качестве метода инкапсуляции ограничивает поле действия PPTP только сетями IP.

После того как кадр PPP был инкапсулирован в кадр с заголовком GRE, выполняется инкапсуляция в кадр с IP-заголовком. IP-заголовок содержит адреса отправителя и получателя пакета. В заключение PPTP добавляет PPP-заголовок и окончание. Система-отправитель посылает данные через туннель. Система-получатель удаляет все служебные заголовки, оставляя только данные PPP.

9.7. ПРОТОКОЛ SSL

Протокол SSL (Secure Sockets Layer — уровень защищенных сокетов) представляет собой криптографический протокол, который *обеспечивает защищенную передачу информации в интернете*.

Чаще всего протокол SSL используется с самым распространенным *протоколом передачи гипертекста* — http. О наличии защищенного соединения свидетельствует суффикс *s* — протокол будет называться https. Стандартный порт http — 80, а https — 443.

Протокол SSL позволяет передавать зашифрованную информацию по незасекреченным каналам, обеспечивая надежный обмен между двумя приложениями, работающими удаленно. Протокол состоит из нескольких слоев. Первый слой — это транспортный протокол TCP, обеспечивающий формирование пакета и непосредственную передачу данных по сети. Второй слой — это защитный SSL Record Protocol. При защищенной передаче данных эти два слоя являются обязательными, формируя некое ядро SSL, на которое в дальнейшем накладываются другие слои. Например, это может быть SSL Handshake Protocol, позволяющий устанавливать соответствие между ключами и алгоритмами шифрования. Для усиления защиты передаваемой информации на SSL могут накладываться другие слои.

Для шифрования данных используются криптографические ключи различной степени сложности — 40-, 56- и 128-битные. Показатель количества бит отражает стойкость применяемого шифра, его надежность. Наименее надежными являются 40-битные ключи, так как методом прямого перебора их можно расшифровать в течение 24 ч. В стандартном браузере Internet Explorer по умолчанию используются 40- и 56-битные ключи.

Для передачи данных с помощью SSL на сервере необходимо наличие SSL-сертификата, который содержит сведения о владельце ключа, центре сертификации, данные об открытом ключе (назначение, сфера действия и т. д.). Сервер может требовать от пользователя предоставления клиентского сертификата, если

это предусматривает используемый способ авторизации пользователя.

При использовании сертификата SSL сервер и клиент обмениваются приветственными сообщениями инициализации, содержащими сведения о версии протокола, идентификаторе сессии, способе шифрования и сжатия. Далее сервер отсылает клиенту сертификат или ключевое сообщение, при необходимости требует клиентский сертификат. После нескольких операций происходит окончательное уточнение алгоритма и ключей, отправка сервером финального сообщения и, наконец, обмен секретными данными. Такой процесс идентификации может занимать немало времени, поэтому при повторном соединении обычно используется идентификатор сессии предыдущего соединения.

Таким образом, в настоящее время протокол SSL получил широкое распространение в сети Интернет, так как он обеспечивает достаточно высокий уровень защиты передаваемой между приложениями информации.

9.8. ПРОТОКОЛ TLS

Протокол TLS (Transport Layer Security) основан на протоколе SSL (Secure Sockets Layer), изначально разработанном в Netscape для повышения безопасности электронной коммерции в интернете. Протокол SSL был реализован на прикладном уровне, непосредственно над TCP (Transmission Control Protocol), что позволяет более высокоуровневым протоколам (таким, как HTTP или протоколу электронной почты) работать без изменений. Если SSL сконфигурирован корректно, то сторонний наблюдатель может узнать лишь параметры соединения (например, тип используемого шифрования), а также частоту пересылки и примерное количество данных, но не может читать и изменять их. После того как протокол SSL был стандартизирован IETF (Internet Engineering Task Force), он был переименован в TLS. Поэтому, хотя имена SSL и TLS взаимозаменяемы, они все-таки отличаются, так как каждое описывает другую версию протокола.

Первая выпущенная версия протокола имела название SSL 2.0, но была довольно быстра заменена на SSL 3.0 из-за обнаруженных уязвимостей. Как уже упоминалось, SSL был разработан компанией Netscape, так что в январе 1999 г. IETF открыто стандартизирует его под именем TLS 1.0. Затем в апреле 2006 г. была опубликована версия TLS 1.1, которая расширяла первоначальные возможно-

сти протокола и закрывала известные уязвимости. Актуальная версия протокола на данный момент — TLS 1.2, выпущенная в августе 2008 г.

Протокол TLS предназначен для предоставления трех услуг всем приложениям, работающим над ним, а именно: *шифрование, аутентификацию и целостность*.

Для того чтобы установить криптографически безопасный канал данных, узлы соединения должны согласовать используемые методы шифрования и ключи. Протокол TLS однозначно определяет данную процедуру, подробнее это рассмотрено в пункте TLS Handshake. Следует отметить, что TLS использует криптографию с открытым ключом, которая позволяет узлам установить общий секретный ключ шифрования без каких-либо предварительных знаний друг о друге.

Также в рамках процедуры TLS Handshake имеется возможность установить подлинность личности, и клиента, и сервера. Например: клиент может быть уверен, что сервер, который предоставляет ему информацию о банковском счете, действительно банковский сервер. И наоборот: сервер компании может быть уверен, что клиент, подключившийся к нему, — именно сотрудник компании, а не стороннее лицо. Наконец, TLS обеспечивает отправку каждого сообщения с кодом MAC (Message Authentication Code), алгоритм создания которого — односторонняя криптографическая функция хеширования (фактически — контрольная сумма), ключи которой известны обоим участникам связи. Всякий раз при отправке сообщения, генерируется его MAC-значение, которое может сгенерировать и приемник, это обеспечивает целостность информации и защиту от ее подмены.

9.9. ОРГАНИЗАЦИЯ VPN-СЕТИ

Корпоративные компьютерные сети являются неотъемлемой частью современных компаний. С помощью таких сетей можно оперативно и безопасно передавать и получать информацию. Они обеспечивают связь между компьютерами одного предприятия, расположенными в пределах одного здания или географически распределенными. Для обеспечения связи удаленных компьютеров используются *виртуальные сети* — Virtual Private Network (VPN), которые строятся поверх глобальных сетей WAN (Wide Area Network), охватывающих большое количество ПК и компьютерных систем по всей планете. К их бесспорным достоинствам

относится простота (а соответственно, и невысокая стоимость) построения, возможность подключения множества абонентов, находящихся в разных концах мира, и безопасность передачи данных.

VPN-сети легко масштабируются и являются оптимальным вариантом для предприятий, обладающих множеством филиалов, а также для фирм, чьи сотрудники часто бывают в командировках или работают из дома. Подключение нового офиса или нового удаленного сотрудника осуществляется без дополнительных затрат на коммуникации. Кроме того, первоначальная организация виртуальной системы требует минимум денежных затрат. В дальнейшем финансовые вложения будут сводиться к оплате услуг интернет-провайдера.

Есть у Virtual Private Network и определенные недостатки. Так, фирмам, использующим их, следует позаботиться о безопасности передаваемых данных, потому что документы в процессе передачи проходят через Всемирную сеть, Интернет. Для решения этой задачи используются специальные алгоритмы шифрования данных, позволяющие защитить файлы во время передачи. Кроме того, в виртуальной структуре скорость обмена файлами заметно ниже, чем в ее частных аналогах. Но для передачи небольших объемов информации этого может быть вполне достаточно. Согласно сведениям, предоставленным исследовательской организацией Forrester Research Inc., 41 % предприятий отдают предпочтение офисным сетям потому, что они позволяют решить проблемы с удаленным доступом, 30 % компаний ценят их за экономию денежных средств, а 20 % — за существенное упрощение работы.

В зависимости от особенностей работы фирмы и ее конкретных задач, Virtual Private Network может быть построена по одной из следующих моделей:

- *Remote Access*. В этом случае создается защищенный канал между офисом и удаленным пользователем, подключающимся к ресурсам предприятия с домашнего ПК через Интернет. Подобные системы просты в построении, но менее безопасны, чем их аналоги. Используются предприятиями с большим количеством удаленных сотрудников;
- *Intranet*. Такой вариант позволяет объединить несколько филиалов организации. Передача данных осуществляется по открытым каналам. Intranet может использоваться для обычных филиалов компаний и для мобильных офисов. Но следует иметь в виду, что такой способ предусматривает установку серверов во всех подключаемых офисах;

- *Extranet*. Доступ к информации предприятия предоставляется клиентам и другим внешним пользователям. При этом их возможности по использованию системы заметно ограничены. Непредназначенные для абонентов файлы надежно защищаются средствами шифрования. Это подходящее решение для фирм, которым необходимо обеспечить своим клиентам доступ к определенным сведениям;
- *Client — Server*. Этот вариант позволяет обмениваться данными между несколькими узлами внутри одного сегмента. Он пользуется наибольшей популярностью у организаций, которым необходимо в рамках одной физической сети создать несколько логических (например, отдельные структуры могут быть созданы для финансового отдела, кадровой службы и др.).

Для защиты трафика во время разделения используется шифрование. Наиболее популярными алгоритмами кодирования считаются DES, Triple DES и AES. Наибольшая безопасность обеспечивается специальными протоколами, которые упаковывают данные в единый компонент и формируют соединение (туннель), а также шифруют информацию внутри образованного туннеля. В настоящее время наиболее широко используются следующие наборы протоколов:

- PPTP (Point-to-Point Tunneling Protocol) — туннельный протокол, обеспечивающий сохранение подлинности, сжатие и шифрование данных. Корпорация Microsoft предлагает для протокола PPTP использовать метод шифрования MPPE. Инкапсуляция данных осуществляется путем добавления заголовков GRE и IP;
- L2TP (Layer Two Tunneling Protocol) — протокол, разработанный путем объединения протоколов PPTP и L2F. Он обеспечивает более надежную защиту файлов, чем PPTP. Шифрование осуществляется посредством протокола IPSec (IP-security) или 3DES. Максимальную безопасность передачи данных обеспечивает второй вариант, но его использование приводит к снижению скорости соединения и повышению нагрузки на центральный процессор. Подтверждение подлинности необходимо для того, чтобы информация дошла до адресата в неизмененном виде. Операция выполняется с помощью алгоритмов MD5 и SHA1 и включает проверку целостности документов, а также идентификацию объектов. Идентификация осуществляется как при помощи традиционных операций введения идентификатора и пароля, так и с помощью более надежных средств — сертификатов и серверов для проверки их подлинности.

Для создания VPN-сети необходимо:
1) выбрать провайдера;
2) купить соответствующее сетевое оборудование.

ВОПРОСЫ И ЗАДАНИЯ ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ ЗНАНИЙ

1. Что такое случайная модель Oracle?
2. Какими свойствами обладает функция, основанная на случайной модели Oracle?
3. Что понимается под целостностью сообщения?
4. Как устанавливается подлинность сообщения?
5. Какие виды криптографических протоколов вам известны?
6. Что такое дайджест сообщения?
7. Что такое инкапсуляция?
8. Какие ключевые носители вы знаете?
9. Что такое транспортный режим IPSec?
10. Что такое туннельный режим IPSec?
11. Из каких частей состоит протокол IPSec?
12. Какие варианты построения корпоративных сетей вы знаете?
13. Для чего используется протокол PPTP?
14. Какие ключи используются при абонентском шифровании?
15. Как создается пакетный ключ?
16. Как работает протокол SSL?
17. Чем отличается протокол TLS от протокола SSL?

IV Раздел

ПРОБЛЕМЫ И НАПРАВЛЕНИЯ РАЗВИТИЯ КРИПТОГРАФИИ

Глава 10

ПРОБЛЕМЫ КРИПТОГРАФИИ

10.1. ПРОБЛЕМА ОГРАНИЧЕНИЯ СКОРОСТИ ШИФРОВАНИЯ

В настоящее время резко увеличился объем цифровой информации для обработки. Поэтому большинство современных персональных компьютеров, предназначенных для работы с большими объемами данных, используют SSD-диски, работающие на скорости 700—1 100 МБ/с. Для того чтобы шифрование данных не влияло на скорость работы компьютера, необходимо иметь эффективные по скорости обработки средства криптографической защиты информации. Однако скорость шифрования самого современного шифратора «Криптон» — 10—14 Мбайт/с, что значительно уступает скорости обмена данными современных скоростных дисков.

Данная проблема также крайне актуальна для широко используемых криптосистем с открытым ключом, стойкость которых основана на сложности решения ряда математических задач. Например, криптосистема на базе алгоритма RSA использует в качестве открытого ключа произведение двух больших чисел. Сложность взлома такого алгоритма состоит в трудности решения за приемлемое время задачи факторизации. Из-за роста компьютерных мощностей вычисления с каждым годом становятся все более быстрыми, что повышает вероятность взлома такой криптосистемы. В табл. 10.1 приведены данные разложения на множители 512-битового числа с помощью алгоритма квадратичного решета.

Одним из способов увеличения криптостойкости криптосистем на базе алгоритма RSA является *увеличение размера ключа*. Однако данный способ негативно влияет на скорость шифрования, что, в конечном счете, может привести к отказу от использования подобных криптосистем.

Таблица 10.1. Показатели скорости разложения на множители с помощью алгоритма квадратичного решета

Год	Число десятичных разрядов в разложенном числе	Сравнительная сложность разложения 512-битового числа
1983	71	> 20 млн
1985	80	> 2 млн
1988	90	250 тыс.
1989	100	30 тыс.
1993	120	500
1994	129	100

Как видно, для современных систем передачи данных *скорость шифрования является сдерживающим фактором повышения скорости передачи информации*, который необходимо устранить.

10.2. ПРОБЛЕМЫ ТЕОРИИ АСИММЕТРИЧНЫХ АЛГОРИТМОВ

На сегодняшний день асимметричная криптография вполне успешно решает задачу распределения ключей по открытым каналам связи. Тем не менее существует ряд проблем, вызывающих определенное опасение за ее будущее. Стойкость всех схем асимметричной криптографии основана на невозможности эффективного вычислительного решения ряда таких математических задач, как факторизация больших чисел и логарифмирование в дискретных полях большого размера. Указанная сложность является всего лишь предположением, которое в любой момент может быть опровергнуто. Это привело бы к краху всей современной криптографии, так как задачи, на неразрешимости которых она базируется, достаточно тесно связаны, и взлом даже одной криптосистемы будет означать взлом большинства других.

Другая угроза современным асимметричным криптосистемам исходит от так называемых *квантовых компьютеров* — устройств обработки информации, построенных на принципах квантовой механики, идея которых впервые была предложена известным американским физиком Р. Фейнманом. В 1994 г. П. Шор предложил алгоритм факторизации для квантового компьютера, который позволяет разложить число на множители за время, зависящее полиномиальным образом от размера числа. В 2001 г. этот алгоритм был успешно реализован на созданном специалистами фирмы IBM и Стэнфордского университета первом действующем макете квантового вычислителя. По оценкам специалистов, квантовый компьютер, способный взломать криптосистему RSA, может быть создан примерно через 15—25 лет.

Еще одним недостатком асимметричных криптосистем является необходимость увеличивать минимальный безопасный размер ключей. За сравнительно короткое время существования асимметричных криптосистем длина ключа увеличилась примерно в 10 раз (в отличие от симметричных криптосистем). Следует отметить, что для достижения одинаковой криптостойкости в обеих системах *длина ключа асимметричных криптосистем должна быть на порядок больше*.

Все вышеперечисленное может в будущем привести к отказу от использования асимметричных систем шифрования.

ВОПРОСЫ И ЗАДАНИЯ ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ ЗНАНИЙ

1. На чем основана криптостойкость асимметричных криптосистем?
2. Доказана ли неразрешимость математических задач, лежащих в основе асимметричной криптографии?
3. Как развитие квантовых компьютеров повлияет на стойкость асимметричных криптосистем?
4. В чем особенность алгоритма факторизации П. Шора и как его создание может повлиять на будущее криптографии?
5. Как зависит стойкость криптосистем от размера используемого ключа?
6. Почему нельзя постоянно увеличивать размер ключа?
7. Как различаются размеры ключей в симметричной и асимметричной криптографии?
8. С чем связана необходимость роста скорости шифрования?
9. Для каких криптосистем проблема ограничения скорости шифрования наиболее актуальна?
10. Какая существует взаимосвязь между пропускной способностью канала и скоростью шифрования на сегодняшний день?
11. Какой вывод можно сделать о будущем асимметричной криптографии?

Глава 11

КОМПЬЮТЕРИЗАЦИЯ ШИФРОВАНИЯ

11.1. АППАРАТНОЕ И ПРОГРАММНОЕ ШИФРОВАНИЕ

Существует два основных типа шифрования: аппаратное и программное. ***Аппаратные средства*** криптографической защиты реализованы в виде специализированных физических устройств. Эти устройства встраиваются в линию связи и осуществляют шифрование всей передаваемой по ней информации. ***Программные средства*** криптографической защиты представляют собой программную реализацию различных криптографических алгоритмов. Оба типа шифрования обладают разными преимуществами и недостатками.

Программное шифрование использует ресурсы компьютера для шифрования данных и выполнения других криптографических операций. Поскольку процессор должен обрабатывать как обычные компьютерные операции, так и шифрование данных, это может привести к замедлению работы всей системы. Аппаратное шифрование может использовать встроенные ресурсы и выполняться на собственном оборудовании. Кроме того, узкоспециализированное оборудование разрабатывается для выполнения конкретных задач. Поэтому *аппаратное шифрование обладает более высокой скоростью*.

Программное шифрование часто использует пароль пользователя в качестве ключа шифрования. В то же время аппаратные устройства, предназначенные для шифрования, часто содержит генератор случайных чисел для генерации ключа шифрования, который потом будет разблокирован паролем пользователя. Данная схема усложняет реализацию возможных атак.

Поскольку программное обеспечение шифрования существует в программном обеспечении компьютера, его необходимо переустановить, если операционная система была изменена. Аппаратное шифрование находится за пределами программного обеспечения компьютера и будет оставаться на месте независимо от того, что происходит с программным обеспечением компьютера. Однако программное обеспечение шифрования можно обновить для исправления ошибок и повышения производительности. Программное шифрование также может быть скопировано на другие диски или компьютеры, если необходимо расширить защиту на другие машины. При аппаратном шифровании может быть трудно или невозможно изменить какую-либо его часть.

При расширении защиты на другие машины с использованием средств аппаратного шифрования часто возникает необходимость дополнительной покупки таких средств, что ведет к удорожанию системы защиты.

Атака грубой силы (наиболее распространенная атака на зашифрованные данные) заключается в повторном угадывании пароля или ключа шифрования. Программные схемы шифрования будут пытаться ограничить количество попыток дешифрования или входа в систему, но поскольку они используют ресурсы компьютера, злоумышленники могут получить доступ к памяти компьютера и сбросить счетчик попыток, предоставляя себе неограниченное время для угадывания пароля или ключа. Аппаратное шифрование выполняет свою обработку на выделенном чипе, к которому компьютер не может получить доступ, поэтому взлом с помощью метода перебора не будет работать.

Программное шифрование зависит от безопасности операционной системы. Злоумышленники могут внедрить вредоносный код, получить доступ к управлению системой шифрования и затем, например, изменить ее настройки или же отключить ее. *Аппаратное шифрование выполняется независимо от операционной системы*, поэтому оно не подвержено подобным уязвимостям.

Программное шифрование, как правило, довольно *дешево в реализации*, что делает его очень популярным среди пользователей. Кроме того, *программные процедуры шифрования не требуют дополнительного оборудования*. Аппаратное же шифрование имеет больше преимуществ с точки зрения обеспечения безопасности данных и скорости работы, однако это увеличивает сложность использования и стоимость всей системы безопасности в целом.

Выбор способа шифрования данных и конкретных программно-аппаратных реализаций зависит от параметров самой защищаемой системы, а именно от степени секретности и важности обрабатываемой информации, организации системы в целом, а так-

же от ресурсов, которые могут быть затрачены на создание системы защиты.

11.2. СТАНДАРТИЗАЦИЯ ПРОГРАММНО-АППАРАТНЫХ КРИПТОГРАФИЧЕСКИХ СИСТЕМ И СРЕДСТВ

Стандартизация обеспечивает экономию средств за счет применения апробированных решений и сокращения необоснованного разнообразия на основе унификации и типизации. Активное внедрение информационно-телекоммуникационных технологий в деятельность государственных и коммерческих организаций требует использования стандартизированных средств криптографической защиты информации для обеспечения безопасности хранимых и передаваемых данных.

Стандартизация алгоритмов криптографических преобразований включает всесторонние исследования и публикацию в виде стандартов элементов криптографических процедур с целью использования разработчиками средств криптографической защиты информации (СКЗИ) апробированных криптографически стойких преобразований, обеспечения возможности совместной работы различных СКЗИ, а также возможности тестирования и проверки соответствия реализации СКЗИ заданному стандартом алгоритму.

В России с целью обеспечения деятельности по разработке криптографических стандартов и нормативных документов, регламентирующих их применение, Приказом Ростехрегулирования от 28 декабря 2007 г. был создан технический комитет по стандартизации «Криптографическая защита информации», получивший сокращенное наименование ТК 26. Этим приказом были утверждены положение о техническом комитете, его структура и перечень организаций (предприятий), ставших его членами. За техническим комитетом закреплены объекты стандартизации, относящиеся к методам шифрования (криптографического преобразования) информации, способам их реализации, а также методам обеспечения безопасности информационных технологий с использованием криптографического преобразования информации, включая аутентификацию, имитозащиту и электронную подпись, а также обязанности голосовать по проектам международных стандартов в части криптографической защиты информации.

На текущий момент в России приняты следующие стандарты в области криптографической защиты информации:

- ГОСТ 28147—89 «Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования»;
- ГОСТ Р 34.10—2012 «Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи»;
- ГОСТ Р 34.11—2012 «Информационная технология. Криптографическая защита информации. Функция хеширования»;
- ГОСТ Р 34.12—2015 «Информационная технология. Криптографическая защита информации. Блочные шифры»;
- ГОСТ Р 34.13—2015 «Информационная технология. Криптографическая защита информации. Режимы работы блочных шифров».

Стандарты в области криптографии содержат описание алгоритмов функционирования криптографических механизмов, но не описывают прикладные протоколы информационного взаимодействия, что, в частности, может вызвать серьезные проблемы при обеспечении корректной встречной работы средств криптографической защиты информации различных производителей.

Исследования показывают заинтересованность отечественных компаний в развитии защищенных информационных технологий, основанных на российских криптографических стандартах. Деятельность, проводимая членами ТК 26, позволит в ближайшем будущем обеспечить возможность замены ряда криптографических сервисов, предоставляемых в настоящее время в информационно-телекоммуникационном пространстве с использованием иностранных криптографических алгоритмов, на криптографические сервисы, базирующиеся на российских криптографических алгоритмах.

11.3. СОВРЕМЕННЫЕ ПРОГРАММНЫЕ И АППАРАТНЫЕ КРИПТОГРАФИЧЕСКИЕ СРЕДСТВА

Описанные в предыдущих разделах методы и алгоритмы криптографической защиты информации нашли широкое применение в практической деятельности человека. В первую очередь шифрование стало активно использоваться в средствах вычислительной техники, связанных с обработкой, хранением и передачей данных.

Будем понимать под *средством криптографической защиты информации* (СКЗИ) «средство вычислительной техники, осуществляющее криптографическое преобразование информации для обеспечения ее безопасности».

Под *средством вычислительной техники* (СВТ) понимается «совокупность программных и технических элементов систем обработки данных, способных функционировать самостоятельно или в составе других систем».

Таким образом, ***средства криптографической защиты информации*** — это совокупность программных и технических элементов систем обработки данных, способных функционировать самостоятельно или в составе других систем и осуществлять криптографическое преобразование информации для обеспечения ее безопасности.

По сути, СКЗИ является любое аппаратное, аппаратно-программное или программное решение, тем или иным образом выполняющее криптографическую защиту информации.

Если за основу классификации взять функциональное предназначение СКЗИ, то к ним относятся:

а) *средства шифрования* — аппаратные, программные и аппаратно-программные средства, системы и комплексы, реализующие алгоритмы криптографического преобразования информации и предназначенные для защиты информации при передаче по каналам связи и (или) для защиты информации от несанкционированного доступа при ее обработке и хранении;

б) *средства имитозащиты* — аппаратные, программные и аппаратно-программные средства, системы и комплексы, реализующие алгоритмы криптографического преобразования информации и предназначенные для защиты от навязывания ложной информации;

в) *средства электронной цифровой подписи* — аппаратные, программные и аппаратно-программные средства, обеспечивающие на основе криптографических преобразований реализацию хотя бы одной из следующих функций: создание электронной цифровой подписи с использованием закрытого ключа электронной цифровой подписи, подтверждение с использованием открытого ключа электронной цифровой подписи подлинности электронной цифровой подписи, создание закрытых и открытых ключей электронной цифровой подписи;

г) *средства кодирования* — средства, реализующие алгоритмы криптографического преобразования информации с выполнением части преобразования путем ручных операций или с использованием автоматизированных средств на основе таких операций;

д) *средства изготовления ключевых документов* (независимо от вида носителя ключевой информации);

е) *ключевые документы* (независимо от вида носителя ключевой информации).

Аппаратные средства криптографической защиты представляют собой платы, модули и отдельные системы, выполняющие различные алгоритмы шифрования «на лету». Ключи в таких системах — чаще всего смарт-карты или идентификаторы TouchMemory (iButton). Во избежание перехвата ключей их загружают в устройство не через компьютер, а напрямую. Данные шифраторы позволяют как кодировать данные внутри закрытой системы, так и передавать информацию через открытые каналы связи.

Программные средства защиты криптографической информации (в отличие от аппаратных) — более гибкие и дешевые, однако они имеют и недостатки: так как программное обеспечение легко модифицировать, этим может воспользоваться злоумышленник. Чтобы это предотвратить, необходимо осуществлять контроль целостности программ. Как правило, для этого используется другое программное обеспечение, что не является надежным. Другой существенной проблемой программных криптографических средств является то, что они используют оперативную память при работе с криптографическим ключом, т. е. какое-то время он присутствует в памяти в открытом виде и может быть из нее извлечен. Кроме того, при работе с программой в памяти могут оставаться временные файлы, в которых может находиться полезная для криптоанализа информация.

Помимо этого, недостатком работы с программными средствами защиты информации является то, что для формирования ключа используется датчик случайных чисел, который генерирует ключевую информацию из данных оперативной памяти, значения системных часов и другой псевдослучайной информации. В некоторых компьютерах используется встроенный аппаратный датчик случайных чисел, однако он доступен операционной системе, поэтому для его гарантированной стойкости необходимо использовать доверенную ОС.

Для того чтобы сочетать достоинства программных и аппаратных средств криптографической защиты информации и по возможности преодолеть их недостатки, используют *программно-аппаратные средства криптографической защиты информации*.

Рассмотрим средства криптографической защиты информации, наиболее широко применяемые в настоящее время.

Программный комплекс «Игла-П». Программный комплекс «Игла-П» предназначен для организации защищенных виртуальных приватных сетей (VPN) как самостоятельно, так и совместно с криптографическим комплексом «ШИП». В качестве среды передачи данных могут быть использованы: сеть Internet, выделенные или коммутируемые каналы связи.

Программный комплекс «Игла-П» *обеспечивает*:

- гарантированную стойкость за счет применения криптографических алгоритмов с симметричным распределением ключей по ГОСТ 28147 – 89;
- защиту от раскрытия и искажения защищаемой информации, а также от навязывания ложной информации;
- одностороннюю аутентификацию узлов защищенной сети;
- надежную работу с учетом специфики реальных сетей связи;
- высокую производительность — до 10 Мбит/сек;
  возможность постепенного ввода в эксплуатацию сети связи.

По сравнению с другими продуктами создания VPN (PPTP) и защиты доступа к удаленным услугам (SSL или SSH) программный комплекс «Игла-П» обладает существенными *преимуществами*:

- легкостью, простотой и низкой стоимостью встраивания в существующие системы обработки информации;
- в отличие от использования SSL и PPTP программный комплекс «Игла-П» позволяет блокировать возможности злоумышленника воспользоваться ошибками ОС Windows по сети Internet;
- в отличие от использования SSL применение программного комплекса «Игла-П» в клиент-серверных приложениях, например, в системах электронной коммерции, не требует модификации существующего программного обеспечения;
- тщательно регламентирован и документирован порядок работы с ключевыми документами;
- используемые криптографические алгоритмы сертифицированы ФАПСИ*;

не требуется остановка работы сети при проведении регламентных работ и смене ключей.

Программный комплекс «Игла-П» поставляется в виде программного обеспечения, устанавливаемого на оборудование пользователя.

Криптографический комплекс ШИП. Криптографический комплекс «Шифратор IP-потоков» (ШИП) предназначен для создания защищенных виртуальных приватных сетей (VPN). В качестве среды передачи данных могут быть использованы: сеть Internet, другие сети общего пользования (X.25, Frame Relay), а также выделенные каналы связи. В состав криптографического комплекса ШИП входят аппаратно-программный комплекс ШИП (АПК ШИП) и Центр управления ключевой системой ШИП (ЦУКС ШИП). Криптографический комплекс ШИП *обеспечивает*:

- гарантированную стойкость за счет применения криптографических алгоритмов с симметричным распределением ключей по ГОСТ 28147—89;
- защиту от раскрытия и искажения защищаемой информации, а также от навязывания ложной информации;
- одностороннюю аутентификацию узлов защищенной сети;
- управление ключевой системой защищенной сети из единого центра управления с возможностью резервирования функций управления из других центров;
- защиту доступа к локальной сети и сокрытие ее IP-адресов;
- поддержку протоколов IP и IPX;
- надежную работу с учетом специфики реальных сетей связи;
- высокую производительность — до 20 Мбит/сек;
- возможность постепенного ввода в эксплуатацию сети связи.

По сравнению с другими продуктами создания VPN, реализованными на основе серийных Firewall (межсетевой экран, МЭ), криптографический комплекс ШИП обладает следующими *достоинствами*:

- легкостью и простотой администрирования разветвленной сети;
- в отличие от межсетевых экранов при защите шифраторами автоматизированных систем пользователя к автоматизированным системам не предъявляется требований по показателям защищенности от несанкционированного доступа;
- тщательно регламентирован и документирован порядок работы с ключевыми документами;
- используемые криптографические алгоритмы сертифицированы ФАПСИ;
- не требуется остановка работы сети при проведении регламентных работ и смене ключей;
- возможностью совместного применения программного комплекса «Игла-П», установленного на персональных компьютерах и notebook.

Криптографический комплекс ШИП поставляется в промышленных корпусах, корпусах для монтажа в коммуникационную

* ФАПСИ — Федеральное агентство правительственной связи и информации при президенте Российской Федерации

стойку, а также в виде программного обеспечения, устанавливаемого на оборудование пользователя.

СКЗИ «Верба-OW», «Верба-W». Разработчик данных средств криптографической защиты — ЗАО «Московское отделение Пензенского научно-исследовательского электротехнического института» (МО ПНИЭИ).

Динамические библиотеки СКЗИ «Верба-W» («Верба-OW») предназначены для встраивания в прикладное программное обеспечение, в результате чего к функциям программного обеспечения добавляются возможности шифрования файлов и областей оперативной памяти, формирования и проверки электронной цифровой подписи в соответствии с российскими стандартами ГОСТ Р 34.10—94, ГОСТ Р 34.11—94, ГОСТ 28147—89.

«Верба-W» использует симметричные ключи шифрования и асимметричные ключи для подписи. СКЗИ «Верба-OW» использует принцип открытого распределения ключей шифрования и электронной подписи.

В зависимости от комплектации библиотеки могут реализовывать следующие *возможности*:

- формирование электронной подписи файла (блока) памяти;
- проверку электронной подписи файла (блока) памяти;
- удаление подписи;
- выработка значения хеш-функции файла (блока) памяти;
- зашифрование / расшифрование файлов (блоков) памяти;
- одновременное зашифрование файлов (блоков) памяти в адрес множества абонентов;
- получение имитовставки для файла (блока) памяти;
- формирование случайного числа заданной длины.

Библиотеки предоставляют возможность формирования и администрирования справочников открытых ключей шифрования и ЭП соответственно и обеспечивают выполнение следующих функций при работе со справочниками:

- добавление и удаление открытого ключа;
- получение атрибутов открытого ключа по идентификатору и наоборот;
- контроль целостности справочника открытых ключей;
- контроль целостности открытого ключа.

Криптографические библиотеки в первую очередь предназначены для встраивания в прикладное программное обеспечение заказчика. Для этих целей в комплект поставки включается подробное описание функций и рекомендаций по построению защищенных систем. Вместе с тем на рынок поставляется большое количество приложений, использующих СКЗИ «Верба-W» и «Верба-OW». В их число входят:

- АРМ шифрования и ЭЦП (файловый криптоменеджер);
- защищенная электронная интернет-почта «Курьер»;
- система защиты HTTP-протокола «Корвет»;
- криптографический сервер с системой управления ключами;
- защищенная электронная почта X.400;
- система документооборота БОСС-Референт.

Криптопровайдер «КриптоПро CSP». Криптопровайдер (Cryptography Service Provider, CSP) — это независимый модуль, позволяющий осуществлять криптографические операции (шифрование, электронную подпись и т.д.) в операционной системе. Проще говоря, это посредник между операционной системой и программой или аппаратный комплексом, которые взаимодействуют с пользователем. Большинство программ для шифрования или электронной подписи не будут работать без установленного криптопровайдера.

Криптопровайдер КриптоПро CSP *предназначен*:

- для авторизации и обеспечения юридической значимости электронных документов при обмене ими между пользователями, посредством использования процедур формирования и проверки электронной подписи в соответствии с отечественными стандартами ГОСТ Р 34.10—2001 / ГОСТ Р 34.10—2012 (с использованием ГОСТ Р 34.11—94 / ГОСТ Р 34.11—2012);
- обеспечения конфиденциальности и контроля целостности информации посредством ее шифрования и имитозащиты в соответствии с ГОСТ 28147—89;
- обеспечения аутентичности, конфиденциальности и имитозащиты соединений по протоколу TLS;
- контроля целостности системного и прикладного программного обеспечения для его защиты от несанкционированных изменений и нарушений правильности функционирования;
- управления ключевыми элементами системы в соответствии с регламентом средств защиты.

В нем реализуются следующие алгоритмы:

- алгоритм выработки значения хеш-функции в соответствии с требованиями ГОСТ Р 34.11—94 / ГОСТ Р 34.11—2012 «Информационная технология. Криптографическая защита информации. Функция хеширования»;
- алгоритмы формирования и проверки электронной подписи в соответствии с требованиями ГОСТ Р 34.10—2001 / ГОСТ Р 34.10—2012 «Информационная технология. Криптографическая

защита информации. Процессы формирования и проверки электронной цифровой подписи»;
- алгоритм зашифрования / расшифрования данных и вычисление имитовставки в соответствии с требованиями ГОСТ 28147—89 «Системы обработки информации. Защита криптографическая»;
- при генерации закрытых и открытых ключей обеспечена возможность генерации с различными параметрами в соответствии с ГОСТ Р 34.10—2001 / ГОСТ Р 34.10—2012;
- при выработке значения хеш-функции и шифровании обеспечена возможность использования различных узлов замены в соответствии с ГОСТ Р 34.11—94 и ГОСТ 28147—89.

Криптопровайдер «КриптоПро CSP» поставляется в виде программного обеспечения, устанавливаемого на оборудование пользователя. «КриптоПро CSP» работает в средах: Windows 2000, Windows XP, Windows 7, Windows 10 и др.

Криптопровайдер ViPNet CSP. ViPNet CSP 4.2 — российский криптопровайдер, сертифицированный ФСБ России как средство криптографической защиты информации (СКЗИ) и электронной подписи.

ViPNet CSP 4.2 предоставляет следующие *возможности*:
- создание ключей ЭП, формирование и проверка ЭП по ГОСТ Р 34.10—2001, ГОСТ Р 34.10—2012;
- хеширование данных по ГОСТ Р 34.11—94 и ГОСТ Р 34.11—2012;
- шифрование и имитозащита данных по ГОСТ 28147—89.

ViPNet CSP 4.2 соответствует требованиям ФСБ России к шифровальным (криптографическим) средствам и требованиям к средствам ЭП, утвержденным приказом ФСБ России от 27 декабря 2011 г. № 796. ViPNet CSP 4.2 может использоваться для реализации функций ЭП в соответствии с Федеральным законом от 6 апреля 2011 г. № 63-ФЗ «Об электронной подписи».

Области применения данного СКЗИ включают в себя:
- системы юридически значимого защищенного электронного документооборота;
- сдачу электронной отчетности в государственные органы;
- защищенную работу с веб-сервисами;
- встраивание криптографических функций в сторонние приложения.

Достоинства рассматриваемого СКЗИ:

поддержка работы с внешними устройствами (токенами) для создания и хранения ключей и сертификатов с использованием интерфейса PKCS#11. Данная функция облегчает интеграцию новых устройств с ViPNet CSP 4;

- возможность экспорта и импорта ключей в формате #PKCS12, что повышает совместимость форматов ключей с решениями других производителей;
- поддержка вызова криптографических функций CSP сторонними приложениями через API PKCS#11, Microsoft CryptoAPI и Microsoft CNG;
- выделенное множество функций API позволяет клиентским приложениям ограничивать объемы сертификационных испытаний только проведением оценки влияния.

Криптопровайдер ViPNet CSP поставляется в виде программного обеспечения, устанавливаемого на оборудование пользователя.

Программно-аппаратный комплекс ViPNet Coordinator HW5000, версия 4. Программно-аппаратный комплекс (ПАК) ViPNet Coordinator HW5000 — шлюз безопасности для защиты высокоскоростных каналов связи (до 10 Гбит/сек). ViPNet Coordinator HW5000 позволяет организовать защищенный доступ как в центр обработки данных (ЦОД), так и в корпоративную облачную инфраструктуру. ПАК ViPNet Coordinator HW5000, исполненный в форм-факторе 1U, потребляет низкое количество электроэнергии, обладает невысоким уровнем тепловыделения и не требует каких-либо особых условий для размещения и эксплуатации, представляя собой высокоэффективное средство сетевой защиты.

Совместно с другими программными продуктами линейки ViPNet Network Security, ПАК ViPNet Coordinator HW5000 обеспечивает эффективную реализацию множества сценариев защиты информации, например:
- построение защищенных каналов связи между офисами компании (Site-to-Site и Multi Site-to-Site);
- защищенный доступ удаленных и мобильных пользователей;
- взаимодействие с сетями ViPNet других организаций;
- защита магистральных каналов, соединяющих ЦОДы;
- защита мультисервисных сетей (включая IP-телефонию и видеоконференц-связь);
- разграничение доступа к информации в локальных сетях, сегментирование локальных сетей (например, выделение DMZ*);
- защищенный контролируемый доступ в интернет;
- организация контролируемого доступа пользователей из публичной сети к предоставляемым организацией ресурсам и сервисам.

Достоинства данного шлюза:

---

* DMZ (от англ. *Demilitarized Zone*) — сегмент сети, в котором размещаются общедоступные сервисы.

- VPN без установления соединения обеспечивает повышенную надежность и устойчивость соединений через шлюз безопасности ViPNet Coordinator HW5000;
- поддержка работы в современных мультисервисных сетях связи без ограничений по совместимости со службами DHCP, WINS, DNS; динамическим преобразованием адресов (NAT, PAT); использованием мультимедийных протоколов (SIP, H323, SCCP и др.);
- в качестве центра генерации ключей шифрования используется программное обеспечение VipNet Administrator;
- специальная архитектура файловой системы предотвращает возможность порчи образа операционной системы и программного обеспечения ViPNet при сбоях питания;
- возможность повышения надежности координатора за счет его развертывания в составе кластера горячего резервирования (failover cluster).

Пропускная способность VPN — до 5,5 Гбит/с; пропускная способность L2 VPN — до 4,5 Гбит/с.

Аппаратно-программный комплекс шифрования «Континент». Аппаратно-программный комплекс шифрования «Континент» (АПКШ «Континент») — аппаратно-программный комплекс, позволяющий обеспечить защиту информационных сетей организации от вторжения со стороны сетей общего пользования (интернет), конфиденциальность при передаче информации по открытым каналам связи (VPN), организовать безопасный доступ пользователей VPN к ресурсам сетей общего пользования, а также защищенное взаимодействие сетей различных организаций.

Продукт объединяет межсетевой экран и средство построения VPN-сетей. Является сертифицированным продуктом и обладает сертификатами ФСТЭК* и ФСБ. Разработчик — ООО «Код Безопасности».

Области применения АПКШ «Континент»:

- защита внешнего периметра сети от вредоносного воздействия со стороны сетей общего пользования;
- создание отказоустойчивой VPN-сети между территориально распределенными сетями;
- защита сетевого трафика в мультисервисных сетях;
- разделение сети на сегменты с различным уровнем доступа;
- организация защищенного удаленного доступа к сети для мобильных сотрудников;

* ФСТЭК — Федеральная служба по техническому и экспортному контролю.

- защита пользовательского траффика, использующего беспроводную сеть в качестве канала;
- организация защищенного межсетевого взаимодействия между конфиденциальными сетями.

Возможности АПКШ «Континент»:

- безопасный доступ пользователей VPN к ресурсам сетей общего пользования;
- криптографическая защита передаваемых данных в соответствии с ГОСТ 28147—89;
- межсетевое экранирование — защита внутренних сегментов сети от несанкционированного доступа;
- безопасный доступ удаленных пользователей к ресурсам VPN-сети;
- создание информационных подсистем с разделением доступа на физическом уровне;
- возможность идентификации и аутентификации пользователей, работающих на компьютерах в защищаемой сети криптошлюзов.

Аппаратно-программный комплекс «Застава-150». Аппаратно-программный комплекс (АПК) «Застава-150» предназначен для защиты локальной вычислительной сети (ЛВС) предприятия на сетевом уровне с использованием технологий VPN на основе интернет-протоколов семейства IPSec.

АПК «Застава-150» *обеспечивает*:

- конфиденциальность информации, передаваемой в корпоративной информационно-телекоммуникационной сети (ИТКС);
- защиту доступа к корпоративным вычислительным ресурсам за счет использования протоколов двусторонней криптографической аутентификации;
- контроль целостности данных на основе применения ГОСТ Р 34.11—94 и / или ГОСТ Р 34.11—2012, а также ГОСТ 28147—89 в режиме имитовставки;
- поддержку схемы открытого распределения ключей Диффи — Хеллмана на основе алгоритмов ГОСТ Р 34.10—2001 VKO и (или) ГОСТ Р 34.10—2012 VKO в 256-битном режиме;
- встречную работу со всеми программными и аппаратно-программными исполнениями «Заставы».

Достоинства данного комплекса:

- АПК «Застава-150» позволяет реализовать выполнение требований по информационной безопасности и обеспечивает максимальную простоту и надежность в эксплуатации с возможностью оперативного масштабирования;
- простота и надежность в эксплуатации;

- устойчивость к DDoS-атакам;
- гибкое масштабирование;
- централизованное обновление ПО;
- возможность отправки событий в SIEM*;
- сертифицированные аппаратные средства контроля вскрытия корпуса;
- контроль целостности программной составляющей до загрузки ОС;
- все элементы АПК имеют российское происхождение и разработаны под условия применения в РФ.

Продукты серии «Криптон». Продукты серии «Криптон» производятся фирмой «Анкад». В состав продуктов данной серии входят:

- аппаратно-программный модуль доверенной загрузки «Криптон-замок», предназначенный для обеспечения разграничения и контроля доступа пользователей к техническим средствам вычислительной сети (серверы и рабочие станции). Он позволяет осуществлять идентификацию пользователя до запуска BIOS** компьютера, выполняя при этом двухфакторную аутентификацию пользователя по паролю и отчуждаемому носителю парольно-ключевой информации;
- абонентские шифраторы «Криптон» применяются в составе средств и систем криптографической защиты данных для обеспечения информационной безопасности (в том числе защиты с высоким уровнем секретности) в государственных и коммерческих структурах. Они гарантируют защиту информации, обрабатываемой на персональном компьютере и / или передаваемой по открытым каналам связи. Изделия выполнены в виде плат расширения PCI и PCIExpress для персонального компьютера. Устройства «Криптон» построены на разработанных фирмой «Анкад» специализированных 32-разрядных шифрпроцессорах серии «Блюминг»;
- шифраторы жестких дисков и USB-носителей. Представляют собой аппаратно-программные криптографические комплексы М-590 («Криптон-ПШД/IDE), М-575 («Криптон-ПШД/SATA») и М-623 («Криптон-интеграл») предназначены для защиты информации (в том числе сведений, составляющих государственную тайну) на дисках компьютера и защиты от НСД ресурсов компьютера. В состав комплексов входит модуль проходного аппаратного шифратора серии «Криптон» и подсистема защиты от НСД на базе аппаратно-программного модуля доверенной загрузки «Криптон-замок»;
- сетевые шифраторы «Криптон AncNet» включают в себя изделия М-524Т, М-524Е, М-524К. Они предназначены для защищенной передачи данных в сети и защиты компьютера сети от несанкционированного вмешательства посторонних лиц в его работу;
- система разграничения доступа «Криптон-щит» представляет собой аппаратно-программный комплекс средств защиты информации, предназначенный для защиты от НСД к информации в 32- и 64-битных операционных системах Microsoft Windows. «Криптон-щит» функционирует как на автономных персональных компьютерах, так и на средствах вычислительной техники, объединенных в локальную сеть. Система «Криптон-щит» работает на уровне микроядра операционной системы, независимо от встроенных в ОС средств контроля доступа, и отличается низкими системными требованиями;
- программное средство защиты информации Crypton Lock предназначено для обеспечения санкционированного доступа на сервер либо на рабочую станцию под управлением Microsoft Windows и домена Windows. При использовании Crypton Lock стандартная аутентификация пользователя по логину-паролю заменяется строгой двухфакторной аутентификацией посредством персонального USB-токена Рутокен;
- программный комплекс Crypton IP Mobile предназначен для защиты данных, передаваемых по компьютерным сетям. Комплекс позволяет создавать поверх общедоступных сетей виртуальные частные сети (VPN) с прозрачным шифрованием информации (полным или выборочным) по алгоритму ГОСТ 28147—89 и контролем целостности. Одним из вариантов исполнения продукта является изделие Crypton VPN, предназначенное для защиты передаваемой информации в сетях VPN;
- средство защиты информации от несанкционированного доступа Crypton Disk позволяет создавать секретные логические диски, содержимое которых шифруется в прозрачном (незаметном для пользователя) режиме и доступно только для владельца диска. При чтении какой-либо программной информации с секретного диска эта информация расшифровывается, а при записи — зашифровывается;
- средство криптографической защиты информации Crypton ArcMail обеспечивает конфиденциальность, проверку авторства

* SIEM — управление событиями информационной безопасности; сбор, обработка и анализ событий безопасности, поступающих в систему из множества источников, в едином интерфейсе.

** BIOS (от англ. *Basic Input-Output system*) — базовая система ввода-вывода.

и целостности файлов, каталогов и областей памяти. Crypton ArcMail в едином сервисе предоставляет функции архивирования, ЭП и шифрования. Crypton ArcMail создает подписанный и / или зашифрованный архив, который можно отправлять адресату (адресатам) по открытым каналам связи, в том числе по сети Интернет. Программа выполняет функции шифрования автономно или с помощью аппаратных шифраторов «Криптон» и программного эмулятора Crypton Emulator.

Более подробную информацию о работе приведенных средств криптографической защиты информации можно получить на сайтах разработчиков.

ВОПРОСЫ И ЗАДАНИЯ ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ ЗНАНИЙ

1. Для чего нужна стандартизация криптографических средств и систем?
2. Какой орган в Российской Федерации занимается вопросами стандартизации в области криптографической защиты информации?
3. Назовите действующие стандарты.
4. Какие объекты подлежат сертификации?
5. Какая тенденция наблюдается на сегодняшний день в отношении использования отечественными компаниями российских криптографических стандартов?
6. Что такое средства защиты информации?
7. Как классифицируются аппаратные криптографические средства по назначению?
8. Для каких задач используются аппаратные криптографические средства?
9. Как классифицируются программные криптографические средства по назначению?
10. Для каких задач используются программные криптографические средства?
11. Как реализовано шифрование: аппаратное и программное?
12. В чем заключается различие между аппаратным и программным шифрованием относительно используемых ресурсов?
13. В чем состоит различие между аппаратным и программным шифрованием относительно использования пароля пользователя?
14. В чем состоит различие между аппаратным и программным шифрованием относительно зависимости от ОС?
15. От чего зависит выбор конкретных программно-аппаратных реализаций?

Глава 12

РАЗВИТИЕ КРИПТОГРАФИИ

12.1. КВАНТОВАЯ КРИПТОГРАФИЯ

Квантовая криптография — это сравнительно новое направление исследований, которая позволяет применять эффекты квантовой физики для создания каналов передачи данных. В отличие от классической криптографии, основанной на математических методах, квантовая криптография основана на *физической природе объектов квантовой механики*.

Идею использования квантовой физики для защиты информации впервые предложил Стивен Визнер в 1970 г. Позднее Чарльзом Беннетом и Жилем Брассаром были предложены идеи передачи секретного ключа и создания защищенного квантового канала передачи данных. В 1984 г. ими был создан первый протокол квантового распределения ключа, в котором пользователи Алиса и Боб обмениваются информацией в виде поляризованных фотонов по квантовому каналу.

Основная идея использования квантовой физики при передаче сообщения заключается в *принципе неопределенности Гейзенберга*: невозможно одновременно получить координаты и импульс частицы; невозможно измерить один параметр фотона, не исказив другой. Квантовая система описывается набором неортогональных состояний, при этом невозможно точно определить их: *при измерении одного параметра, состояние всей системы изменяется*. Таким образом, можно создать систему, способную обнаруживать любое подслушивание. Кроме того, в квантовой механике используется *теорема о запрете клонирования*, что делает невозможным создание копий исследуемой системы и последующее их тестирование.

Идеальный квантовый канал. Рассмотрим работу идеального квантового канала, предполагая, что приемно-передающая аппа-

ратура и каналы связи идеальны. В качестве носителей информации в квантовой криптографии, как правило, используются отдельные фотоны или связанные фотонные пары. Значения 0 и 1 битов информации кодируются различными направлениями поляризации фотонов.

Линейной комбинацией векторов $\overline{x_1}$, $\overline{x_2}$, ..., $\overline{x_y}$ называют вектор

$$\overline{y} = \pm_1 \overline{x_1} + \pm_2 \overline{x_2} + \ldots + \pm_y \overline{x_y} = \sum_{k=1}^{y} \pm_k \overline{x_k},$$

где $\alpha_1, \alpha_2, \ldots, \alpha_y \in R(C)$ — коэффициенты линейной комбинации.

Базис — это множество векторов в линейном пространстве таких, что любой вектор пространства может быть представлен в виде некоторой их конечной линейной комбинации, и такое представление для любого вектора единственно.

Для передачи сигнала в квантовой системе отправитель случайным образом выбирает один из двух или в некоторых схемах из трех базисов. Для однозначного детектирования сигнала необходимо, чтобы получатель правильно определил базис, в котором сигнал был передан, в противном случае исход не определен.

Так как базис определяется случайным образом, то при подготовке к передаче информации получатель и отправитель связываются по каналу связи, который может быть не секретным, и определяют дальнейшие способы правильного определения получателем базиса для исключения тех случаев, когда получатель неверно угадал базис. Информация, которой при этом обмениваются, зависит от *квантового протокола*.

В момент осуществления попытки подслушать информацию, передаваемую по квантовому каналу, противник неизбежно будет совершать ошибки в определении базиса, что приведет к искажению данных. Кроме того, при детектировании квант разрушается, поэтому противнику необходимо испустить новый квант, который поляризуется в зависимости от используемого базиса. *А* так как базис может быть определен неверно, то попытка создания нового кванта будет также сопровождаться искажением исходного сигнала. По данным искажениям корреспондентами могут быть обнаружены попытки прослушивания информации.

Квантовый канал передачи данных обладает еще одной особенностью. Для передачи информации используются низкоэнергетические импульсы, которые в идеале должны состоять из одного фотона. Это сильно снижает скорость передачи информации в сравнении с волоконно-оптической связью. В силу данной особенности квантовый канал не эффективен для передачи данных и больше подходит для асимметричного шифрования или схем открытого распределения ключей.

Проблемы реальных систем квантовой криптографии. Идеальная квантовая система способна реагировать на любое прослушивание, что обеспечивает секретность передачи данных, однако реальные системы обладают рядом недостатков.

Во-первых, во время информационного обмена (в силу несовершенства приемо-передающего оборудования) возникают ошибки приема-передачи. Для реального канала связи характерен определенный допустимый уровень ошибок, который система не должна воспринимать как подслушивание. При этом у противника появляется возможность маскировать свою деятельность, выдавая перехват информации за собственные ошибки системы.

Во-вторых, сигналы в линиях связи затухают, что также приводит к увеличению собственных ошибок системы, под которые противник может замаскировать подслушивание. Для решения проблемы затухания отправитель может увеличить мощность сигнала, т. е. число фотонов. У противника появляется возможность отводить часть фотонов, поляризованных одинаковым образом, что не будет влиять на основной сигнал, поэтому воздействие противника не будет детектировано. Данный вид перехвата следует осуществлять максимально близко к отправителю, так как сигнал в этой области обладает максимальной мощностью.

В-третьих, невозможность клонирования состояний фотонов приводит к увеличению уровня ошибок при увеличении мощности квантового сигнала. Это создает трудности для использования усилителей в квантовых системах связи, поэтому обеспечить передачу информации по квантовому каналу возможно только на ограниченные расстояния.

В-четвертых, квантовые законы запрещают лишь идеальное клонирование системы. В настоящее время теоретически доказана возможность однократного копирования системы с вероятностью 5,6, а с ростом числа копий эта вероятность снижается до 2/3. Уровень ошибок при копировании фотонов ниже, чем при угадывании базиса, поэтому у противника с определенной вероятностью появляется возможность скопировать часть передаваемой информации, замаскировав подслушивание под собственные ошибки системы.

Таким образом, реальные системы квантовой связи не способны обеспечить абсолютную секретность передаваемых данных. Наличие собственных ошибок системы делает возможным маски-

ровку прослушивания. Кроме того, уровень собственных ошибок сильно зависит от расстояния между отправителем и получателем, что создает трудности при передаче на дальние расстояния.

Квантовая криптография в настоящий момент активно развивается. Основные направления развития можно условно разбить на следующие группы:

- квантовые коммуникационные технологии (квантовые криптографические системы выработки и распределения ключей);
- технологии квантовой обработки информации (системы квантового шифрования и квантовые генераторы случайных последовательностей);
- технологии квантовых вычислений (квантовые компьютеры и алгоритмы, квантовый криптоанализ);
- постквантовая криптография.

Несмотря на перечисленные недостатки квантовых каналов связи, современные достижения в области квантовой криптографии, а также перспективы дальнейшего развития постепенно выводят квантовую криптографию на практический уровень.

12.2. ПЕРСПЕКТИВЫ РАЗВИТИЯ КРИПТОГРАФИИ

Увеличение роли информационных технологий в современном мире делает задачу информационной безопасности все более важной. Увеличение вычислительных мощностей открывает новые возможности для злоумышленников. Появление и развитие новых технологий порождает необходимость поиска принципиально новых подходов к решению актуальных проблем. Рассмотрим основные направления развития современной криптографии.

Эллиптическая криптография. Эллиптическая криптография — это раздел криптографии, который изучает ***асимметричные криптосистемы, основанные на эллиптических кривых над конечными полями***.

Эллиптической кривой называется множество точек (x, y), удовлетворяющих уравнению:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6.$$

Использование алгебраических свойств эллиптических кривых предложили Миллер и Коблиц в 1985 г. Основным преимуществом криптографии на эллиптических кривых является *отсутствие субэкспоненциальных алгоритмов*, позволяющих решить задачу дискретного логарифмирования в группе точек эллиптической кривой.

В настоящее время эллиптическая криптография получила распространение в таких широко используемых протоколах, как Bitcoin, SSH, TLS. Российский стандарт формирования и проверки электронной цифровой подписи ГОСТ Р 34.10—2012 также основан на использовании эллиптических кривых.

Исследователи N. Tirthani и R. Ganesan в 2014 г. разработали механизм защиты данных в облачных хранилищах, основанный на эллиптических кривых. Важным является использование в протоколе ключа небольшого размера при сохранении криптостойкости. Это позволяет использовать эллиптическую криптографию в условиях ограничения вычислительных ресурсов.

Гомоморфное шифрование. Гомоморфное шифрование — это форма шифрования, позволяющая производить определенные математические действия с зашифрованным текстом и ***получать зашифрованный результат, который соответствует результату операций, выполненных с открытым текстом***. Идею использования операций над зашифрованными данными впервые предложили Рональд Ривест, Леонард Адлеман и Майкл Дертузос в 1978 г. Задача создания системы полного гомоморфного шифрования, т. е. системы, гомоморфной для операций сложения и умножения одновременно, была решена Крейгом Джентри в 2009 г.

Гомоморфное шифрование может применяться в различных областях:

- облачные вычисления. В облачных системах гомоморфное шифрование отлично справляется с задачей обеспечения целостности, доступности и конфиденциальности информации. Для обеспечения производительности следует применять различные алгоритмы, которые максимально подходят для решения поставленной задачи;
- электронное голосование. Система позволит произвести расчеты с зашифрованными данными голосов избирателей, сохраняя анонимность каждого;
- защищенный поиск информации;
- защита беспроводных децентрализованных сетей связи. Гомоморфное шифрование встраивается в протоколы маршрутизации, позволяя производить различные вычисления без расшифрования информации, что усиливает безопасность данных;
- системы с обратной связью. Гомоморфное шифрование обеспечивает сохранение анонимности пользователя, неизменность собранных данных и безопасность внутренних операций для анализа данных;

- обфускация (запутывание кода) для защиты программных продуктов. Гомоморфное шифрование позволяет полностью зашифровать программу, не влияя на ее функционирование.

Легковесная криптография. Легковесная криптография — это раздел криптографии, главной целью которого является ***разработка алгоритмов для применения в устройствах, не обладающих достаточными ресурсами*** для функционирования существующих криптографических систем.

Одним из важнейших направлений развития сети Интернет стал интернет вещей. В современном обществе человек для повышения комфорта, автоматизации своей деятельности использует все большее количество различных электронных устройств, а именно устройства бытовой техники, транспорта, различные датчики и сенсоры, имеющие доступ в интернет. Стремительное развитие данных технологий порождает необходимость обеспечения их информационной безопасности.

Квантовая криптография. Квантовая криптография — это криптография, основной идеей которой является ***использование особенностей физической природы объектов квантовой механики***. Подробно данное направление было рассмотрено в подразд. 12.1.

Постквантовая криптография. Постквантовая криптография — это раздел криптографии, которая изучает ***методы защиты информации, актуальные для эпохи квантовых компьютеров***.

Появление квантовых компьютеров сильно повлияет на современную криптографию. Криптографические системы с открытым ключом, основанные на сложности решения некоторых математических задач, станут неактуальными, так как на текущий момент существуют решения указанных задач с использованием квантовых компьютеров, время выполнения которых полиномиально зависит от размера данных. Квантовые компьютеры сделают прорыв в вычислительной мощности компьютеров, поэтому их появление необходимо обеспечить стойкими криптографическими методами защиты.

Основными разделами постквантовой криптографии являются:

- криптография, основанная на кодах исправления ошибок;
- криптография, основанная на хеш-функциях;
- криптография, основанная на решетках;
- криптография, основанная на многомерных квадратичных системах;
- шифрование с секретным ключом;

- шифрование с использованием суперсингулярной изогении (алгоритма, позволяющего двум и более участникам процесса получить общий секретный ключ, используя незащищенный от прослушивания канал связи).
- Передовые исследования в области криптографии несомненно впечатляют и являются важной инвестицией в будущее.

ВОПРОСЫ И ЗАДАНИЯ ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ ЗНАНИЙ

1. Что такое квантовая криптография?
2. Кем и когда была предложена идея использования квантовой физики для защиты информации?
3. Какой основной принцип квантовой физики лежит в основе квантовой криптографии?
4. Что такое базис и как это понятие используется?
5. От чего идеальный квантовый канал полностью защищен?
6. Какая трудность возникает у злоумышленника при попытке угадывания базиса, при копировании фотонов?
7. Какие проблемы реальной квантовой криптографии существуют?
8. Назовите основные направления развития квантовой криптографии.
9. Какая перспектива ожидает квантовую криптографию в будущем и как развитие квантовых компьютеров повлияет на существующие криптографические системы?
10. Что такое эллиптическая криптография? В чем ее особенность?
11. Где применяется эллиптическая криптография?
12. Что такое гомоморфное шифрование и в чем его преимущества?
13. Назовите актуальные области применения гомоморфного шифрования.
14. Что такое легковесная криптография и чем обусловлена ее актуальность на сегодняшний день?
15. Что такое постквантовая криптография?

Список литературы

Адигеев М. Г. Введение в криптографию. Основные понятия, задачи и методы криптографии. Ч. 1. Методические указания для студентов механико-математического факультета / М. Г. Адигеев. — Ростов н/Д. : Изд-во РГУ, 2002.

Алферов А. П. Основы криптографии : учеб. пособие / А. П. Алферов, А. Ю. Зубов, А. С. Кузьмин. — 2-е изд., испр. и доп. — М. : Гелиос АРВ, 2002.

Анин Б. Ю. Защита компьютерной информации : серия «Мастер» / Б. Ю. Анин. — СПб. : БХВ-Петербург, 2002.

Бабаш А. В. Криптография / А. В. Бабаш, Г. П. Шанкин ; под. ред. В. П. Шерстюка, Э. А. Применко. — М. : Солон-Пресс, 2007.

Басалов Г. В. Основы криптографии / Г. В. Басалов. — М. : Национальный Открытый Университет «ИНТУИТ», 2016.

Грибунин В. Г. Криптография и безопасность цифровых систем : учеб. пособие / [В. Г. Грибунин, А. П. Мартынов, Д. Б. Николаев и др.] ; под ред. А. И. Астайкина. — Саров : ФГУП «РФЯЦ-ВНИИЭФ», 2011.

Криптографическая защита информации: учеб. пособие ; под ред. проф. С. О. Комарова. — М. : РИОР : Инфра-М, 2018.

Лапонина О. Р. Основы сетевой безопасности: криптографические алгоритмы и протоколы взаимодействия / О. Р. Лапонина. — М. : Национальный Открытый Университет «ИНТУИТ», 2016.

Панасенко С. П. Современные методы вскрытия алгоритмов шифрования, ч. 2 [Электронный ресурс] / С. П. Панасенко. — Режим доступа: http://old.cio-world.ru/weekly/293694/

Панасенко С. П. Современные методы вскрытия алгоритмов шифрования, ч. 3 [Электронный ресурс] / С. П. Панасенко. — Режим доступа: http://old.cio-world.ru/weekly/295841/page2.html

Петров А. А. Компьютерная безопасность. Криптографические методы защиты / А. А. Петров. — М. : ДМК, 2000.

Полянская О. Ю. Инфраструктуры открытых ключей / О. Ю. Полянская, В. С. Горбатов. — М. : Национальный Открытый Университет «ИНТУИТ», 2010.

Фергюсон Н. Практическая криптография : [пер. с англ.] / Н. Фергюсон, Б. Шнайдер. — М. : Издательский дом «Вильямс», 2005.

Фомичев В. М. Криптографические методы защиты информации. В 2 ч. Ч. 2. Системные и прикладные аспекты: учебник для академического бакалавриата / В. М. Фомичев, Д. А. Мельников ; под ред. В. М. Фомичева. — М. : Юрайт, 2017

Черемушкин А. В. Криптографические протоколы. Основные свойства и уязвимости : учеб. пособие для студ. учреждений высш. проф. образования / А. В. Черемушкин. — М. : Издательский центр «Академия», 2009.

Шеннон К. Работы по теории информации и кибернетике / К. Шеннон. М. : Изд-во «Иностранная литература», 1963.

Lieven M. K. Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance / M. K. Lieven et al. // Nature 414, 20-27 Dec., 2001.

Shor P. W. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer / P. W. Shor // Proc. the 35th Annual Symp. FOCS, 1994.

Wang X. Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD / X. Wang, D. Feng, X. Lai, X. Yu. — Cryptology ePrint Archive, Report 2004/199 (2007).

20. ГОСТ Р 50922 – 2006. «Защита информации. Основные термины и определения».

Руководящий документ Гостехкомиссии России «Защита от несанкционированного доступа к информации. Термины и определения».

Руководящий документ Гостехкомиссии России «Средства вычислительной техники. Защита от несанкционированного доступа к информации. Показатели защищенности от несанкционированного доступа к информации».

Константин Черезов, ведущий специалист SafeLine, группа компаний «Информзащита»

http://itsec.ru/articles2/Oborandteh/sredstva_kriptograf_zasch_inform

«Положения о разработке, производстве, реализации и эксплуатации шифровальных (криптографических) средств защиты информации (Положение ПКЗ-2005)», утвержденное Приказом ФСБ РФ от 9 февраля 2005 г. № 66.

Оглавление

Учебное издание

Ильин Михаил Евгеньевич,
Калинкина Татьяна Ивановна,
Пржегорлинский Виктор Николаевич

Криптографическая защита информации в объектах информационной инфраструктуры

Учебник

Редактор ~~*О. А. Кузнецова*~~
Компьютерная верстка: *Р. Ю. Волкова*
Корректоры *Л. В. Гаврилина*, ~~*С. Ю. Свиридова*~~

Изд. № 101119535. Подписано в печать 00.00.2018. Формат 60 × 90/16.
Гарнитура «Балтика». Бумага офс. № 1. Печать офсетная. Усл. печ. л. 00,0.
Тираж 0000 экз. Заказ №

ООО «Издательский центр «Академия». www.academia-moscow.ru
129085, Москва, пр-т Мира, 101В, стр. 1. Тел./факс: (495) 648-0507, 616-0029.
Сертификат соответствия № РОСС RU. АД77. Н 02114 от 31.05.2018.

Отпечатано с электронных носителей, предоставленных издательством,
в ОАО «Саратовский полиграфкомбинат». www.sarpk.ru
410004, г. Саратов, ул. Чернышевского, 59.