МЕТОДЫ И СРЕДСТВА КРИПТОГРАФИЧЕСКОЙ ЗАЩИТЫ ИНФОРМАЦИИ

Электронное учебное пособие для дистанционного образования

Новосибирск
Сибирский государственный университет
телекоммуникаций и информатики
2008

Данное электронное учебное пособие базируется на следующем источнике:

Рябко Б. Я., Фионов А. Н. Криптографические методы защиты информации: Учебное пособие для вузов. – М.: Горячая линия–Телеком, 2005. – 229 с.

ОГЛАВЛЕНИЕ

Глава 1. ВВЕДЕНИЕ

Мы начинаем изложение основ криптографии с классической задачи передачи секретных сообщений от некоторого отправителя A к получателю B.

Отправитель сообщений и их получатель могут быть физическими лицами, организациями, какими-либо техническими системами. Иногда об A и B говорят как об абонентах некоторой сети, о пользователях некоторой компьютерной системы или, еще более формально, как об абстрактных «сторонах» (англоязычный термин «party») или «сущностях» (entity), участвующих в информационном взаимодействии. Но чаще бывает удобно отождествлять участников обмена с некоторыми людьми и заменить формальные обозначения A и B на Алиса и Боб.

Предполагается, что сообщения передаются по так называемому «открытому» каналу связи, в принципе доступному для прослушивания некоторым другим лицам, отличным от получателя и отправителя. Такая ситуация возникает при радиопередаче сообщений (например, посредством мобильного телефона) и возможна при использовании даже таких «проверенных» каналов связи, как проволочный телефон, телеграф, да и обычная почта. Особый интерес как средство передачи данных, стремительно завоевывающее лидирующие позиции во всем мире и в то же время чрезвычайно уязвимое с точки зрения возможности несанкционированного доступа третьих лиц, представляет Интернет. В этой среде легко реализуется не только копирование, но и подмена передаваемых сообщений.

В криптографии обычно предполагается, что у лица, передающего сообщения и (или) их принимающего, есть некоторый противник

E, который может быть конкурентом в бизнесе, членом преступной группировки, представителем иностранной разведки или даже чрезмерно ревнивой женой, и этот противник может перехватывать сообщения, передаваемые по открытому каналу, и анализировать их. Часто удобно рассматривать противника как некую особу по имени Ева, которая имеет в своем распоряжении мощную вычислительную технику и владеет методами криптоанализа. Естественно, Алиса и Боб хотят, чтобы их сообщения были непонятны Еве, и используют для этого специальные шифры.

Перед тем как передать сообщение по открытому каналу связи от A к B, A шифрует сообщение, а B, приняв зашифрованное сообщение, дешифрует его, восстанавливая исходный текст. Важно то, что в рассматриваемой нами в этой главе задаче Алиса и Боб могут договариваться об используемом ими шифре (или, скорее, о некоторых его параметрах) не по открытому каналу, а по специальному «закрытому» каналу, недоступному для прослушивания противником. Такой «закрытый канал» может быть организован при помощи курьеров, или же Алиса и Боб могут обмениваться шифрами во время личной встречи и т.п. При этом надо учитывать, что обычно организация такого закрытого канала и передача по нему сообщений слишком дороги по сравнению с открытым каналом и (или) закрытый канал не может быть использован в любое время. Например, курьерская почта намного дороже обычной, передача сообщений с ее помощью происходит намного медленнее, чем, скажем, по телеграфу, да и использовать ее можно не в любое время суток и не в любой ситуации.

Чтобы быть более конкретными, рассмотрим пример шифра. Так как проблема шифрования сообщений возникла еще в глубокой древности, некоторые шифры связаны с именами известных исторических личностей и в качестве первых примеров обычно используют именно такие шифры. Мы также будем придерживаться этой традиции. Начнем с известного шифра Гая Юлия Цезаря (см., например, [2, 23]), адаптировав его к русскому языку. В этом шифре каждая бук-

ва сообщения заменяется на другую, номер которой в алфавите на три больше. Например, А заменяется на Г, Б на Д и т.д. Три последние буквы русского алфавита — Э, Ю, Я — шифруются буквами А, Б, В соответственно. Например, слово ПЕРЕМЕНА после применения к нему шифра Цезаря превращается в ТИУИПИРГ (если исключить букву Ё и считать, что в алфавите 32 буквы).

Последующие римские цезари модифицировали шифр, используя смещение в алфавите на четыре, пять и более букв. Мы можем описать их шифр в общем виде, если пронумеруем (закодируем) буквы русского алфавита числами от 0 до 31 (исключив букву Ё). Тогда правило шифрования запишется следующим образом:

$$c = (m + k) \bmod 32, \tag{1.1}$$

где m и c — номера букв соответственно сообщения и шифротекста, а k — некоторое целое число, называемое ключом шифра (в рассмотренном выше шифре Цезаря $k = 3$). (Здесь и в дальнейшем $a \bmod b$ обозначает остаток от деления целого числа a на целое число b, причем остаток берется из множества $\{0, 1, \ldots, b-1\}$. Например, $13 \bmod 5 = 3$.)

Чтобы дешифровать зашифрованный текст, нужно применить «обратный» алгоритм

$$m = (c - k) \bmod 32. \tag{1.2}$$

Можно представить себе ситуацию, когда источник и получатель сообщений договорились использовать шифр (1.1), но для того, чтобы усложнить задачу противника, решили иногда менять ключ шифра. Для этого Алиса каким-либо образом генерирует число k, передает его Бобу по закрытому каналу связи, и после этого они обмениваются сообщениями, зашифрованными с помощью этого ключа k. Замену ключа можно проводить, например, перед каждым сеансом связи или после передачи фиксированного числа букв (скажем, каждую десятку символов шифровать со своим k) и т.п. В таком случае

говорят, что ключ порождается источником ключа. Схема рассмотренной криптосистемы с секретным ключом приведена на рис. 1.1.

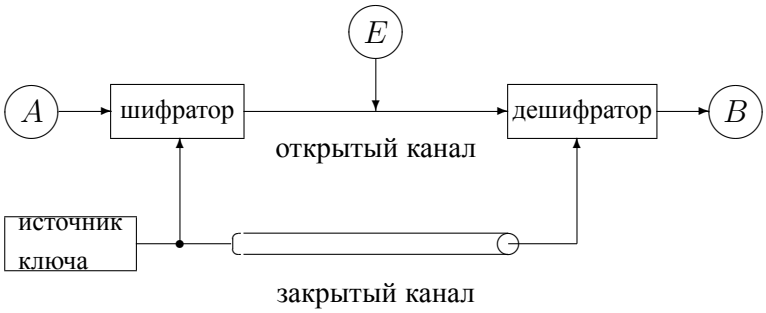

Рис. 1.1. Классическая система секретной связи

Обратимся теперь к анализу действий противника, пытающегося расшифровать сообщение и узнать секретный ключ, иными словами, вскрыть, или взломать шифр. Каждая попытка вскрытия шифра называется атакой на шифр (или на криптосистему). В криптографии принято считать, что противник может знать использованный алгоритм шифрования, характер передаваемых сообщений и перехваченный шифротекст, но не знает секретный ключ. Это называется «правилом Керкхоффса» (см. [23]) в честь ученого, впервые сформулировавшего основные требования к шифрам (A. Kerckhoffs, 1883). Иногда это правило кажется «перестраховкой», но такая «перестраховка» отнюдь не лишняя, если, скажем, передается распоряжение о переводе миллиона долларов с одного счета на другой.

В нашем примере Ева знает, что шифр был построен в соответствии с (1.1), что исходное сообщение было на русском языке и что был передан шифротекст ТИУИПИРГ, но ключ Еве не известен.

Наиболее очевидная попытка расшифровки — последовательный перебор всех возможных ключей (это так называемый метод «гру-

бой силы» (brute-force attack)). Итак, Ева перебирает последовательно все возможные ключи $k = 1, 2, \ldots$, подставляя их в алгоритм дешифрования и оценивая получающиеся результаты. Попробуем и мы использовать этот метод. Результаты дешифрования по (1.2) при различных ключах и шифротексте ТИУИПИРГ сведены в табл. 1.1. В большинстве случаев нам достаточно было расшифровать две–три буквы, чтобы отвергнуть соответствующий ключ (из-за отсутствия слова в русском языке, начинающегося с такого фрагмента).

Таблица 1.1. **Расшифровка слова ТИУИПИРГ путем перебора ключей**

k	m	k	m	k	m	k	m
1	СЗТ	9	ЙЯ	17	БЧ	25	ЩП
2	РЖС	10	ИЮЙ	18	АЦБ	26	ШОЩ
3	ПЕРЕМЕНА	11	ЗЭИ	19	ЯХА	27	ЧН
4	ОДП	12	ЖЬ	20	ЮФ	28	ЦМ
5	НГ	13	ЕЫ	21	ЭУ	29	ХЛЦ
6	МВ	14	ДЪ	22	Ь	30	ФК
7	ЛБМ	15	ГЩ	23	Ы	31	УЙ
8	КАЛАЗ	16	ВШГ	24	Ъ	32	ТИУИПИРГ

Из табл. 1.1 мы видим, что был использован ключ $k = 3$ и зашифровано сообщение ПЕРЕМЕНА. Причем для того, чтобы проверить остальные возможные значения ключа, нам не требовалось дешифровать все восемь букв, а в большинстве случаев после анализа двух–трех букв ключ отвергался (только при $k = 8$ надо было дешифровать пять букв, зато при $k = 22, 23, 24$ хватало и одной, так как в русском языке нет слов, начинающихся с Ь, Ъ, Ы).

Из этого примера мы видим, что рассмотренный шифр совершенно нестоек, для его вскрытия достаточно проанализировать несколько первых букв сообщения и после этого ключ k однозначно определяется (и, следовательно, однозначно дешифруется все сообщение).

В чем же причины нестойкости рассмотренного шифра и как мож-

но было бы увеличить его стойкость? Рассмотрим еще один пример. Алиса спрятала важные документы в ячейке камеры хранения, снабженной пятидекадным кодовым замком. Теперь она хотела бы сообщить Бобу комбинацию цифр, открывающую ячейку. Она решила использовать аналог шифра Цезаря, адаптированный к алфавиту, состоящему из десятичных цифр:

$$c = (m + k) \bmod 10. \tag{1.3}$$

Допустим, Алиса послала Бобу шифротекст 26047. Ева пытается расшифровать его, последовательно перебирая все возможные ключи. Результаты ее попыток сведены в табл. 1.2.

Таблица 1.2. **Расшифровка сообщения 26047 путем перебора ключей**

k	m	k	m
1	15936	6	60481
2	04825	7	59370
3	93714	8	48269
4	82603	9	37158
5	71592	0	26047

Мы видим, что все полученные варианты равнозначны и Ева не может понять, какая именно комбинация истинна. Анализируя шифротекст, она не может найти значения секретного ключа. Конечно, до перехвата сообщения у Евы было 10^5 возможных значений кодовой комбинации, а после — только 10. Однако важно отметить то, что в данном случае всего 10 значений ключа. Поэтому при таком ключе (одна десятичная цифра) Алиса и Боб и не могли расчитывать на бóльшую секретность.

В первом примере сообщение — текст на русском языке, поэтому оно подчиняется многочисленным правилам, различные буквы

и их сочетания имеют различные вероятности и, в частности, многие наборы букв вообще запрещены. (Это свойство называется избыточностью текста). Поэтому-то и удалось легко подобрать ключ и дешифровать сообщение, т.е. избыточность позволила «взломать» шифр. В противоположность этому, во втором примере все комбинации цифр допустимы. «Язык» кодового замка не содержит избыточности. Поэтому даже простой шифр, примененный к сообщениям этого языка, становится невскрываемым. В классической работе К. Шеннона [17] построена глубокая и изящная теория шифров с секретным ключом и, в частности, предложена «правильная» количественная мера избыточности. Мы кратко коснемся этих вопросов в главе 3, а в главе 4 будут описаны современные шифры с секретным ключом.

Описанная в приведенных примерах атака называется атакой по *шифротексту*. Но часто на шифр может быть проведена атака по *известному тексту*. Это происходит, если Ева получает в свое распоряжение какие-либо открытые тексты, соответствующие раннее переданным зашифрованным. Сопоставляя пары «текст–шифротекст», Ева пытается узнать секретный ключ, чтобы с его помощью дешифровать все последующие сообщения от Алисы к Бобу.

Можно представить себе и более «серьезную» атаку — атаку по *выбранному тексту*, когда противник пользуется не только предоставленными ему парами «текст–шифротекст», но может и сам формировать нужные ему тексты и шифровать их с помощью того ключа, который он хочет узнать. Например, во время Второй мировой войны американцы, подкупив охрану, выкрали шифровальную машину в японском посольстве на два дня и имели возможность подавать ей на вход различные тексты и получать соответствующие шифровки. (Они не могли взломать машину с целью непосредственного определения заложенного в нее секретного ключа, так как это было бы замечено и повлекло бы за собой смену всех ключей.)

Может показаться, что атаки по известному и выбранному тексту

надуманы и далеко не всегда возможны. Отчасти это так. Но разработчики современных криптосистем стремятся сделать их неуязвимыми даже и по отношению к атакам по выбранному тексту, и на этом пути достигнуты значительные успехи. Иногда считается, что более надежно использовать шифр, противостоящий атаке по выбранному тексту, чем организационно обеспечивать неосуществимость такой атаки, хотя наиболее осторожные пользователи делают и то, и другое.

Итак, мы познакомились с основными героями криптографии — Алисой, Бобом и Евой и с важными понятиями этой науки — шифром, ключом, атакой, открытым и защищенным каналом. Заметим, что с последним понятием связан один интригующий факт — возможно построение надежных криптосистем без защищенного канала! В таких системах Алиса и Боб вычисляют секретный ключ так, что Ева не может этого сделать. Это открытие было сделано в основополагающих работах Диффи, Хеллмана и Меркля (см., например, [18]) в 1976 году и открыло новую эру в современной криптографии. Большая часть этой книги будет связана именно с такими системами, называемыми схемами с открытым, или несимметричным ключом.

Задачи и упражнения

1.1. Определить ключи шифра Цезаря, если известны следующие пары открытый текст – шифротекст:

а. АПЕЛЬСИН – САЦЬНВЩЮ,

б. АБРИКОС – ЫЬЛГЕЙМ.

См. ответ.

1.2. Расшифровать следующие сообщения, зашифрованные шифром Цезаря с неизвестным ключом k, $0 < k < 32$:

а. ФХНЗКЧ,

б. ЦЩЕБФ.

См. ответ.

Глава 2. КРИПТОСИСТЕМЫ С ОТКРЫТЫМ КЛЮЧОМ

2.1. Предыстория и основные идеи

Рассмотрим три задачи, решение которых поможет нам лучше понять идеи и методы криптографии с открытым ключом. Все эти задачи имеют важное практическое значение.

Первая задача — хранение паролей в компьютере. Мы знаем, что каждый пользователь в сети имеет свой секретный пароль. При входе в сеть пользователь указывает свое имя (несекретное) и затем вводит пароль. Проблема состоит в следующем: если хранить пароль на диске компьютера, то Ева может прочитать его, а затем использовать для несанкционированного доступа (особенно легко это сделать, если Ева работает системным администратором этой сети). Поэтому необходимо организовать хранение паролей в компьютере так, чтобы такой «взлом» был невозможен.

Вторая задача возникла с появлением радиолокаторов и системы ПВО. При пересечении самолетом границы радиолокатор спрашивает пароль. Если пароль верный, то самолет «свой», в противном случае — «чужой». Здесь возникает такая проблема: так как пароль должен передаваться по открытому каналу (воздушной среде), то противник может прослушивать все переговоры и узнавать правильный пароль. Затем «чужой» самолет в случае запроса повторит перехваченный ранее «правильный» пароль в качестве ответа локатору и будет пропущен.

Третья задача похожа на предыдущую и возникает в компьютерных сетях с удаленным доступом, например, при взаимодействии

банка и клиента. Обычно в начале сеанса банк запрашивает у клиента имя, а затем секретный пароль, но Ева может узнать пароль, так как линия связи открытая.

Сегодня все эти проблемы решаются с использованием криптографических методов. Решение всех этих задач основано на важном понятии односторонней функции (one-way function).

Определение 2.1. Пусть дана функция

$$y = f(x), \tag{2.1}$$

определенная на конечном множестве X ($x \in X$), для которой существует обратная функция

$$x = f^{-1}(y). \tag{2.2}$$

Функция называется *односторонней*, если вычисление по формуле (2.1) — простая задача, требующая немного времени, а вычисление по (2.2) — задача сложная, требующая привлечения массы вычислительных ресурсов, например, 10^6–10^{10} лет работы мощного суперкомпьютера.

Данное определение, безусловно, неформально. Строгое определение односторонней функции может быть найдено в [22, 23], но для наших целей достаточно и вышеприведенного.

В качестве примера односторонней функции рассмотрим следующую:

$$y = a^x \bmod p, \tag{2.3}$$

где p — некоторое простое число (т.е. такое, которое делится без остатка только на себя и на единицу), а x — целое число из множества $\{1, 2, \ldots, p-1\}$. Обратная функция обозначается

$$x = \log_a y \bmod p \tag{2.4}$$

и называется *дискретным логарифмом*.

Для того чтобы обеспечить трудность вычисления по (2.4) при использовании лучших современных компьютеров, в настоящее время используются числа размером более 512 бит. На практике часто применяются и другие односторонние функции, например, так называемые хеш-функции, оперирующие с существенно более короткими числами порядка 60–120 бит (они будут рассмотрены в главе 4).

Сначала мы покажем, что вычисление по (2.3) может быть выполнено достаточно быстро. Начнем с примера вычисления числа $a^{16} \mathrm{mod} p$. Мы можем записать

$$a^{16} \mathrm{mod} p = \left(\left(\left(a^2 \right)^2 \right)^2 \right)^2 \mathrm{mod} p,$$

т.е.значение данной функции вычисляется всего за 4 операции умножения вместо 15 при «наивном» варианте $a \cdot a \cdot \ldots \cdot a$. На этом основан общий алгоритм.

Для описания алгоритма введем величину $t = \lfloor \log_2 x \rfloor$ — целую часть $\log_2 x$ (далее все логарифмы будут двоичные, поэтому в дальнейшем мы не будем указывать основание 2). Вычисляем числа ряда

$$a, \quad a^2, \quad a^4, \quad a^8, \quad \ldots, \quad a^{2^t} \qquad (\mathrm{mod}\ p). \tag{2.5}$$

В ряду (2.5) каждое число получается путем умножения предыдущего числа самого на себя по модулю p. Запишем показатель степени x в двоичной системе счисления:

$$x = (x_t x_{t-1} \ldots x_1 x_0)_2.$$

Тогда число $y = a^x \mathrm{mod} p$ может быть вычислено как

$$y = \prod_{i=0}^{t} a^{x_i \cdot 2^i} \ \mathrm{mod}\ p \tag{2.6}$$

(все вычисления проводятся по модулю p).

Пример 2.1. Пусть требуется вычислить $3^{100} \bmod 7$. Имеем $t = \lfloor \log 100 \rfloor = 6$. Вычисляем числа ряда (2.5):

$$\begin{array}{cccccccc} a & a^2 & a^4 & a^8 & a^{16} & a^{32} & a^{64} \\ 3 & 2 & 4 & 2 & 4 & 2 & 4 \end{array} \tag{2.7}$$

Записываем показатель в двоичной системе счисления:

$$100 = (1100100)_2$$

и проводим вычисления по формуле (2.6):

$$\begin{array}{ccccccccccc} a^{64} & & a^{32} & & & & a^4 & & & & \\ 4 & \cdot & 2 & \cdot & 1 & \cdot & 1 & \cdot & 4 & \cdot & 1 & \cdot & 1 & = & 4 \end{array} \tag{2.8}$$

Нам потребовалось всего 8 операций умножения (6 для вычисления ряда (2.7) и 2 для (2.8)). □

В общем случае справедливо следующее

Утверждение 2.1 (о сложности вычислений (2.3)). *Количество операций умножения при вычислении* (2.3) *по описанному методу не превосходит* $2 \log x$.

Доказательство. Для вычисления чисел ряда (2.5) требуется t умножений, для вычисления y по (2.6) не более, чем t умножений (см. пример 2.1). Из условия $t = \lfloor \log x \rfloor$, учитывая, что $\lfloor \log x \rfloor \leq \log x$, делаем вывод о справедливости доказываемого утверждения. □

Замечание. Как будет показано в дальнейшем, при возведении в степень по модулю p имеет смысл использовать только показатели $x < p$. В этом случае мы можем сказать, что количество операций умножения при вычислении (2.3) не превосходит $2 \log p$.

Важно отметить, что столь же эффективные алгоритмы вычисления обратной функции (2.4) неизвестны. Так, один из методов вычисления (2.4), называемый «шаг младенца, шаг великана» (см. [12]), требует порядка $2\sqrt{p}$ операций. Покажем, что при больших p функция (2.3) действительно односторонняя, если для вычисления обратной функции используется метод «шаг младенца, шаг великана». Получаем следующий результат (табл. 2.1).

Таблица 2.1. **Количество умножений для вычисления прямой и обратной функции**

Количество десятичных знаков в записи p	Вычисление (2.3) ($2\log p$ умножений)	Вычисление (2.4) ($2\sqrt{p}$ умножений)
12	$2 \cdot 40 = 80$	$2 \cdot 10^6$
60	$2 \cdot 200 = 400$	$2 \cdot 10^{30}$
90	$2 \cdot 300 = 600$	$2 \cdot 10^{45}$

Мы видим, что если использовать модули, состоящие из 50–100 десятичных цифр, то «прямая» функция вычисляется быстро, а обратная практически не вычислима. Рассмотрим, например, суперкомпьютер, который умножает два 90-значных числа за 10^{-14} сек. (для современных компьютеров это пока не доступно). Для вычисления (2.3) такому компьютеру потребуется

$$T_{\text{выч.пр.}} = 600 \cdot 10^{-14} = 6 \cdot 10^{-12} \text{ сек.},$$

а для вычисления (2.4) —

$$T_{\text{выч.обр.}} = 10^{45} \cdot 10^{-14} = 10^{31} \text{ сек.},$$

т.е. более 10^{22} лет. Мы видим, что вычисление обратных функций практически невозможно при длине чисел порядка 90 десятичных

цифр, и использование параллельных вычислений и компьютерных сетей существенно не меняет ситуацию. В рассмотренном примере мы предполагали, что обратная функция вычисляется за $2\sqrt{p}$ операций. В настоящее время известны и более «быстрые» методы вычисления дискретного логарифма, однако общая картина та же — количество требуемых в них операций много больше $2\log p$. Таким образом, можно утверждать, что функция (2.3) действительно односторонняя, но с оговоркой. Никем не доказано, что обратная функция (2.4) не может быть вычислена столь же быстро, как и «прямая».

Используем одностороннюю функцию (2.3) для решения всех трех задач, описанных в начале данного раздела, не забывая, однако, что точно так же может быть использована и любая другая односторонняя функция.

Начнем с хранения паролей в памяти компьютера. Решение задачи основано на том, что пароли вообще не хранятся! Точнее, при регистрации в сети пользователь набирает свое имя и пароль; пусть, например, его имя — «фрукт», а пароль — «абрикос». Компьютер рассматривает слово «абрикос» как двоичную запись числа x и вычисляет (2.3), где a и p — два несекретные числа, возможно даже, всем известные. После этого в памяти компьютера заводится пара (имя, y), где y вычислено по (2.3) при $x =$ пароль. При всех дальнейших входах этого пользователя после ввода пары («фрукт», «абрикос»), компьютер вычисляет по (2.3) новое значение $y_{\text{нов}}$ с $x =$«абрикос» и сравнивает с хранящимся в памяти ранее вычисленным значением y. Если $y_{\text{нов}}$ совпадает с хранящимся в памяти y, соответствующим данному имени, то это законный пользователь. В противном случае это Ева.

Ева могла бы попытаться найти x по y. Однако мы видели, что уже при 90-значных числах для этого потребуется более чем 10^{22} лет. Таким образом, представленная система хранения пароля весьма надежна и в настоящее время используется во многих реальных операционных системах.

Рассмотрим решение второй задачи (ПВО и самолет). Можно использовать следующий метод. Каждому «своему» самолету присваивается секретное имя, известное системе ПВО и летчику, точнее, бортовому компьютеру. Пусть, например, одному из самолетов присвоено секретное имя СОКОЛ, и этот самолет приближается к границе 01 февраля 2005 года в 12 час.45 мин. Тогда перед приближением к границе бортовой компьютер самолета формирует слово

СОКОЛ	05	02	01	12	45
(имя	год	месяц	день	часы	минуты).

Другими словами, компьютер на самолете и станция ПВО прибавляют к секретному слову сведения о текущем времени и, рассматривая полученное слово как число x, вычисляют $y = a^x \bmod p$, где a и p не секретны. Затем самолет сообщает число y станции ПВО. Станция сравнивает вычисленное ею число y с полученным от самолета. Если вычисленное и полученное значения совпали, то самолет признается как «свой».

Противник не может взломать эту систему. Действительно, с одной стороны, он не знает секретного слова СОКОЛ и не может найти его по y, так как вычисление x по y занимает, скажем, 10^{22} лет. С другой стороны, он не может просто скопировать y и использовать его в качестве ответа в будущем, так как время пересечения границы никогда не повторяется и последующие значения y будут отличаться от первоначального.

Рассмотренный вариант решения «задачи ПВО» требует точной синхронизации часов в самолете и в локаторе. Эта проблема достаточно легко решается. Например, служба навигации постоянно передает метки времени в открытом виде (время не секретно), и все самолеты и локаторы используют эти метки для синхронизации своих часов. Но есть более тонкие проблемы. Метка времени добавляется в слово x для того, чтобы все вычисляемые значения y были различны и противник не мог их повторно использовать. Однако противник

может попытаться мгновенно повторить y в пределах текущей минуты. Как предотвратить эту возможность? Это первый вопрос. Другое затруднение возникает в ситуации, когда самолет посылает число y в конце 45-й минуты, а локатор принимает его в начале 46-й. Мы предоставляем читателю возможность самостоятельно предложить вариант решения этих проблем.

Другой способ решения «задачи ПВО» возможен, если мы будем использовать дополнительный открытый канал передачи данных от локатора к самолету. Как и выше, каждый «свой» самолет и локатор знают секретное слово (типа СОКОЛ), которое не заменяется. Обнаружив цель, локатор посылает ей случайно сгенерированное число a («вызов»). Самолет вычисляет $y = a^x \bmod p$, где x — секретное слово («СОКОЛ»), и сообщает число y локатору. Локатор воспроизводит те же вычисления и сравнивает вычисленное y и принятое. В этой схеме не нужно синхронизировать часы, но, как и ранее, противник не может повторить число y, так как локатор всякий раз посылает разные вызовы (a). Интересно, что эта задача, по-видимому, была исторически первой, при решении которой использовались односторонние функции.

Третья задача решается совершенно аналогично, и оба рассмотренных метода формирования пароля применимы и используются в реальных сетевых протоколах.

2.2. Первая система с открытым ключом — система Диффи–Хеллмана

Эта криптосистема была открыта в середине 70-х годов американскими учеными Диффи (Whitfield Diffie) и Хеллманом (Martin Hellman) и привела к настоящей революции в криптографии и ее практических применениях. Это первая система, которая позволяла защищать информацию без использования секретных ключей, передаваемых по защищенным каналам. Для того чтобы продемонстрировать одну из схем применения таких систем, рассмотрим сеть связи с N пользователями, где N — большое число. Пусть мы хотим организовать секретную связь для каждой пары из них. Если мы будем использовать обычную систему распределения секретных ключей, то каждая пара абонентов должна быть снабжена своим секретным ключом, т.е. всего потребуется $C_N^2 = \frac{N(N-1)}{2} \approx \frac{N^2}{2}$ ключей.

Если абонентов 100, то требуется 5000 ключей, если же абонентов 10^4, то ключей должно быть $5 \cdot 10^7$. Мы видим, что при большом числе абонентов система снабжения их секретными ключами становится очень громоздкой и дорогостоящей.

Диффи и Хеллман решили эту проблему за счет открытого распространения и вычисления ключей. Перейдем к описанию предложенной ими системы.

Пусть строится система связи для абонентов $A, B, C, \ldots$. У каждого абонента есть своя секретная и открытая информация. Для организации этой системы выбирается большое простое число p и некоторое число g, $1 < g < p-1$, такое, что все числа из множества $\{1, 2, \cdots, p-1\}$ могут быть представлены как различные степени $g \bmod p$ (известны различные подходы для нахождения таких чисел g, один из них будет представлен ниже). Числа p и g известны всем абонентам.

Абоненты выбирают большие числа X_A, X_B, X_C, которые хра-

нят в секрете (обычно такой выбор рекомендуется проводить случайно, используя датчики случайных чисел). Каждый абонент вычисляет соответствующее число Y, которое открыто передается другим абонентам,

$$\begin{cases} Y_A = g^{X_A} \bmod p, \\ Y_B = g^{X_B} \bmod p, \\ Y_C = g^{X_C} \bmod p. \end{cases} \tag{2.9}$$

В результате получаем следующую таблицу.

Таблица 2.2. **Ключи пользователей в системе Диффи–Хеллмана**

Абонент	Секретный ключ	Открытый ключ
A	X_A	Y_A
B	X_B	Y_B
C	X_C	Y_C

Допустим, абонент A решил организовать сеанс связи с B, при этом обоим абонентам доступна открытая информация из табл. 2.2. Абонент A сообщает B по открытому каналу, что он хочет передать ему сообщение. Затем абонент A вычисляет величину

$$Z_{AB} = (Y_B)^{X_A} \bmod p \tag{2.10}$$

(никто другой кроме A этого сделать не может, так как число X_A секретно). В свою очередь, абонент B вычисляет число

$$Z_{BA} = (Y_A)^{X_B} \bmod p. \tag{2.11}$$

Утверждение 2.2. $Z_{AB} = Z_{BA}$.

Доказательство. Действительно,

$$Z_{AB} = (Y_B)^{X_A} \bmod p = (g^{X_B})^{X_A} \bmod p = \\ = g^{X_A X_B} \bmod p = (Y_A)^{X_B} \bmod p = Z_{BA}.$$

(Здесь первое равенство следует из (2.10), второе и четвертое — из (2.9), последнее — из (2.11).) □

Отметим главные свойства системы:

1) абоненты A и B получили одно и то же число $Z = Z_{AB} = Z_{BA}$, которое не передавалось по открытой линии связи;

2) Ева не знает секретных чисел $X_A, X_B, \ldots$, поэтому не может вычислить число Z_{AB} (вообще говоря, она могла бы попытаться найти секретное число X_A по Y_A (см. (2.9)), однако при больших p это практически невозможно (требуются миллионы лет)).

Абоненты A и B могут использовать Z_{AB} в качестве секретного ключа для шифрования и дешифрования данных. Таким же образом любая пара абонентов может вычислить секретный ключ, известный только им.

Остановимся теперь на упомянутой выше задаче выбора числа g. При произвольно заданном p она может оказаться трудной задачей, связанной с разложением на простые множители числа $p-1$. Дело в том, что для обеспечения высокой стойкости рассмотренной системы число $p-1$ должно обязательно содержать большой простой множитель (в противном случае алгоритм Полига–Хеллмана, описанный, например, в [23], быстро вычисляет дискретный логарифм). Поэтому часто рекомендуют использовать следующий подход. Простое число p выбирается таким, чтобы выполнялось равенство

$$p = 2q + 1,$$

где q — также простое число. Тогда в качестве g можно взять любое число, для которого справедливы неравенства

$$1 < g < p-1 \quad \text{и} \quad g^q \bmod p \neq 1.$$

П р и м е р 2.2. Пусть $p = 23 = 2 \cdot 11 + 1$ $(q = 11)$. Выберем параметр g. Попробуем взять $g = 3$. Проверим: $3^{11} \bmod 23 = 1$ и значит, такое g не подходит. Возьмем $g = 5$. Проверим: $5^{11} \bmod 23 = 22$. Итак, мы выбрали параметры $p = 23$, $g = 5$. Теперь каждый абонент выбирает секретное число и вычисляет соответствующее ему открытое число. Пусть выбраны $X_A = 7$, $X_B = 13$. Вычисляем $Y_A = 5^7 \bmod 23 = 17$, $Y_B = 5^{13} \bmod 23 = 21$. Пусть A и B решили сформировать общий секретный ключ. Для этого A вычисляет $Z_{AB} = 21^7 \bmod 23 = 10$, а B вычисляет $Z_{BA} = 17^{13} \bmod 23 = 10$. Теперь они имеют общий ключ 10, который не передавался по каналу связи. □

2.3. Элементы теории чисел

Многие криптографические алгоритмы базируются на результатах классической теории чисел. Мы рассмотрим необходимый минимум из этой теории. Классические теоремы Ферма, Эйлера и ряд других результатов из теории чисел будут даны без доказательств, которые могут быть найдены практически в любом учебнике по теории чисел (см., например, [3]). Читатели, знакомые с теорией чисел, могут непосредственно перейти к разд. 2.4.

Определение 2.2. Целое положительное число p называется *простым*, если оно не делится ни на какое другое число, кроме самого себя и единицы.

П р и м е р 2.3. Числа 11, 23 — простые; числа 27, 33 — составные (27 делится на 3 и на 9, 33 делится на 3 и на 11). □

Теорема 2.3 (основная теорема арифметики). *Любое целое положительное число может быть представлено в виде произведения простых чисел, причем единственным образом.*

Пример 2.4. $27 = 3 \cdot 3 \cdot 3,\ 33 = 3 \cdot 11.$ □

Определение 2.3. Два числа называются *взаимно простыми*, если они не имеют ни одного общего делителя кроме единицы.

Пример 2.5. Числа 27 и 28 взаимно просты (у них нет общих делителей кроме единицы), числа 27 и 33 — нет (у них есть общий делитель 3). □

Определение 2.4 (функция Эйлера). Пусть дано целое число $N \geq 1$. Значение *функции Эйлера* $\varphi(N)$ равно количеству чисел в ряду $1, 2, 3, \ldots, N-1$, взаимно простых с N.

Пример 2.6.

$\varphi(10) = ?$	$\varphi(12) = ?$
$1, \not{2}, 3, \not{4}, \not{5}, \not{6}, 7, \not{8}, 9,$	$1, \not{2}, \not{3}, \not{4}, 5, \not{6}, 7, \not{8}, \not{9}, \not{10}, 11$
$\varphi(10) = 4$	$\varphi(12) = 4$

(здесь зачеркнуты числа, не взаимнопростые с аргументом). □

Утверждение 2.4. *Если p — простое число, то $\varphi(p) = p - 1$.*

Доказательство. В ряду $1, 2, 3, \ldots, p-1$ все числа взаимно просты с p, так как p — простое число и по определению не делится ни на какое другое число. □

Утверждение 2.5. *Пусть p и q — два различных простых числа ($p \neq q$). Тогда $\varphi(pq) = (p-1)(q-1)$.*

Доказательство. В ряду $1, 2, \ldots, pq-1$ не взаимнопростыми с pq будут числа

$$p, 2p, 3p, \cdots, (q-1)\,p$$

и

$$q, 2q, 3q, \cdots, (p-1)\,q.$$

Всего таких чисел будет $(q-1)+(p-1)$. Следвательно, количество чисел, взаимнопростых с pq, будет $pq-1-(p-1)-(q-1)=pq-q-p+1=(p-1)(q-1)$. □

Теорема 2.6 (Ферма). *Пусть p — простое число и $0<a<p$. Тогда*

$$a^{p-1} \bmod p = 1.$$

Пример 2.7. $p=13$, $a=2$;

$$2^{12} \bmod 13 = \left(2^2\right)^2 \cdot \left(\left(2^2\right)^2\right)^2 \bmod 13 = 3 \cdot 9 \bmod 13 = 1,$$

$$10^{10} \bmod 11 = 10^2 \cdot \left(\left(10^2\right)^2\right)^2 \bmod 11 = 1 \cdot 1 = 1. \qquad \square$$

Теорема 2.7 (Эйлер). *Пусть a и b — взаимно простые числа. Тогда*

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

Теорема Ферма является частным случаем теоремы Эйлера, когда b — простое число.

Пример 2.8.

$$\varphi(12) = 4,$$

$$5^4 \bmod 12 = \left(5^2\right)^2 \bmod 12 = \left(1^2\right)^2 \bmod 12 = 1.$$

$$\varphi(21) = 2 \cdot 6 = 12,$$

$$2^{12} \bmod 21 = 2^4 \cdot \left(2^4\right)^2 \bmod 21 = 16 \cdot 4 \bmod 21 = 1. \qquad \square$$

Нам понадобится еще одна теорема, близкая к теореме Эйлера.

Теорема 2.8. *Если p и q — простые числа, $p \neq q$ и k — произвольное целое число, то*

$$a^{k\varphi(pq)+1} \bmod (pq) = a. \tag{2.12}$$

Пример 2.9. Возьмем $p = 5$, $q = 7$. Тогда $pq = 35$, а функция Эйлера — $\varphi(35) = 4 \cdot 6 = 24$. Рассмотрим случай $k = 2$, т.е. будем возводить числа в степень $2 \cdot 24 + 1 = 49$. Получим

$$9^{49} \bmod 35 = 9, \quad 23^{49} \bmod 35 = 23.$$

Это не удивительно, так как каждое из чисел 9 и 23 взаимно просто с модулем 35, и по теореме Эйлера $9^{24} \bmod 35 = 1$, $23^{24} \bmod 35 = 1$. Однако теорема 2.8 остается верной и для следующих чисел:

$$10^{49} \bmod 35 = 10, \quad 28^{49} \bmod 35 = 28,$$

в то время как теорема Эйлера для них не применима (каждое из чисел 10 и 28 не взаимно просто с модулем 35 и $10^{24} \bmod 35 = 15$, $28^{24} \bmod 35 = 21$). □

Определение 2.5. Пусть a и b — два целых положительных числа. *Наибольший общий делитель* чисел a и b есть наибольшее число c, которое делит и a и b:

$$c = \gcd(a, b).$$

(Обозначение gcd для наибольшего общего делителя происходит от английских слов greatest common divisor и принято в современной литературе.)

Пример 2.10. $\gcd(10, 15) = 5$; $\gcd(8, 28) = 4$. □

Для нахождения наибольшего общего делителя можно использовать следующий алгоритм, известный как алгоритм Евклида.

Алгоритм 2.1. АЛГОРИТМ ЕВКЛИДА

ВХОД: Положительные целые числа a,b, $a \geq b$.
ВЫХОД: Наибольший общий делитель $\gcd(a, b)$.
1. WHILE $b \neq 0$ DO
2. $r \leftarrow a \bmod b$, $a \leftarrow b$, $b \leftarrow r$.
3. RETURN a.

П р и м е р 2.11. Покажем, как с помощью алгоритма Евклида вычисляется $\gcd(28, 8)$:

$$\begin{array}{rcccc} a: & 28 & 8 & 4 \\ b: & 8 & 4 & 0 \\ r: & 4 & 0 & \end{array}$$

Здесь каждый столбец представляет собой очередную итерацию алгоритма. Процесс продолжается до тех пор, пока b не станет равным нулю. Тогда в значении переменной a содержится ответ (4). □

Для многих криптографических систем, рассматриваемых в следующих разделах и главах, актуален так называемый обобщенный алгоритм Евклида, с которым связана следующая теорема.

Теорема 2.9. *Пусть a и b — два целых положительных числа. Тогда существуют целые (не обязательно положительные) числа x и y, такие, что*

$$ax + by = \gcd(a, b). \tag{2.13}$$

Обобщенный алгоритм Евклида служит для отыскания $\gcd(a, b)$ и x, y, удовлетворяющих (2.13). Введем три строки $U = (u_1, u_2, u_3)$, $V = (v_1, v_2, v_3)$ и $T = (t_1, t_2, t_3)$. Тогда алгоритм записывается следующим образом.

Алгоритм 2.2. Обобщенный алгоритм Евклида

ВХОД: Положительные целые числа a,b, $a \geq b$.
ВЫХОД: $\gcd(a,b)$, x, y, удовлетворяющие (2.13).
1. $U \leftarrow (a,1,0)$, $V \leftarrow (b,0,1)$.
2. WHILE $v_1 \neq 0$ DO
3. $q \leftarrow u_1 \operatorname{div} v_1$;
4. $T \leftarrow (u_1 \bmod v_1, u_2 - qv_2, u_3 - qv_3)$;
5. $U \leftarrow V$, $V \leftarrow T$.
6. RETURN $U = (\gcd(a,b), x, y)$.

Результат содержится в строке U.

Операция div в алгоритме — это целочисленное деление

$$a \operatorname{div} b = \lfloor a/b \rfloor.$$

Доказательство корректности алгоритма 2.2 может быть найдено в [1, 9].

Пример 2.12. Пусть a=28, b=19. Найдем числа x и y, удовлетворяющие (2.13).

U				28	1	0	
V	U			19	0	1	
T	V	U		9	1	-1	$q=1$
	T	V	U	1	-2	3	$q=2$
		T	V	0	19	-28	$q=9$

Поясним представленную схему. Вначале в строку U записываются числа (28,1,0), а в строку V — числа (19,0,1) (это первые две строки на схеме). Вычисляется строка T (третья строка в схеме). После этого в качестве строки U берется вторая строка в схеме, а в качестве V — третья, и опять вычисляется строка T (четвертая строка в схеме).

Этот процесс продолжается до тех пор, пока первый элемент строки V не станет равным нулю. Тогда предпоследняя строка в схеме содержит ответ. В нашем случае $\gcd(28, 19) = 1$, $x = -2$, $y = 3$. Выполним проверку: $28 \cdot (-2) + 19 \cdot 3 = 1$. □

Рассмотрим одно важное применение обобщенного алгоритма Евклида. Во многих задачах криптографии для заданных чисел c, m требуется находить такое число $d < m$, что

$$cd \bmod m = 1. \tag{2.14}$$

Отметим, что такое d существует тогда и только тогда, когда числа c и m взаимно простые.

Определение 2.6. Число d, удовлетворяющее (2.14), называется *инверсией c по модулю m* и часто обозначается $c^{-1} \bmod m$.

Данное обозначение для инверсии довольно естественно, так как мы можем теперь переписать (2.14) в виде

$$cc^{-1} \bmod m = 1.$$

Умножение на c^{-1} соответствует делению на c при вычислениях по модулю m. По аналогии можно ввести произвольные отрицательные степени при вычислениях по модулю m:

$$c^{-e} = (c^e)^{-1} = (c^{-1})^e \pmod m.$$

Пример 2.13. $3 \cdot 4 \bmod 11 = 1$, поэтому число 4 — это инверсия числа 3 по модулю 11. Можно записать $3^{-1} \bmod 11 = 4$. Число $5^{-2} \bmod 11$ может быть найдено двумя способами:

$$5^{-2} \bmod 11 = (5^2 \bmod 11)^{-1} \bmod 11 = 3^{-1} \bmod 11 = 4,$$

$$5^{-2} \bmod 11 = (5^{-1} \bmod 11)^2 \bmod 11 = 9^2 \bmod 11 = 4.$$

При вычислениях по второму способу мы использовали равенство $5^{-1} \bmod 11 = 9$. Действительно, $5 \cdot 9 \bmod 11 = 45 \bmod 11 = 1$. □

Покажем, как можно вычислить инверсию с помощью обобщенного алгоритма Евклида. Равенство (2.14) означает, что для некоторого целого k

$$cd - km = 1. \tag{2.15}$$

Учитывая, что c и m взаимно просты, перепишем (2.15) в виде

$$m(-k) + cd = \gcd(m, c), \tag{2.16}$$

что полностью соответствует (2.13), здесь только по-другому обозначены переменные. Поэтому, чтобы вычислить $c^{-1} \bmod m$, т.е. найти число d, нужно просто использовать обобщенный алгоритма Евклида для решения уравнения (2.16). Заметим, что значение переменной k нас не интересует, поэтому можно не вычислять вторые элементы строк U, V, T. Кроме того, если число d получается отрицательным, то нужно прибавить к нему m, так как по определению число $a \bmod m$ берется из множества $\{0, 1, \ldots, m-1\}$.

Пример 2.14. Вычислим $7^{-1} \bmod 11$. Используем такую же схему записи вычислений, как в примере 2.12 :

$$\begin{array}{rrl} 11 & 0 & \\ 7 & 1 & \\ 4 & -1 & q = 1 \\ 3 & 2 & q = 1 \\ 1 & -3 & q = 1 \\ 0 & 11 & q = 3. \end{array}$$

Получаем $d = -3$ и $d \bmod 11 = 11 - 3 = 8$, т.е. $7^{-1} \bmod 11 = 8$. Проверим результат: $7 \cdot 8 \bmod 11 = 56 \bmod 11 = 1$. □

Одной из важнейших операций в криптографии с открытыми ключами является операция возведения в степень по модулю. Идея построения эффективного алгоритма возведения степень была ранее

проиллюстрирована с помощью (2.5) и (2.6). Рассмотренный алгоритм можно реализовать и без хранения в памяти ряда чисел (2.5). Дадим описание этого алгоритма в форме, пригодной для непосредственной программной реализации. В названии алгоритма отражен тот факт, что биты показателя степени просматриваются справа-налево, т.е. от младшего к старшему.

Алгоритм 2.3. ВОЗВЕДЕНИЕ В СТЕПЕНЬ (СПРАВА-НАЛЕВО)

ВХОД: Целые числа $a,\ x = (x_t x_{t-1} \dots x_0)_2,\ p$.
ВЫХОД: Число $y = a^x \bmod p$.
1. $y \leftarrow 1,\ s \leftarrow a$.
2. FOR $i = 0, 1, \dots, t$ DO
3. IF $x_i = 1$ THEN $y \leftarrow y \cdot s \bmod p$;
4. $s \leftarrow s \cdot s \bmod p$.
5. RETURN y.

Чтобы показать, что по представленному алгоритму действительно вычисляется y согласно (2.6), запишем степени переменных после каждой итерации цикла. Пусть $x = 100 = (1100100)_2$, как в примере 2.1, тогда:

i:	0	1	2	3	4	5	6
x_i:	0	0	1	0	0	1	1
y:	1	1	a^4	a^4	a^4	a^{36}	a^{100}
s:	a^2	a^4	a^8	a^{16}	a^{32}	a^{64}	a^{128}

В некоторых ситуациях более эффективным оказывается следующий алгоритм, в котором биты показателя степени просмативаются слева-направо, т.е. от старшего к младшему.

Алгоритм 2.4. ВОЗВЕДЕНИЕ В СТЕПЕНЬ (СЛЕВА-НАПРАВО)

ВХОД: Целые числа $a,\ x = (x_t x_{t-1} \dots x_0)_2,\ p$.
ВЫХОД: Число $y = a^x \bmod p$.
1. $y \leftarrow 1$.
2. FOR $i = t, t-1, \dots, 0$ DO
3. $y \leftarrow y \cdot y \bmod p$;
4. IF $x_i = 1$ THEN $y \leftarrow y \cdot a \bmod p$.
5. RETURN y.

Чтобы убедиться в том, что алгоритм 2.4 вычисляет то же самое, что и алгоритм 2.3, запишем степени переменной y после каждой итерации цикла для $x = 100$:

i:	6	5	4	3	2	1	0
x_i:	1	1	0	0	1	0	0
y:	a	a^3	a^6	a^{12}	a^{25}	a^{50}	a^{100}

Приведенных в данном разделе сведений из теории чисел будет достаточно для описания основных криптографических алгоритмов и методов.

2.4. Шифр Шамира

Этот шифр, предложенный Шамиром (Adi Shamir), был первым, позволяющим организовать обмен секретными сообщениями по открытой линии связи для лиц, которые не имеют никаких защищенных каналов и секретных ключей и, возможно, никогда не видели друг друга. (Напомним, что система Диффи–Хеллмана позволяет сформировать только секретное слово, а передача сообщения потребует использования некоторого шифра, где это слово будет использоваться как ключ.)

Перейдем к описанию системы. Пусть есть два абонента A и B, соединенные линией связи. A хочет передать сообщение m абоненту B так, чтобы никто не узнал его содержание. A выбирает случайное большое простое число p и открыто передает его B. Затем A выбирает два числа c_A и d_A, такие, что

$$c_A d_A \bmod (p-1) = 1. \tag{2.17}$$

Эти числа A держит в секрете и передавать не будет. B тоже выбирает два числа c_B и d_B, такие, что

$$c_B d_B \bmod (p-1) = 1, \tag{2.18}$$

и держит их в секрете.

После этого A передает свое сообщение m, используя трехступенчатый протокол. Если $m < p$ (m рассматривается как число), то сообщение m передается сразу , если же $m \geq p$, то сообщение предствляется в виде $m_1, m_2, \ldots, m_t$, где все $m_i < p$, и затем передаются последовательно $m_1, m_2, \ldots, m_t$. При этом для кодирования каждого m_i лучше выбирать случайно новые пары (c_A, d_A) и (c_B, d_B) — в противном случае надежность системы понижается. В настоящее время такой шифр, как правило, используется для передачи чисел, например, секретных ключей, значения которых меньше p. Таким обра-

зом, мы будем рассматривать только случай $m < p$. Дадим описание протокола.

Шаг 1. A вычисляет число

$$x_1 = m^{c_A} \bmod p, \tag{2.19}$$

где m — исходное сообщение, и пересылает x_1 к B.

Шаг 2. B, получив x_1, вычисляет число

$$x_2 = x_1^{c_B} \bmod p \tag{2.20}$$

и передает x_2 к A.

Шаг 3. A вычисляет число

$$x_3 = x_2^{d_A} \bmod p \tag{2.21}$$

и передает его B.

Шаг 4. B, получив x_3, вычисляет число

$$x_4 = x_3^{d_B} \bmod p. \tag{2.22}$$

Утверждение 2.10 (свойства протокола Шамира).

1) $x_4 = m$, *т.е. в результате реализации протокола от* A *к* B *действительно передается исходное сообщение;*

2) *злоумышленник не может узнать, какое сообщение было передано.*

Д о к а з а т е л ь с т в о . Вначале заметим, что любое целое число $e \geq 0$ может быть представлено в виде $e = k(p-1) + r$, где $r = e \bmod (p-1)$. Поэтому на основании теоремы Ферма

$$x^e \bmod p = x^{k(p-1)+r} \bmod p = \\ = (1^k \cdot x^r) \bmod p = x^{e \bmod (p-1)} \bmod p. \quad (2.23)$$

Справедливость первого пункта утверждения вытекает из следующей цепочки равенств:

$$\begin{aligned} x_4 &= x_3^{d_B} \bmod p = (x_2^{d_A})^{d_B} \bmod p = \\ &= (x_1^{c_B})^{d_A d_B} \bmod p = (m^{c_A})^{c_B d_A d_B} \bmod p = \\ &= m^{c_A d_A c_B d_B} \bmod p = m^{(c_A d_A c_B d_B) \bmod (p-1)} \bmod p = m \end{aligned}$$

(предпоследнее равенство следует из (2.23), а последнее выполняется в силу (2.17) и (2.18)).

Доказательство второго пункта утверждения основано на предположении, что для злоумышленника, пытающегося определить m, не существует стратегии более эффективной, чем следующая. Вначале он вычисляет c_B из (2.20), затем находит d_B и, наконец, вычисляет $x_4 = m$ по (2.22). Но для осуществления этой стратегии злоумышленник должен решить задачу дискретного логарифмирования (2.20), что практически невозможно при больших p. □

Опишем метод нахождения пар c_A, d_A и c_B, d_B, удовлетворяющих (2.17) и (2.18). Достаточно описать только действия для абонента A, так как действия для B совершенно аналогичны. Число c_A выбираем случайно так, чтобы оно было взаимно простым с $p-1$ (поиск целесообразно вести среди нечетных чисел, так как $p-1$ четно). Затем вычисляем d_A с помощью обобщенного алгоритма Евклида, как это было объяснено в разд. 2.3.

П р и м е р 2.15. Пусть A хочет передать B сообщение $m = 10$. A выбирает $p = 23$, $c_A = 7$ $(\gcd(7, 22) = 1)$ и вычисляет $d_A = 19$.

Аналогично, B выбирает параметры $c_B = 5$ (взаимно простое с 22) и $d_B = 9$. Переходим к протоколу Шамира.

Шаг 1. $x_1 = 10^7 \bmod 23 = 14$.

Шаг 2. $x_2 = 14^5 \bmod 23 = 15$.

Шаг 3. $x_3 = 15^{19} \bmod 23 = 19$.

Шаг 4. $x_4 = 19^9 \bmod 23 = 10$.

Таким образом, B получил передаваемое сообщение $m = 10$. □

2.5. Шифр Эль-Гамаля

Пусть имеются абоненты A, B, C, ..., которые хотят передавать друг другу зашифрованные сообщения, не имея никаких защищенных каналов связи. В этом разделе мы рассмотрим шифр, предложенный Эль-Гамалем (Taher ElGamal), который решает эту задачу, используя, в отличие от шифра Шамира, только одну пересылку сообщения. Фактически здесь используется схема Диффи–Хеллмана, чтобы сформировать общий секретный ключ для двух абонентов, передающих друг другу сообщение, и затем сообщение шифруется путем умножения его на этот ключ. Для каждого следующего сообщения секретный ключ вычисляется заново. Перейдем к точному описанию метода.

Для всей группы абонентов выбираются некоторое большое простое число p и число g, такие, что различные степени g суть различные числа по модулю p (см. разд. 2.2). Числа p и g передаются абонентам в открытом виде (они могут использоваться всеми абонентами сети).

Затем каждый абонент группы выбирает свое секретное число c_i, $1 < c_i < p-1$, и вычисляет соответствующее ему открытое число d_i,

$$d_i = g^{c_i} \bmod p. \tag{2.24}$$

В результате получаем таблицу 2.3.

Т а б л и ц а 2.3. **Ключи пользователей в системе Эль-Гамаля**

Абонент	Секретный ключ	Открытый ключ
A	c_A	d_A
B	c_B	d_B
C	c_C	d_C

Покажем теперь, как A передает сообщение m абоненту B. Будем предполагать, как и при описании шифра Шамира, что сообщение представлено в виде числа $m < p$.

Шаг 1. A формирует случайное число k, $1 \le k \le p-2$, вычисляет числа

$$r = g^k \bmod p, \tag{2.25}$$

$$e = m \cdot {d_B}^k \bmod p \tag{2.26}$$

и передает пару чисел (r,e) абоненту B.

Шаг 2. B, получив (r,e), вычисляет

$$m' = e \cdot r^{p-1-c_B} \bmod p. \tag{2.27}$$

Утверждение 2.11 (свойства шифра Эль-Гамаля).

1) *Абонент B получил сообщение, т.е. $m' = m$;*

2) *противник, зная p, g, d_B, r и e, не может вычислить m.*

Доказательство. Подставим в (2.27) значение e из (2.26):

$$m' = m \cdot {d_B}^k \cdot r^{p-1-c_B} \bmod p.$$

Теперь вместо r подставим (2.25), а вместо d_B — (2.24):

$$m' = m \cdot (g^{c_B})^k \cdot (g^k)^{p-1-c_B} \bmod p = \\ = m \cdot g^{c_B k + k(p-1) - k c_B} \bmod p = m \cdot g^{k(p-1)} \bmod p.$$

По теореме Ферма

$$g^{k(p-1)} \bmod p = 1^k \bmod p = 1,$$

и, таким образом, мы получаем первую часть утверждения.

Для доказательства второй части заметим, что противник не может вычислить k в равенстве (2.25), так как это задача дискретного логарифмирования. Следовательно, он не может вычислить m в равенстве (2.26), так как m было умножено на неизвестное ему число. Противник также не может воспроизвести действия законного получателя сообщения (абонента B), так как ему не известно секретное число c_B (вычисление c_B на основании (2.24) — также задача дискретного логарифмирования). □

Пример 2.16. Передадим сообщение $m = 15$ от A к B. Выберем параметры аналогично тому, как это было сделано в примере 2.2 стр. 24. Возьмем $p = 23$, $g = 5$. Пусть абонент B выбрал для себя секретное число $c_B = 13$ и вычислил по (2.24)

$$d_B = 5^{13} \bmod 23 = 21.$$

Абонент A выбирает случайно число k, например $k = 7$, и вычисляет по (2.25), (2.26):

$$r = 5^7 \bmod 23 = 17, \quad e = 15 \cdot 21^7 \bmod 23 = 15 \cdot 10 \bmod 23 = 12.$$

Теперь A посылает к B зашифрованное сообщение в виде пары чисел $(17, 12)$. B вычисляет по (2.27)

$$m' = 12 \cdot 17^{23-1-13} \bmod 23 = 12 \cdot 17^9 \bmod 23 = 12 \cdot 7 \bmod 23 = 15.$$

Мы видим, что B смог расшифровать переданное сообщение. □

Ясно, что по аналогичной схеме могут передавать сообщения все абоненты в сети. Заметим, что любой абонент, знающий открытый ключ абонента B, может посылать ему сообщения, зашифрованные с помощью открытого ключа d_B. Но только абонент B, и никто другой, может расшифровать эти сообщения, используя известный только ему секретный ключ c_B. Отметим также, что объем шифра в два

раза превышает объем сообщения, но требуется только одна передача данных (при условии, что таблица с открытыми ключами заранее известна всем абонентам).

2.6. Одностороння функция с «лазейкой» и шифр RSA

Названный в честь его разработчиков Ривеста (Ron Rivest), Шамира (Adi Shamir) и Адлемана (Leonard Adleman), этот шифр до сих пор является одним из самых широко используемых.

Мы видели, что шифр Шамира полностью решает задачу обмена сообщениями, закрытыми для прочтения, в случае, когда абоненты могут пользоваться только открытыми линиями связи. Однако при этом сообщение пересылается три раза от одного абонента к другому, что является недостатком. Шифр Эль-Гамаля позволяет решить ту же задачу за одну пересылку данных, но объем передаваемого шифротекста в два раза превышает объем сообщения. Система RSA лишена подобных недостатков. Интересно то, что она базируется на другой односторонней функции, отличной от дискретного логарифма. Кроме того, здесь мы встретимся с еще одним изобретением современной криптографии – *односторонней функцией с «лазейкой»* (trapdoor function).

Эта система 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 мы знаем только n, а p и q — большие числа (это так называемая задача факторизации).

Пусть в нашей системе есть абоненты A, B, C, ... Каждый абонент выбирает случайно два больших простых числа P и Q. Затем он вычисляет число

$$N = PQ. \tag{2.28}$$

(Число N является открытой информацией, доступной другим абонентам.) После этого абонент вычисляет число $\phi = (P-1)(Q-1)$ и выбирает некоторое число $d < \phi$, взаимно простое с ϕ, и по обобщенному алгоритму Евклида находит число c, такое, что

$$cd \bmod \phi = 1. \tag{2.29}$$

Вся информация, связанная с абонентами и являющаяся их открытыми и секретными ключами, представлена в табл. 2.4.

Т а б л и ц а 2.4. **Ключи пользователей в системе RSA**

Абонент	Секретный ключ	Открытый ключ
A	c_A	d_A, N_A
B	c_B	d_B, N_B
C	c_C	d_C, N_C

Опишем протокол RSA. Пусть Алиса хочет передать сообщение m Бобу, причем сообщение m рассматривается как число, удовлетворяющее неравенству $m < N_B$ (далее индекс B указывает на то, что соответствующие параметры принадлежат Бобу).

Шаг 1. Алиса шифрует сообщение по формуле

$$e = m^{d_B} \bmod N_B, \tag{2.30}$$

используя открытые параметры Боба, и пересылает e по открытой линии.

Шаг 2. Боб, получивший зашифрованное сообщение, вычисляет

$$m' = e^{c_B} \bmod N_B. \tag{2.31}$$

Утверждение 2.12. *Для описанного протокола $m' = m$, т.е. абонент B получает исходящее от A сообщение.*

Доказательство. По построению протокола

$$m' = e^{c_B} \bmod N_B = m^{d_B c_B} \bmod N_B.$$

Равенство (2.29) означает, что для некоторого k

$$c_B d_B = k\phi_B + 1.$$

Согласно утверждению 2.5

$$\phi_B = (P_B - 1)(Q_B - 1) = \varphi(N_B),$$

где $\varphi(\cdot)$ — функция Эйлера. Отсюда и из теоремы 2.8 следует

$$m' = m^{k\varphi(N_B)+1} \bmod N_B = m.$$

□

Утверждение 2.13 (свойства протокола RSA).

1) *Протокол шифрует и дешифрует информацию корректно;*

2) *злоумышленник, перехватывающий все сообщения и знающий всю открытую информацию, не сможет найти исходное сообщение при больших P и Q.*

Доказательство. Первое свойство протокола следует из утверждения 2.12. Для доказательства второго свойства заметим, что злоумышленник знает только открытые параметры N и d. Для того чтобы найти c, он должен знать значение $\phi = (P-1)(Q-1)$, а для этого, в свою очередь, ему требуется знать P и Q. Вообще говоря, он может найти P и Q, разложив N на множители, однако это трудная задача (см. пункт 2 в начале раздела). Отметим, что выбор больших случайных P и Q возможен за приемлемое время, так как справедлив пункт 1. □

Односторонняя функция $y = x^d \bmod N$, применяемая в системе RSA, обладает так называемой «лазейкой», позволяющей легко вычислить обратную функцию $x = \sqrt[d]{y} \bmod N$, если известно разложение N на простые множители. (Действительно, легко вычислить $\phi = (P-1)(Q-1)$, а затем $c = d^{-1} \bmod \phi$.) Если P и Q неизвестны, то вычисление значения обратной функции практически невозможно, а найти P и Q по N очень трудно, т.е. знание P и Q — это «лазейка» или «потайной ход»). Такие односторонние функции с лазейкой находят применение и в других разделах криптографии.

Отметим, что для схемы RSA важно, чтобы каждый абонент выбирал собственную пару простых чисел P и Q, т.е. все модули N_A, N_B, N_C, ... должны быть различны (в противном случае один абонент мог бы читать зашифрованные сообщения, предназначенные для другого абонента). Однако этого не требуется от второго открытого параметра d. Параметр d может быть одинаковым у всех абонентов. Часто рекомендуется выбирать $d = 3$ (при соответствующем выборе P и Q, см. [23]). Тогда шифрование выполняется максимально быстро, всего за два умножения.

Пример 2.17. Допустим, Алиса хочет передать Бобу сообщение $m = 15$. Пусть Боб выбрал следующие параметры:

$$P_B = 3,\ Q_B = 11,\ N_B = 33,\ d_B = 3$$

(3 взаимно просто с $\varphi(33) = 20$). Найдем c_B с помощью обобщенного алгоритма Евклида:

$$c_B = 7$$

(проверим: $3 \cdot 7 \bmod 20 = 1$). Кодируем m по формуле (2.30):

$$e = 15^3 \bmod 33 = 15^2 \cdot 15 \bmod 33 = 27 \cdot 15 \bmod 33 = 9.$$

Число 9 Алиса передает Бобу по открытому каналу связи. Только Боб знает $c_B = 7$, поэтому он декодирует принятое сообщение, используя (2.31):

$$m' = 9^7 \bmod 33 = \left(9^2\right)^2 \cdot 9^2 \cdot 9 \bmod 33 = 15^2 \cdot 15 \cdot 9 \bmod 33 = 15.$$

Таким образом, Боб расшифровал сообщение Алисы. □

Рассмотренная система невскрываема при больших P и Q, но обладает следующим недостатком: A передает сообщение B, используя открытую информацию абонента B (числа N_B и d_B). Злоумышленник не может читать сообщения, предназначенные для B, однако он может передать сообщение к B от имени A. Избежать этого можно, используя более сложные протоколы, например, следующий.

A хочет передать B сообщение m. Сначала A вычисляет число $e = m^{c_A} \bmod N_A$. Злоумышленник не может этого сделать, так как c_A секретно. Затем A вычисляет число $f = e^{d_B} \bmod N_B$ и передает f к B. B получает f и вычисляет последовательно числа $u = f^{c_B} \bmod N_B$ и $w = u^{d_A} \bmod N_A$.

В результате абонент B получает сообщение $w = m$. Как и в исходной схеме RSA, злоумышленник не может прочитать переданное сообщение, но здесь, в отличие от RSA, он не может также послать сообщение от имени A (поскольку не знает секретного c_A).

Здесь мы встречаемся с новой ситуацией. B знает, что сообщение пришло от A, т.е. A как бы «подписал» его, зашифровав своим секретным c_A. Это пример так называемой *электронной* или *цифровой*

подписи. Она — одно из широко используемых на практике изобретений современной криптографии и будет систематически изучаться в главе 5.

Задачи и упражнения

2.1. Привести результат выражений 5, 16, 27, -4, -13, $3+8$, $3-8$, $3\cdot 8$, $3\cdot 8\cdot 5$:

а. по модулю 10,

б. по модулю 11.

См. ответ.

2.2. Вычислить, используя быстрые алгоритмы возведения в степень, $2^8 \bmod 10$, $3^7 \bmod 10$, $7^{19} \bmod 100$, $7^{57} \bmod 100$. См. ответ.

2.3. Разложить на простые множители числа 108, 77, 65, 30, 159. См. ответ.

2.4. Определить, какие из пар чисел $(25, 12)$, $(25, 15)$, $(13, 39)$, $(40, 27)$ взаимно просты. См. ответ.

2.5. Найти значения функции Эйлера $\varphi(14)$, $\varphi(20)$. См. ответ.

2.6. Используя свойства функции Эйлера, вычислить $\varphi(53)$, $\varphi(21)$, $\varphi(159)$. См. ответ.

2.7. Используя теорему Ферма, вычислить $3^{13} \bmod 13$, $5^{22} \bmod 11$, $3^{17} \bmod 5$. См. ответ.

2.8. Используя теорему Эйлера, вычислить $3^9 \bmod 20$, $2^{14} \bmod 21$, $2^{107} \bmod 159$. См. ответ.

2.9. С помощью алгоритма Евклида найти $\gcd(21, 12)$, $\gcd(30, 12)$, $\gcd(24, 40)$, $\gcd(33, 16)$. См. ответ.

2.10. С помощью обобщенного алгоритма Евклида найти значения x и y в уравнениях

а. $21x + 12y = \gcd(21, 12)$,

б. $30x + 12y = \gcd(30, 12)$,

в. $24x + 40y = \gcd(24, 40)$,

г. $33x + 16y = \gcd(33, 16)$.

См. ответ.

2.11. Вычислить $3^{-1} \bmod 7$, $5^{-1} \bmod 8$, $3^{-1} \bmod 53$, $10^{-1} \bmod 53$. См. ответ.

2.12. Выписать все простые числа, меньшие 100. Какие из них соответствуют виду $p = 2q + 1$, где q также простое? См. ответ.

2.13. Найти все допустимые варианты выбора параметра g в системе Диффи–Хеллмана при $p = 11$. См. ответ.

2.14. Вычислить открытые ключи Y_A, Y_B и общий ключ Z_{AB} для системы Диффи–Хеллмана с параметрами:

а. $p = 23$, $g = 5$, $X_A = 5$, $X_B = 7$,

б. $p = 19$, $g = 2$, $X_A = 5$, $X_B = 7$,

в. $p = 23$, $g = 7$, $X_A = 3$, $X_B = 4$,

г. $p = 17$, $g = 3$, $X_A = 10$, $X_B = 5$,

д. $p = 19$, $g = 10$, $X_A = 4$, $X_B = 8$.

См. ответ.

2.15. Для шифра Шамира с заданными параметрами p, c_A, c_B найти недостающие параметры и описать процесс передачи сообщения m от A к B:

а. $p = 19$, $c_A = 5$, $c_B = 7$, $m = 4$,

б. $p = 23$, $c_A = 15$, $c_B = 7$, $m = 6$,

в. $p = 19$, $c_A = 11$, $c_B = 5$, $m = 10$,

г. $p = 23$, $c_A = 9$, $c_B = 3$, $m = 17$,

д. $p = 17$, $c_A = 3$, $c_B = 13$, $m = 9$.

См. ответ.

2.16. Для шифра Эль-Гамаля с заданными параметрами p, g, c_B, k найти недостающие параметры и описать процесс передачи сообщения m пользователю B:

а. $p = 19$, $g = 2$, $c_B = 5$, $k = 7$, $m = 5$,

б. $p = 23$, $g = 5$, $c_B = 8$, $k = 10$, $m = 10$,

в. $p = 19$, $g = 2$, $c_B = 11$, $k = 4$, $m = 10$,

г. $p = 23$, $g = 7$, $c_B = 3$, $k = 15$, $m = 5$,

д. $p = 17$, $g = 3$, $c_B = 10$, $k = 5$, $m = 10$.

См. ответ.

2.17. В системе RSA с заданными параметрами P_A, Q_A, d_A найти недостающие параметры и описать процесс передачи сообщения m пользователю A:

а. $P_A = 5$, $Q_A = 11$, $d_A = 3$, $m = 12$,

б. $P_A = 5$, $Q_A = 13$, $d_A = 5$, $m = 20$,

в. $P_A = 7$, $Q_A = 11$, $d_A = 7$, $m = 17$,

г. $P_A = 7$, $Q_A = 13$, $d_A = 5$, $m = 30$,

д. $P_A = 3$, $Q_A = 11$, $d_A = 3$, $m = 15$.

См. ответ.

2.18. Пользователю системы RSA с параметрами $N = 187$, $d = 3$ передано зашифрованное сообщение $e = 100$. Расшифровать это сообщение, взломав систему RSA пользователя. См. ответ.

Глава 3. ТЕОРЕТИЧЕСКАЯ СТОЙКОСТЬ КРИПТОСИСТЕМ

3.1. Введение

По-видимому, одна из первых открытых работ по криптографии появилась в 1949 году и принадлежала Шеннону (Claude Shannon, см. [17]). В ней рассматривается классическая схема криптосистемы с секретным ключом, которая была представлена на рис. 1.1 (стр. 7).

В этой схеме имеется защищенный канал, предназначенный для передачи секретных ключей. Однако отметим, что в настоящее время можно рассмотреть в качестве защищенного канала схему вычисления секретного ключа на основе методов криптографии с открытым ключом, например, схему Диффи–Хеллмана. В дальнейшем мы будем рассматривать только классическую схему с секретным ключом, но многие результаты переносятся на случай создания секретного канала средствами криптографии с открытым ключом.

Все методы шифрования можно грубо разделить на два больших класса:

1) схемы, принципиально не вскрываемые, что доказывается строго;

2) схемы, стойкость которых основана на том, что дешифрование без ключа, вообще говоря, возможно, но для этого требуется перебор очень большого количества вариантов.

В этой главе мы будем заниматься системами первого типа, стой-

кость которых теоретически доказана. Системы второго типа будут рассмотрены в следующей главе.

3.2. Теория систем с совершенной секретностью

Пусть $M = \{M_1, M_2, M_3, \ldots, M_m\}$ — множество всех возможных сообщений (например, множество всех текстов длины не более, чем 1000 букв), $K = \{K_1, K_2, K_3, \ldots, K_n\}$ — множество всех возможных ключей, $E = \{E_1, E_2, \ldots, E_k\}$ — множество всех криптограмм (т.е. зашифрованных сообщений). Зашифрованные сообщения зависят от исходного сообщения и ключа, т.е. $E_j = f(M_i, K_l)$.

Мы будем считать, что на всем множестве сообщений M задано распределение вероятностей P, т.е. определены вероятности $P(M_i)$, $i = 1, 2, \ldots, m$. Это априорное распределение вероятностей, которое известно и противнику. Запись вида $P(A|B)$ будет, как обычно, обозначать условную вероятность события A при условии наступления события B.

Определение 3.1. Криптосистема называется *совершенно секретной*, если выполняется равенство

$$P(M_i|E_j) = P(M_i) \tag{3.1}$$

при всех M_i, K_l и $E_j = f(M_i, K_l)$.

Поясним это определение. Пусть Ева перехватила криптограмму E_j. Если (3.1) выполняется для всех возможных сообщений, то это означает, что она не получила никакой информации о переданном сообщении, т.е. знание E_j совершенно бесполезно для нее. Рассмотрим схематичный пример. Пусть M — множество всех сообщений из шести букв на русском языке . Пусть априори известно, что для

некоторой системы

$$P(\text{сообщение} = \text{«доллар»}) = 0.00015,$$
$$P(\text{сообщение} = \text{«бутыль»}) = 0.0000012 \text{ и т.д.}$$

Допустим, мы имеем несовершенную систему, и Ева после перехвата и вычислений получила следующие данные:

$$P(\text{сообщение} = \text{«доллар»}) = 10^{-20},$$
$$P(\text{сообщение} = \text{«бутыль»}) = 0.9999.$$

Это означает, что Ева практически расшифровала сообщение: она практически уверена, что передано слово «бутыль», так как вероятность, что передано другое сообщение меньше 0.0001.

Если же для рассмотренной системы при любой перехваченной криптограмме E_j мы получаем

$$P(\text{сообщение} = \text{«доллар»}|E_j) = 0.00015,$$
$$P(\text{сообщение} = \text{«бутыль»}|E_j) = 0.0000012$$

и такие же равенства выполняются для всех остальных сообщений, то Ева вообще может не обращать внимание на перехваченный шифротекст E_j, а, например, отгадывать сообщение на основе исходных вероятностей. Другими словами, (3.1) — действительно разумное определение совершенно секретной системы.

Исследуем свойства совершенно секретной системы.

Теорема 3.1. *Если система является совершенно секретной (выполняется* (3.1)*), то справедливо равенство*

$$P\left(E_j|M_i\right) = P\left(E_j\right) \tag{3.2}$$

при всех i и j . Верно и обратное утверждение: если (3.2) *выполняется, то система совершенно секретна.*

Д о к а з а т е л ь с т в о . По определению условной вероятности

$$P\,(A|B) = \frac{P(AB)}{P(B)},$$

при $P(B) \neq 0$ (см., например, [15]). Поэтому при $P(E_j) \neq 0$ можно записать

$$P(M_i|E_j) = \frac{P(M_iE_j)}{P(E_j)} = \frac{P(M_i)P(E_j|M_i)}{P(E_j)}.$$

Принимая во внимание (3.1), получаем

$$P\,(M_i|E_j) = \frac{P(M_i|E_j)P(E_j|M_i)}{P(E_j)},$$

т.е.

$$\frac{P(E_j|M_i)}{P(E_j)} = 1.$$

Таким образом, (3.2) доказано. Обратное утверждение теоремы доказывается «обратным ходом» приведенных равенств, см. [17]. □

3.3. Шифр Вернама

Этот шифр был предложен в 1926 году американским инженером Вернамом (Gilbert Vernam) и использовался на практике, но доказательство его невскрываемости было получено значительно позже Шенноном [17]. Для шифра Вернама часто используется название «одноразовая лента» (one-time pad). Мы опишем этот шифр для случая двоичного алфавита, чтобы упростить обозначения.

Пусть множество сообщений M состоит из слов двоичного алфавита длины n, т.е. всего сообщений не более, чем 2^n. В шифре Вернама множество ключей также состоит из слов той же длины n и каждый ключ используется с вероятностью $1/2^n$. Другими словами, все ключи используются с одинаковой вероятностью.

Пусть необходимо зашифровать сообщение $\bar{m} = m_1 m_2 \dots m_n$ и пусть выбран ключ $\bar{k} = k_1 k_2 \dots k_n$. Тогда зашифрованное сообщение $\bar{e} = e_1 e_2 \dots e_n$ получается по формуле:

$$e_i = m_i \oplus k_i, \tag{3.3}$$

где $i = 1, 2, \dots, n$ и $\oplus$ обозначает сложение по модулю 2. Другими словами, сообщение шифруется по схеме

$$\begin{array}{cccc} & m_1 & m_2 & \dots & m_n \\ \oplus & k_1 & k_2 & \dots & k_n \\ \hline & e_1 & e_2 & \dots & e_n \end{array}.$$

Так как сложение и вычитание по модулю 2 совпадают, то легко видеть, что дешифрование осуществляется по формуле

$$m_i = e_i \oplus k_i. \tag{3.4}$$

Пример 3.1. Пусть $\bar{m} = 01001$, $\bar{k} = 11010$. Тогда получаем $\bar{e} = 10011$. Сложив $\bar{e}$ с $\bar{k}$, восстанавливаем $\bar{m}$. □

Теорема 3.2. *Шифр Вернама является совершенно секретной криптосистемой.*

Доказательство. Согласно теореме 3.1 достаточно доказать справедливость (3.2). Имеем

$$\begin{aligned} P(E_j|M_i) &= P(e^n|m^n) = \\ &= P(k_1 = e_1 \oplus m_1;\ k_2 = e_2 \oplus m_2; \dots; k_n = e_n \oplus m_n) = \\ &= P(\text{ключ} = k_1 \dots k_n) = 2^{-n} \end{aligned}$$

(в последнем равенстве мы использовали то, что, по предположению, все ключи равновероятны).

Найдем $P(E_j)$. По формуле полной вероятности

$$P(E_j) = \sum_{i=1}^{2^n} P(M_i)P(E_j|M_i).$$

Учитывая, что $P(E_j|M_i) = 2^{-n}$, получаем

$$P(E_j) = 2^{-n} \sum_{i=1}^{2^n} P(M_i).$$

Так как сумма вероятностей всех возможных сообщений равна 1, получаем

$$P(E_j) = 2^{-n}.$$

Таким образом, справедливо (3.2). Теорема доказана. □

Известно, что шифр Вернама использовался при защите правительственной связи: например, на так называемой горячей линии «Москва – Вашингтон» [23], а также в других системах, где можно позволить дорогой способ доставки секретного ключа. Однако шифр Вернама можно использовать во многих других практически важных ситуациях. Например, на основе этого шифра легко организовать связь между банком и его филиалами (или связи между банком и клиентами), когда они находятся в одном городе, или защитить электронные письма, скажем, для студентов Алисы и Боба, расстающихся на каникулы (мы предлагаем читателю придумать схему и написать программу, считая, что общая длина писем, которыми они будут обмениваться во время каникул, не превосходит 1.44 Мбайт, т.е. объема стандартной дискеты).

3.4. Элементы теории информации

Мы доказали, что шифр Вернама совершенен, однако при его использовании длина ключа равна длине сообщения. К. Шеннон [17] пока-

зал, что у любой совершенной системы длина ключа должна быть больше энтропии сообщения (с которой мы кратко познакомимся ниже), т.е. пропорциональна его длине. Во многих практически важных ситуациях приходится использовать короткие ключи (скажем, сотни или тысячи бит) для шифрования длинных сообщений (сотни килобайт и более). В этом случае можно построить только так называемые идеальные системы, впервые описанные Шенноном. Для построения идеальных систем и исследования их свойств Шеннон предложил использовать понятия теории информации. В этом разделе мы определим и кратко проиллюстрируем эти понятия. Достаточно полное и строгое их изложение может быть найдено, например, в [4].

Начнем с определения основного понятия — энтропии Шеннона. Пусть дана дискретная случайная величина ξ, принимающая значения $a_1, a_2, \cdots a_r$ с вероятностями $P_1, P_2, \cdots, P_r$.

Определение 3.2. *Энтропия случайной величины* ξ определяется равенством

$$H(\xi) = -\sum_{i=1}^{r} P_i \log P_i, \tag{3.5}$$

где $0 \log 0 = 0$.

Если используется двоичный логарифм, то энтропия измеряется в битах, что общепринято в криптографии, теории информации и в компьютерных науках. В случае натуральных логарифмов единица измерения — нат, в случае десятичных — дит.

При $r = 2$ можно (3.5) записать иначе, введя следующие обозначения: $P_1 = p$, $P_2 = 1 - p$. Тогда

$$H = -\left(p \log p + (1-p) \log (1-p)\right). \tag{3.6}$$

График энтропии для этого случая приведен на рис. 3.1.

Рассмотрим простейшие свойства энтропии.

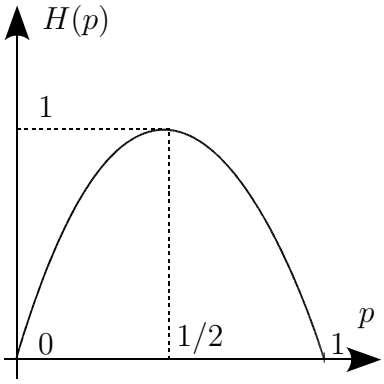

Рис. 3.1. График двоичной энтропии

Утверждение 3.3.

1) $H(\xi) \geq 0$;

2) $H(\xi) \leq \log r$;

3) $H(\xi) = \log r$ *при* $P_i = 1/r,\ i = 1, 2, \ldots, r$.

Доказательство. Первое свойство довольно очевидно (см. (3.5)). Второе свойство докажем только для случая $r = 2$, так как общий случай аналогичен. Исследуем график энтропии. Нужно найти максимум функции (3.6). Для этого мы найдем первую и вторую производные $H(p)$, считая, что логарифм натуральный.

$$H'(p) = -\left(\ln p + p \cdot \frac{1}{p} - \ln(1-p) - \frac{1-p}{1-p}\right) = -\ln p + \ln(1-p).$$

Отсюда $H'(p) = 0$ при $p = 1/2$. Найдем вторую производную

$$H''(p) = -\frac{1}{p} - \frac{1}{1-p}.$$

Мы видим, что $H''(p) < 0$ при $p \in (0;\ 1)$. Это означает, что функция $H(p)$ достигает максимума в точке $1/2$ и выпукла на отрезке $(0;\ 1)$. Таким образом, график, изображенный на рис. 3.1, обоснован. Очевидно, при любом основании логарифма график будет аналогичный.

Для доказательства третьего свойства заметим, что наибольшее значение энтропии для $r = 2$ равно

$$\max H(p) = H(1/2) = -\left((1/2)\log(1/2) + (1/2)\log(1/2)\right) = 1,$$

т.е. составляет один бит в случае двоичного логарифма. □

«Физический» смысл энтропии состоит в том, что энтропия — это количественная мера неопределенности. В качестве примера рассмотрим три случайные величины для $r = 2$. Иными словами, будем считать, что у нас есть три источника сообщений, которые порождают буквы a_1, a_2, т.е. имеются три случайные величины $\xi_i, i = 1, 2, 3$, принимающие значения a_1 или a_2:

$$\begin{array}{lll} \xi_1: & P(a_1) = 1, & P(a_2) = 0; \\ \xi_2: & P(a_1) = 0.5, & P(a_2) = 0.5; \\ \xi_3: & P(a_1) = 0.01, & P(a_2) = 0.99. \end{array}$$

Интуитивно ясно, что неопределенность случайной величины ξ_1 равна нулю. И действительно,

$$H(\xi_1) = -(1 \cdot \log 1 + 0 \cdot \log 0) = 0.$$

Посмотрим на ξ_2 и ξ_3. Интуитивно кажется, что неопределенность у ξ_2 выше неопределенности у ξ_3. Вычислим энтропии:

$$H\left(\xi_2\right) = 1 \text{ бит}$$

(уже считали выше),

$$H\left(\xi_3\right) = -\left(0.01 \cdot \log 0.01 + 0.99 \cdot \log 0.99\right) \approx 0.08 \text{ бит}.$$

Мы видим, что энтропия действительно является разумной мерой неопределенности. Но главное, конечно, не примеры такого типа, а то, что эта величина играет ключевую роль во многих задачах теории передачи и хранения информации. В частности, энтропия характеризует максимальную степень сжатия данных. Точнее, если источник сообщений порождает текст достаточно большой длины n с определенной ниже предельной энтропией h на бит сообщения, то этот текст может быть «сжат» в среднем до величины сколь угодно близкой к nh. Например, если $h = 1/2$, то текст сжимается вдвое и т.п.. Подчеркнем, что речь идет о так называемом неискажающем сжатии, когда по «сжатому» сообщению можно восстановить исходное.

Рассмотрим теперь двумерную случайную величину, заданную рядом распределения

$$P_{ij} = P\left(\xi_1 = a_i,\ \xi_2 = b_j\right), \quad 1 \le i \le r,\ 1 \le j \le s. \tag{3.7}$$

П р и м е р 3.2. Из многолетнего опыта преподавания в некотором вузе известно, что оценки за первый и второй контрольный срок по математике (соответственно ξ_1 и ξ_2) подчиняются закону распределения, задаваемому табл. 3.1. □

Т а б л и ц а 3.1. **Распределение оценок за контрольный срок**

$\xi_1 \downarrow \xi_2 \rightarrow$	0	1	2
0	0.20	0.05	0
1	0.05	0.30	0.05
2	0	0.05	0.30

Введем следующие обозначения:

$$P_{i\cdot} = P(\xi_1 = a_i) = \sum_{j=1}^{s} P_{ij},$$
$$P_{\cdot j} = P(\xi_2 = b_j) = \sum_{i=1}^{r} P_{ij}.$$

Напомним некоторые элементарные соотношения, известные из теории вероятностей (см. [15]):

$$P\left(A|B\right) = \frac{P(AB)}{P(B)}, \tag{3.8}$$

$$P\left(AB\right) = P\left(A\right)P\left(B|A\right) = P\left(B\right)P\left(A|B\right), \tag{3.9}$$

$$P\left(A\right) = \sum_{i} P\left(H_i\right)P\left(A|H_i\right) \tag{3.10}$$

((3.10) — уже встречавшаяся формула полной вероятности, в которой H_i — попарно несовместные события, сумма которых содержит событие A).

В соответствии с (3.5) определим энтропию двумерной случайной величины

$$H(\xi_1, \xi_2) = -\sum_{i=1}^{r}\sum_{j=1}^{s} P_{ij}\log P_{ij}. \tag{3.11}$$

Аналогично для трехмерной случайной величины (ξ_1, ξ_2, ξ_3) и распределения вероятностей P_{ijk} определим

$$H(\xi_1, \xi_2, \xi_3) = -\sum_{i}\sum_{j}\sum_{k} P_{ijk}\log P_{ijk}. \tag{3.12}$$

Подобным же образом определяется энтропия для n-мерной случайной величины.

Представим теперь, что значение случайной величины ξ_1 известно, а ξ_2 — неизвестно. Тогда естественно определить условную энтропию

$$H(\xi_2|\xi_1) = -\sum_{i=1}^{r} P_{i\cdot} \sum_{j=1}^{s} \frac{P_{ij}}{P_{i\cdot}} \log \frac{P_{ij}}{P_{i\cdot}}. \tag{3.13}$$

Это — средняя условная энтропия случайной величины ξ_2 при условии, что значение ξ_1 известно.

Утверждение 3.4 (свойство двумерной энтропии).

$$H(\xi_1, \xi_2) = H(\xi_1) + H(\xi_2|\xi_1), \tag{3.14}$$

в частности, для независимых случайных величин ξ_1 *и* ξ_2

$$\begin{aligned} H(\xi_2|\xi_1) &= H(\xi_2), \\ H(\xi_1, \xi_2) &= H(\xi_1) + H(\xi_2). \end{aligned} \tag{3.15}$$

Напомним, что ξ_1, ξ_2 независимы, если $P_{ij} = P_i P_j$ для всех i и j. Доказательство утверждения достаточно просто и может быть найдено в [4]. Мы ограничимся только его интерпретацией. Пусть в первом опыте порождается ξ_2, во втором — ξ_1. Тогда общая неопределенность экперимента должна быть равна неопределенности первого опыта, сложенной с условной неопределенностью второго опыта. В случае независимых ξ_1 и ξ_2 знание одной величины не несет никакой информации о другой, что соответствует (3.15).

Пусть дана n-мерная случайная величина $(\xi_1, \xi_2, \ldots, \xi_n)$. Справедливо следующее соотношение [4]:

$$H(\xi_1, \ldots, \xi_n) = H(\xi_1) + H(\xi_2|\xi_1) + H(\xi_3|\xi_1, \xi_2) + \ldots + H(\xi_n|\xi_1, \ldots, \xi_{n-1}). \tag{3.16}$$

Для независимых случайных величин

$$H(\xi_1, \ldots, \xi_n) = \sum_{i=1}^{n} H(\xi_i) \tag{3.17}$$

(заметим, что (3.14), (3.15) — частный случай (3.16), (3.17)).

В общем случае

$$H(\xi_k|\xi_1,\ldots,\xi_{k-1}) \le H(\xi_k). \tag{3.18}$$

Рассмотрим последовательность случайных величин $\xi_1, \xi_2, \xi_3, \ldots$ (ξ_i принимают значения в A), которую можно рассматривать как случайный процесс с дискретным временем. Мы будем считать, что этот процесс стационарный, т.е., неформально, вероятностные характеристики для $(\xi_1 \ldots \xi_n)$ те же, что и для $(\xi_{\Delta+1} \ldots \xi_{\Delta+n})$ при всех положительных n и Δ (точное определение дано в [4]).

Пусть $H(\xi_1,\ldots,\xi_n)$ — энтропия n-мерной случайной величины $(\xi_1 \ldots \xi_n)$. Обозначим через

$$h_n^+ = \frac{1}{n} H(\xi_1,\ldots,\xi_n)$$

удельную энтропию n-го порядка и определим

$$h_n^- = H(\xi_n|\xi_1,\ldots,\xi_{n-1}).$$

Отметим следующие свойства:

$$h_n^+ \le h_{n-1}^+, \quad n > 1, \tag{3.19}$$

$$h_n^- \le h_n^+, \tag{3.20}$$

$$h_n^- \le h_{n-1}^-, \quad n > 1. \tag{3.21}$$

Их доказательство может быть найдено в [4].

Для независимых $\xi_1, \xi_2, \ldots, \xi_n$ справедливы равенства

$$h_n^+ = h_n^- = h.$$

(Процесс, порождающий независимые случайные величины, называется процессом без памяти.)

Теорема 3.5. *Для стационарного процесса существуют пределы* $\lim_{n\to\infty} h_n^+$ *и* $\lim_{n\to\infty} h_n^-$*, причем эти пределы равны.*

Доказательство теоремы см. в [4].

Обозначим общее значение этих пределов через h_∞,

$$h_\infty = \lim_{n\to\infty} h_n^+ = \lim_{n\to\infty} h_n^-. \tag{3.22}$$

Пусть дан алфавит $A = (a_1, a_2, \ldots a_r)$. Мы знаем, что

$$\max H(\xi_1) = \log r$$

для процесса без памяти, поэтому, принимая во внимание (3.21) и (3.22), получаем $\max h_\infty = \log r$, причем максимум достигается для процессов без памяти, у которых все буквы порождаются с равными вероятностями $1/r$. Естественно ввести величину

$$R = \log r - h_\infty, \tag{3.23}$$

называемую избыточностью (на букву сообщения). Неформально, это как бы неиспользованная часть алфавита. Избыточность — количественная мера взаимной зависимости символов и их «неравновероятности». Отметим, что в примере из первой главы во втором случае, когда даже простой шифр Цезаря не вскрываем, избыточность шифруемого сообщения равна нулю, так как все десять символов независимы и равновероятны, т.е. $h_\infty = \log 10$ и $R = 0$.

3.5. Расстояние единственности шифра с секретным ключом

Рассмотрим криптосистему с секретным ключом, схема которой показана на рис. 1.1 (стр. 7). Пусть источник порождает сообщение

$\bar{m} = m_1 m_2 \dots m_n$. Например, m_i при каждом i — это буква из русского алфавита или знак пробела, а $\bar{m}$ — сообщение на русском языке. Алиса и Боб обладают секретным ключом k, известным только им, и пусть $\bar{e} = e_1 e_2 \dots e_n$ — сообщение, зашифрованное при помощи этого ключа.

Пример 3.3. Пусть источник порождает буквы из алфавита $A = \{a, b, c\}$ с вероятностями $P(a) = 0.8$, $P(b) = 0.15$, $P(c) = 0.05$, и пусть это источник без памяти. Пусть шифратор, применяя ключ k, заменяет буквы в исходном сообщении, используя какую-либо перестановку символов:

$$\begin{array}{ll} (a,b,c) & k=1 \\ (a,c,b) & k=2 \\ (b,a,c) & k=3 \\ (b,c,a) & k=4 \\ (c,a,b) & k=5 \\ (c,b,a) & k=6, \end{array}$$

т.е. ключ принимает значения от 1 до 6, и если, например, $k = 5$, то в исходном тексте осуществляется следующая замена символов: $a \to c,\ b \to a,\ c \to b$.

Пусть Ева перехватила зашифрованное сообщение

$$\bar{e} = cccbc$$

и хочет определить значение ключа. Оценим количественно вероятности использования всех возможных ключей, используя формулу Байеса

$$P\left(K_i|E\right) = \frac{P(K_i)P(E|K_i)}{\sum_{j=1}^{t} P(K_j)P(E|K_j)},$$

где E, $K_1, \dots, K_t$ — некоторые события, причем K_i попарно несовместны и $E \subset \sum_{i=1}^{t} K_i$. В нашем случае событие E — это получение зашифрованного сообщения $\bar{e} = cccbc$, $t = 6$, а K_i означает, что выбран ключ $k = i$.

Мы предполагаем, что все ключи равновероятны, т.е.

$$P(K_1) = P(K_2) = P(K_3) = P(K_4) = P(K_5) = P(K_6) = 1/6.$$

Тогда

$$\begin{aligned}
P(E|K_1) &= P(\bar{m} = cccbc) = 0.05^4 \cdot 0.15 \approx 0.000001,\\
P(E|K_2) &= P(\bar{m} = bbbcb) = 0.15^4 \cdot 0.05 \approx 0.000025,\\
P(E|K_3) &= P(\bar{m} = cccac) = 0.8 \cdot 0.05^4 \approx 0.000005,\\
P(E|K_4) &= P(\bar{m} = bbbab) = 0.8 \cdot 0.15^4 \approx 0.000405,\\
P(E|K_5) &= P(\bar{m} = aaaca) = 0.8^4 \cdot 0.05 \approx 0.020480,\\
P(E|K_6) &= P(\bar{m} = aaaba) = 0.8^4 \cdot 0.15 \approx 0.061440.
\end{aligned}$$

Отсюда легко находим

$$\sum_{j=1}^{6} P(K_j)\, P(E|K_j) \approx 0.013726,$$

и получаем по формуле Байеса апостериорную вероятность того, что был использован ключ $k = 1$, при условии, что получено сообщение $\bar{e} = cccbc$:

$$P(K_1|E) = P(\bar{m} = cccbc|\bar{e} = cccbc) \approx \frac{(1/6) \cdot 0.000001}{0.013726} \approx 0.000011.$$

Продолжая аналогично, находим, что наиболее вероятны ключи $k = 5$ и $k = 6$:

$$P(K_5|E) = P(\bar{m} = aaaca|\bar{e} = cccbc) \approx 0.25,$$

$$P(K_6|E) = P(\bar{m} = aaaba|\bar{e} = cccbc) \approx 0.75,$$

а вероятности всех остальных ключей меньше 0.01. □

Мы видим, что, перехватив всего 5 букв, Ева может определить ключ почти однозначно. Таким образом, из этого примера и из примера с шифром Цезаря в первой главе мы заключаем, что, по-видимому, существует некоторая длина перехваченного сообщения, после которой ключ может быть найден с вероятностью, близкой к единице.

Утверждение 3.6 (о расстоянии единственности шифра (Шеннон [17])). *Пусть рассматривается система с секретным ключом, и пусть $H(K)$ — энтропия ключа. Пусть R — избыточность шифруемого сообщения, а n — длина сообщения, такая, что*

$$H\left(K|e_1,\ldots,e_n\right) \approx 0, \tag{3.24}$$

т.е. при этой длине перехваченного сообщения ключ почти однозначно восстановлен. Тогда справедливо неравенство

$$n \geq \frac{H(K)}{R}. \tag{3.25}$$

Дадим несколько замечаний к этому утверждению.

1. Число n, удовлетворяющее неравенству (3.25), называется расстоянием единственности шифра. Это означает, что в среднем достаточно перехватить n букв зашифрованного сообщения для восстановления ключа.

2. Мы видим, что если избыточность сообщения $R = 0$, то ключ никогда не будет определен, так как $n = \infty$. Т.е. шифр невозможно взломать (мы видели это в примере с номером замка из первой главы).

3. Уменьшение избыточности может быть достигнуто за счет сжатия данных. Дело в том, что при сжатии данных энтропия «сжатого» текста сохраняется, а длина уменьшается. Следовательно,

энтропия на букву в сжатом тексте больше, чем в исходном, а избыточность меньше, см. (3.23). Значит, после сжимающего кодирования расстояние единственности шифра увеличивается.

4. С практической точки зрения лучше использовать системы, в которых ключ меняется задолго до достижения расстояния единственности шифра.

Доказательство. Мы дадим здесь только основную идею доказательства. Пусть противник, перехватив переданный шифротекст $\bar{e} = e_1 e_2 \dots e_n$, однозначно восстановил ключ, а тем самым и исходное сообщение. Значит, неопределенность противника уменьшилась на $H(K) + H(m_1, \dots, m_n)$, так как он узнал и ключ, и исходное сообщение. При этом он получил n букв из r-буквенного алфавита $\{a_1, \dots, a_r\}$. Мы знаем, что максимальное значение энтропии $h_\infty = \log r$, значит, неопределенность противника не может уменьшаться больше, чем на $n \log r$. Отсюда

$$n \log r \geq H(K) + H(m_1, \dots, m_n),$$

следовательно,

$$n \left(\log r - H(m_1, \dots, m_n)/n\right) \geq H(K),$$

откуда получаем, что

$$n \geq \frac{H(K)}{\log r - H(m_1, \dots, m_n)/n} = \frac{H(K)}{R}$$

(здесь мы воспользовались тем, что $H(m_1, \dots, m_n)/n \to h_\infty$, и определением избыточности (3.23)). □

Пример 3.4. Оценим расстояние единственности для шифра из примера 3.3. Имеем $H(K) = \log 6 \approx 2.58$, $\log r = \log 3 \approx 1.58$, и энтропия на букву источника равна

$$H = -(0.8 \log 0.8 + 0.15 \log 0.15 + 0.05 \log 0.05) \approx 0.88.$$

Поэтому

$$n \geq \frac{2.58}{1.58 - 0.88} \approx 3.7.$$

Мы видели, что пяти букв было практически достаточно для раскрытия ключа. И неравенство (3.25) хорошо согласуется с нашим примером. □

Поясним еще на одном примере, как взаимная зависимость символов увеличивает избыточность и тем самым уменьшает расстояние единственности.

Пример 3.5. Пусть дан марковский источник, т.е. источник с памятью, в котором вероятность появления очередного символа зависит только от предыдущего символа. Источник описывается следующей матрицей переходов:

	a	b	c
a	0	0.9	0.1
b	0	0.1	0.9
c	0.4	0.3	0.3

и начальными вероятностями $P(a) = 0.19$, $P(b) = 0.34$, $P(c) = 0.47$. Пусть, как и в примере 3.3, используется шифр с шестью возможными ключами, и пусть перехвачен зашифрованный текст

$$\bar{e} = bbacbac.$$

Мы видим по матрице переходов, что сочетание aa невозможно (после буквы a вероятность появления снова буквы a равна нулю), а сочетание bb — маловероятно (вероятность появления b после b равна 0.1). Следовательно, скорее всего первая пара букв соответствует буквам cc исходного сообщения, т.е. при шифровании была использована подстановка $c \to b$. Тогда ac соответствует исходным парам ab

или ba. По матрице мы видим, что сочетание ba невозможно, а возможно только ab. Поэтому мы можем предположить, что в качестве ключа была использована перестановка номер 2:

$$k = 2 \quad (a \to a,\ b \to c,\ c \to b).$$

Вычислим точные вероятности использования различных ключей, как в примере 3.3. Заметим, что вероятность конкретного сообщения источника равна произведению вероятности начальной буквы и вероятностей переходов от одной буквы к другой.

$$\begin{aligned}
P(E|K_1) = P(\bar{m} = bbacbac) &= 0.34 \cdot 0.1 \cdot 0 = 0,\\
P(E|K_2) = P(\bar{m} = ccabcab) &= 0.47 \cdot 0.3 \cdot 0.4 \cdot 0.9 \cdot 0.9 \cdot 0.4 \cdot 0.9 =\\
&= 0.016446,\\
P(E|K_3) = P(\bar{m} = aabcabc) &= 0.19 \cdot 0 = 0,\\
P(E|K_4) = P(\bar{m} = aacbacb) &= 0.19 \cdot 0 = 0,\\
P(E|K_5) = P(\bar{m} = ccbacba) &= 0.47 \cdot 0.3 \cdot 0.3 \cdot 0 = 0,\\
P(E|K_6) = P(\bar{m} = bbcabca) &= 0.34 \cdot 0.1 \cdot 0.9 \cdot 0.4 \cdot 0.9 \cdot 0.9 \cdot 0.4 =\\
&= 0.003966.
\end{aligned}$$

Отсюда находим

$$\sum_{j=1}^{6} P\left(K_j\right) P\left(E|K_j\right) \approx 0.003402,$$

и получаем по формуле Бейеса апостериорные вероятности исполь-

зованных ключей при условии, что получено сообщение $\bar{e} = bbacbac$:

$$
\begin{aligned}
&P\left(K_1|E\right) = 0,\\
&P\left(K_2|E\right) = P(\bar{m} = ccabcab|\bar{e} = bbacbac) \approx 0.8,\\
&P\left(K_3|E\right) = 0,\\
&P\left(K_4|E\right) = 0,\\
&P\left(K_5|E\right) = 0,\\
&P\left(K_6|E\right) = P(\bar{m} = bbcabca|\bar{e} = bbacbac) \approx 0.2.
\end{aligned}
$$

Эти вычисления подтверждают справедливость приведенного ранее неформального рассуждения. □

Оценку расстояния единственности шифра можно использовать при конструировании криптосистем. Например, кажется разумным менять ключ на новый, когда общая длина зашифрованных с его помощью сообщений приближается к величине расстояния единственности.

Новые подходы к построению теоретически стойких криптосистем, связанные с применением методов специального кодирования, изложены в работах [10, 13, 14, 16, 24, 25] авторов данной книги. Предлагаемые там методы достаточно сложны для описания, однако они эффективны с вычислительной точки зрения и позволяют строить невскрываемые шифры с секретным ключом. Основная идея этих методов состоит в обеспечении путем кодирования нулевой избыточности сообщения, которое подлежит шифрованию. Один из таких методов будет рассмотрен в следующем разделе.

3.6. Идеальные криптосистемы

В разд. 3.2 было введено понятие совершенной секретности, а затем было показано, что шифр Вернама совершенно секретен. Мы видели, что в этом шифре длина ключа равна длине сообщения и ключ

используется всего один раз. Если же мы хотим использовать короткий многоразовый ключ (что актуально для большинства практических приложений), то какой наилучший результат в смысле стойкости шифра мы можем достичь? При обсуждении утверждения 3.6 указывалось, что при нулевой избыточности сообщения расстояние единственности шифра бесконечно. Это означает, что даже короткий (или, что эквивалентно, применяемый много раз) ключ, используемый для шифрования очень длинного сообщения нулевой избыточности, не может быть раскрыт. А это, в свою очередь, означает, что у противника, пытающегося разгадать зашифрованный текст, остается неопределенность, равная неопределенности ключа. Очевидно, это лучшее, что может быть достигнуто в данных условиях (здесь можно снова вспомнить пример с кодовым замком из первой главы). Эти рассуждения подводят нас к понятию строго идеального шифра, впервые введенному Шенноном [17].

Пусть сообщение $m_1 m_2 \dots m_t$ шифруется при помощи секретного ключа $\bar{k} = k_1 k_2 \dots k_s$, в результате чего получается зашифрованное сообщение $\bar{e} = e_1 e_2 \dots e_t$. Обозначим через $H(m_1 m_2 \dots m_t)$ энтропию сообщения, через $H(\bar{e})$ и $H(\bar{k})$ — соответственно энтропии шифротекста и ключа. Тогда $H(m_1 m_2 \dots m_t|\bar{e})$ представляет неопределенность сообщения, а $H(\bar{k}|\bar{e})$ — неопределенность ключа при условии, что известен шифротекст $\bar{e}$.

Определение 3.3. Шифр называется *строго идеальным*, если

$$H(m_1 m_2 \dots m_t|\bar{e}) = H(\bar{k}|\bar{e}) = \min\{H(m_1 m_2 \dots m_t), H(\bar{k})\}. \quad (3.26)$$

Если энтропия ключа меньше энтропии сообщения источника, то (3.26) упрощается:

$$H(m_1 m_2 \dots m_t|\bar{e}) = H(\bar{k}|\bar{e}) = H(\bar{k}) \quad (3.27)$$

при всех достаточно больших t. Так как мы будем рассматривать случай, когда длина сообщения t велика, то в качестве определения строго идеального шифра будем использовать (3.27) .

Неформально, строгая идеальность шифра означает, что количество решений криптограммы равно количеству различных ключей и все решения равновероятны, как в примере с кодовым замком.

В этом разделе мы рассмотрим конструкцию идеальной системы, недавно предложенную в [10], ограничившись описанием только основной идеи применительно к случаю, когда сообщение порождается двоичным источником без памяти с неизвестной статистикой, иными словами, делается последовательно и независимо случайный выбор буквы из алфавита $A = \{a_1, a_2\}$, причем вероятности выбора букв неизвестны.

Пусть источник порождает потенциально неограниченные сообщения $m_1 m_2 \dots m_t$, $t \to \infty$, и имеется ключ фиксированной длины $\bar{k} = k_1 k_2 \dots k_s$, $s \geq 1$. (Как мы упомянули выше, предполагается также, что энтропия источника на букву отлична от нуля, так как в противном случае вообще нет необходимости в передаче сообщений.) Будем последовательно разбивать сообщение источника на блоки символов длины n, где $n > 1$ — параметр метода. Обозначим один из таких блоков через $\bar{m}$. Опишем преобразования, выполняемые над каждым таким n-буквенным блоком.

Вначале определим количество букв a_1 и a_2 в блоке $\bar{m}$. Пусть, скажем, имеется n_1 букв a_1 и $n_2 = n - n_1$ букв a_2. Определим $u(\bar{m})$ как слово длины $\lceil \log(n+1) \rceil$ бит, кодирующее n_1.

Теперь рассмотрим множество S всех последовательностей, состоящих из n_1 букв a_1 и n_2 букв a_2. В этом множестве

$$|S| = C_n^{n_1} = \frac{n!}{n_1!(n-n_1)!}$$

элементов. Несмотря на то, что нам не известны вероятности последовательностей из множества S, одно мы можем сказать точно — все они равны между собой (в силу независимости выбора отдельных букв сообщения). Зададим на множестве S некоторый порядок, например, расположим сообщения в порядке возрастания соответству-

ющих им двоичных чисел (считая, что $a_1 = 0$, $a_2 = 1$). Вычислим номер данного конкретного блока $\bar{m}$ внутри упорядоченного множества S (для вычисления такого номера известен эффективный алгоритм [11], описание которого выходит за рамки нашей книги). Обозначим этот номер через $\omega(\bar{m})$.

Разобьем множество S на непересекающиеся подмножества S_0, $S_1, \ldots, S_\nu$ с числами элементов, равными различным степеням двойки (например, если $|S| = 21$, то получаем три подмножества с числами элементов 16, 4 и 1). По $\omega(\bar{m})$ определим, какому подмножеству принадлежит $\bar{m}$ (обозначим номер такого подмножества через $v(\bar{m})$), и найдем номер $\bar{m}$ внутри данного подмножества (обозначим этот номер через $w(\bar{m})$).

Посмотрим внимательно на номер сообщения внутри подмножества, $w(\bar{m})$. Замечательно то, что $w(\bar{m})$ — это полностью случайная последовательность нулей и единиц (т.е. такая, где символы независимы, а вероятности нуля и единицы равны). Действительно, $w(\bar{m})$ — это номер одной из равновероятных последовательностей букв в подмножестве из 2^b элементов (для некоторого b). Номера всех таких последовательностей — это всевозможные последовательности из b двоичных цифр. Но если все последовательности из b двоичных цифр равновероятны, то отдельные символы равновероятны и независимы.

Итак, обрабатывая описанным образом последовательные блоки сообщения, мы представляем сообщение в виде

$$u(\bar{m}_1)v(\bar{m}_1)w(\bar{m}_1)u(\bar{m}_2)v(\bar{m}_2)w(\bar{m}_2)\ldots\ .$$

Теперь перейдем к описанию процесса шифрования преобразованного сообщения. На первый взгляд это может показаться странным, но слова $u(\cdot)$ и $v(\cdot)$ вообще не шифруются! Шифруются только слова $w(\cdot)$ с использованием секретного ключа $\bar{k}$. В качестве шифра можно, например, использовать побитовое сложение по модулю

2 с периодически продолженным ключом. Для описания этого шифра удобно занумеровать символы слов $w(\cdot)$ подряд и обозначить их через $w_1w_2w_3\ldots$. Тогда шифрование проводится по формуле

$$z_i = w_i \oplus k_{i \bmod s}.$$

В результате применения описанного метода мы зашифровали сообщение следующим образом:

$$m_1m_2\ldots m_t \longrightarrow \bar{e} = u(\bar{m}_1)v(\bar{m}_1)z(\bar{m}_1)u(\bar{m}_2)v(\bar{m}_2)z(\bar{m}_2)\ldots\ . \tag{3.28}$$

По построению алгоритма ясно, что из правой части (3.28) можно восстановить сообщение, если знать $\bar{k}$. Вначале нужно дешифровать символы слов $w(\cdot)$, используя формулу

$$w_i = z_i \oplus k_{i \bmod s}, \tag{3.29}$$

а затем из слов $u(\cdot)v(\cdot)w(\cdot)$ восстанавливаются последовательные блоки сообщения.

Пример 3.6. Пусть требуется зашифровать сообщение

$$a_2a_2a_1a_2a_2a_2a_1a_2a_2a_1$$

с трехбитовым ключом $\bar{k} = 011$. Разобьем сообщение на два блока по пять символов, $n = 5$.

Выполним преобразование для первого блока $\bar{m}_1 = a_2a_2a_1a_2a_2$. Для этого блока $n_1 = 1$ и $u(\bar{m}_1) = (001)_2$. Рассмотрим теперь упорядоченное множество всех сообщений, состоящих из одной буквы a_1 и четырех букв a_2 (табл. 3.2). Всего таких сообщений $\frac{5!}{1!4!} = 5$, поэтому имеем два подмножества S_0 и S_1 с числом элементов соответственно 4 и 1. Мы видим, что $\bar{m}_1$ входит в S_0 под номером $2 = (10)_2$. Таким образом, мы получаем следующие два слова: $v(\bar{m}_1) = 0$, $w(\bar{m}_1) = (10)_2$.

Таблица 3.2. **Множество равновероятных сообщений;** $n_1 = 1$**,** $n_2 = 4$

Сообщение	Номер в S	S_v	w
$a_1a_2a_2a_2a_2$	000		00
$a_2a_1a_2a_2a_2$	001		01
$a_2a_2a_1a_2a_2$	010	S_0	10
$a_2a_2a_2a_1a_2$	011		11
$a_2a_2a_2a_2a_1$	100	S_1	-

Теперь выполним преобразование для второго блока сообщения $\bar{m}_2 = a_2a_1a_2a_2a_1$. Для этого блока $n_1 = 2$ и $u(\bar{m}_2) = (010)_2$. Рассмотрим упорядоченное множество всех сообщений, состоящих из двух букв a_1 и трех букв a_2 (табл. 3.3). Всего таких сообщений $\frac{5!}{2!3!} = 10$, поэтому имеем два подмножества S_0 и S_1 с числом элементов соответственно 8 и 2. Мы видим, что $\bar{m}_2$ входит под номером $6 = (110)_2$ в S_0. Таким образом, мы получаем $v(\bar{m}_2) = 0$, $w(\bar{m}_2) = (110)_2$.

В результате мы получили следующий двоичный код преобразованного сообщения:

$$001\ 0\ 10\ 010\ 0\ 110$$

(пробелы здесь поставлены только для удобства чтения; для однозначного декодирования они не нужны).

Теперь зашифруем преобразованное сообщение. Периодически продолженный ключ имеет вид $\bar{k} = 011011\ldots$. Сложение битов слов $w(\cdot)$ с этим ключом дает

$$\begin{array}{r} 1\ 0\ 1\ 1\ 0 \\ \oplus\ 0\ 1\ 1\ 0\ 1 \\ \hline 1\ 1\ 0\ 1\ 1 \end{array}.$$

Т а б л и ц а 3.3. **Множество равновероятных сообщений;** $n_1 = 2$**,** $n_2 = 3$

Сообщение	Номер в S	S_v	w
$a_1a_1a_2a_2a_2$	0000		000
$a_1a_2a_1a_2a_2$	0001		001
$a_1a_2a_2a_1a_2$	0010		010
$a_1a_2a_2a_2a_1$	0011	S_0	011
$a_2a_1a_1a_2a_2$	0100		100
$a_2a_1a_2a_1a_2$	0101		101
$a_2a_1a_2a_2a_1$	0110		110
$a_2a_2a_1a_1a_2$	0111		111
$a_2a_2a_1a_2a_1$	1000	S_1	0
$a_2a_2a_2a_1a_1$	1001		1

Зашифрованное сообщение выглядит следующим образом:

$$\bar{e} = 001\ 0\ 11\ 010\ 0\ 011. \qquad \square$$

Остановимся теперь на основных свойствах рассмотренного метода.

Утверждение 3.7. *Построенный шифр строго идеален.*

Д о к а з а т е л ь с т в о . Как уже отмечалось при изложении метода, слово $w(\bar{m})$ для каждого блока $\bar{m}$ состоит из равновероятных и независимых символов 0 и 1, другими словами, «полностью» случайно. Так как блоки в сообщении независимы, то в преобразованном сообщении последовательность слов $w(\bar{m}_1)w(\bar{m}_2)\ldots = w_1w_2w_3\ldots$ также полностью случайна. Но любая последовательность $w_1w_2w_3\ldots$ соответствует некоторому сообщению, и все такие сообщения равновероятны. Поэтому при подстановке любого ключа в дешифрующее

выражение (3.29) мы получаем какое-либо решение, причем все решения равновероятны. В результате, имея только шифротекст $\bar{e}$, мы ничего не можем сказать об использованном ключе, т.е.

$$H(\bar{k}|\bar{e}) = H(\bar{k}).$$

Далее, каждому конкретному сообщению $m_1 m_2 \dots m_t$ соответствует одна и только одна последовательность $w_1 w_2 w_3 \dots$, и при достаточно большом t, а именно, таком, что длина последовательности $w_1 w_2 w_3 \dots$ не меньше длины ключа, каждому ключу, подставляемому в (3.29), соответствуют различные равновероятные сообщения. Поэтому

$$H(m_1 m_2 \dots m_t|\bar{e}) = H(\bar{k}).$$

Ненулевая энтропия источника гарантирует то, что требуемое достаточно большое t всегда существует.

Таким образом, мы доказали, что (3.27) выполняется. □

Особенностью предложенного шифра является то, что шифрованию подвергается не все преобразованное сообщение, а только его часть. В приведенном примере даже может показаться, что слишком много информации остается «открытой». Какая все-таки доля информации скрывается этим шифром? Следующее утверждение дает ответ на этот вопрос.

Утверждение 3.8. *Пусть сообщение порождается источником без памяти с энтропией h на букву . Тогда для каждого блока $\bar{m}$ из n символов сообщения средняя длина шифруемого слова $w(\bar{m})$ удовлетворяет неравенству*

$$E(|w(\bar{m})|) > nh - 2\log(n+1) \tag{3.30}$$

(здесь $E(\cdot)$ — математическое ожидание, а $|\cdot|$ — длина).

Д о к а з а т е л ь с т в о . Компонента кода $u(\bar{m})$ может принимать любое значение от 0 до n, и поэтому ее максимальная энтропия равна $\log(n+1)$.

Слово $v(\bar{m})$ может прининять любое значение от 0 до ν, что связано с разбиением S на подмножества S_0, S_1, …, S_ν. Очевидно, что $\nu \le \lfloor \log|S| \rfloor$. Из известного неравенства $|S| = C_n^{n_1} < 2^n$ получаем $\log|S| < n$ и $\nu \le \log|S| < n$. Поэтому максимальная энтропия слова $v(\bar{m})$ строго меньше $\log(n+1)$.

Энтропия блока равна $H(\bar{m}) = nh$ (так как символы порождаются источником без памяти). При преобразовании блока энтропия не изменяется, поэтому для энтропии слова $w(\bar{m})$ имеем

$$H(w(\bar{m})) > nh - 2\log(n+1)$$

(из общей энтропии мы вычли верхнюю границу максимальной энтропии слов $u(\bar{m})$ и $v(\bar{m})$). Но так как слово $w(\bar{m})$ состоит из равновероятных и независимых символов 0 и 1, его средняя длина совпадает с энтропией, что завершает доказательство. □

Неформально, утверждение 3.8 говорит о том, что «почти вся» информация сообщения содержится в шифруемой компоненте кода w, если длина блока n достаточно велика. Иными словами, представленный шифр скрывает «почти всю» информацию. Причем даже полный перебор ключей не позволяет вскрыть шифр.

Конечно, рассмотренная конструкция идеальной криптосистемы может иметь различные варианты построения. Например, может представлять интерес вариант, в котором часть ключа используется для «закрытия» префикса (т.е. компонент u и v кода). Правда, в этом случае система будет «просто» идеальной (не строго идеальной).

Задачи и упражнения

3.1. Зашифровать сообщение $\bar{m}$ шифром Вернама с ключом $\bar{k}$:

а. $\bar{m} = 1001101011$, $\bar{k} = 0110100101$,

б. $\bar{m} = 0011101001$, $\bar{k} = 1100011100$,

в. $\bar{m} = 1000011100$, $\bar{k} = 1001011010$,

г. $\bar{m} = 0011100010$, $\bar{k} = 0110111001$,

д. $\bar{m} = 1001101011$, $\bar{k} = 1000111010$.

См. ответ.

3.2. Пусть источник без памяти порождает буквы из алфавита $A = \{a, b, c\}$ с вероятностями $P(a)$, $P(b)$, $P(c)$. Шифратор заменяет буквы, используя одну из шести возможных перестановок, как это делалось в примере 3.3. Определить апостериорные вероятности использованных ключей для заданного зашифрованного сообщения $\bar{e}$:

а. $P(a) = 0.1$, $P(b) = 0.7$, $P(c) = 0.2$, $\bar{e} = abaacac$,

б. $P(a) = 0.9$, $P(b) = 0.09$, $P(c) = 0.01$, $\bar{e} = cbaccca$,

в. $P(a) = 0.14$, $P(b) = 0.06$, $P(c) = 0.8$, $\bar{e} = bbabbcab$,

г. $P(a) = 0.7$, $P(b) = 0.05$, $P(c) = 0.25$, $\bar{e} = cccacbbc$,

д. $P(a) = 0.1$, $P(b) = 0.7$, $P(c) = 0.2$, $\bar{e} = abbbbab$.

См. ответ.

3.3. Для источников задачи 3.2 вычислить энтропию и расстояние единственности. См. ответ.

3.4. По имеющемуся зашифрованному сообщению $\bar{e}$ найти апостериорные вероятности использованных ключей и соответствующие им сообщения, если известно, что используется шифр примера 3.3, а сообщения порождаются марковским источником, описанным в примере 3.5:

а. $\bar{e} = bcacbcacc$,

б. $\bar{e} = caaabaaba$,

в. $\bar{e} = aacabcaac$,

г. $\bar{e} = bcaaaacaa$,

д. $\bar{e} = aaacaaaca$.

См. ответ.

Глава 4. СОВРЕМЕННЫЕ ШИФРЫ С СЕКРЕТНЫМ КЛЮЧОМ

4.1. Введение

В этой главе мы рассмотрим вычислительно стойкие шифры с секретным ключом, которые, в принципе, могут быть вскрыты, но требуют для этого очень большого количества вычислений, скажем, 10^{20} лет для суперкомпьютера. Эти шифры обеспечивают шифрование и дешифрование данных со скоростями, значительно превышающими скорости шифров с открытыми ключами и теоретически стойких шифров, что и объясняет их широкое практическое использование. В последующих разделах мы опишем некоторые наиболее популярные шифры и режимы их функционирования, однако вначале, чтобы пояснить принципы построения этих шифров, продолжим пример с шифром Цезаря, начатый в первой главе.

Там мы рассмотрели атаку по шифротексту на шифр Цезаря. Было показано, что в случае избыточных сообщений шифр легко вскрывается путем перебора ключей. Поищем возможности повышения стойкости шифра Цезаря. Пожалуй, самое простое, что приходит в голову — увеличить количество возможных значений ключа. В этом случае Еве придется перебирать больше ключей, прежде чем она найдет единственный правильный.

Естественный способ увеличить количество возможных значений ключа для шифра Цезаря — использовать разные ключи для разных букв сообщения. Например, мы можем шифровать каждую нечетную букву ключом k_1, а четную ключом k_2. Тогда секретный ключ $k = (k_1, k_2)$ будет состоять из двух чисел, и количество возможных

ключей будет $32^2 = 1024$. Зашифруем наше прежнее сообщение из первой главы ключом $k = (3, 5)$:

$$\text{ПЕРЕМЕНА} \xrightarrow{3,5} \text{ТКУКПКРЕ}. \tag{4.1}$$

Эта схема легко обобщается на произвольную длину секретного ключа $k = (k_1, k_2, \cdots, k_t)_{32}$. При t порядка 10 и выше задача полного перебора ключей становится практически нерешаемой.

Тем не менее, данный шифр легко вскрывается путем так называемого частотного криптоанализа. Для этого используется статистика языка, на котором написано передаваемое сообщение. При использовании частотного криптоанализа перебор начинают с ключей, соответствующих наиболее часто встречающимся буквам и их сочетаниям. Например, известно, что в типичном тексте на русском языке буква О встречается чаще других. Смотрим на ТКУКПКРЕ в (4.1) и определяем, какие буквы встречаются чаще других на четных и нечетных местах. На четных местах это К. Предполагаем, что это код О, следовательно, $k_2 = \text{К} - \text{О} = 28 \pmod{32}$. На нечетных местах все буквы различны, поэтому для поиска k_1 знание частот букв языка не помогает (дело в том, что для нашего примера взято очень короткое сообщение). Пытаемся, как и прежде, найти k_1 путем перебора, но убеждаемся, что приемлемых расшифровок не получается. Это означает, что наша гипотеза о том, что О заменяется в шифре на К, неверна. Берем вместо О другую часто встречающуюся букву — букву Е. Вычисляем $k_2 = \text{К} - \text{Е} = 5$. Повторяем аналогичные действия для поиска k_1 и на этот раз находим решение $k_1 = 3$. В результате, чтобы расшифровать сообщение, из всех возможных 1024 ключей нам понадобилось проверить только 36 (мы проверяли $(0, 28), \ldots, (31, 28), (0, 5), \ldots, (3, 5)$).

Попробуем слегка усложнить шифр, чтобы затруднить частотный криптоанализ. Нам нужно каким-то образом перемешать символы сообщения, заставить их влиять друг на друга, чтобы скрыть индиви-

дуальные частоты их появления. По-прежнему будем использовать ключ $k = (k_1, k_2)$ и шифровать сообщение блоками по два символа m_i, m_{i+1}. Один из простейших вариантов шифра может выглядеть так:

$$\begin{aligned} \tilde{m}_i &= m_i + m_{i+1}, \\ \tilde{m}_{i+1} &= m_{i+1} + \tilde{m}_i, \\ c_i &= \tilde{m}_i + k_1, \\ c_{i+1} &= \tilde{m}_{i+1} + k_2 \qquad (\mathrm{mod}\ 32) \end{aligned} \tag{4.2}$$

(все суммы вычисляются по модулю 32). Здесь m_i — нечетная буква исходного текста, m_{i+1} — четная буква, k_1, k_2 — символы ключа, а c_i, c_{i+1} — получаемые символы шифротекста. Например, пара символов ПЕ шифруется ключом $k = (3, 5)$ следующим образом:

$$\begin{aligned} \tilde{m}_i &= \text{П} + \text{Е} = \text{Ф}, \\ \tilde{m}_{i+1} &= \text{Е} + \text{Ф} = \text{Щ}, \\ c_i &= \text{Ф} + 3 = \text{Ч}, \\ c_{i+1} &= \text{Щ} + 5 = \text{Ю}, \end{aligned}$$

т.е. ПЕ превращается в ЧЮ.

Отметим, что этот шифр можно дешифровать. Алгоритм дешифрования, называемый обычно обратным шифром, выглядит следующим образом:

$$\begin{aligned} \tilde{m}_{i+1} &= c_{i+1} - k_2, \\ \tilde{m}_i &= c_i - k_1, \\ m_{i+1} &= \tilde{m}_{i+1} - \tilde{m}_i, \\ m_i &= \tilde{m}_i - m_{i+1} \qquad (\mathrm{mod}\ 32). \end{aligned} \tag{4.3}$$

Применяя к нашему сообщению шифр (4.2) с ключом $(3, 5)$, получаем

$$\text{ПЕРЕМЕНА} \longrightarrow \text{ФЩХЪСЦНН} \xrightarrow{3,5} \text{ЧЮШЯФЫРТ}. \tag{4.4}$$

Здесь для наглядности после первой стрелки показан промежуточный результат, получающийся после выполнения первых двух операций в (4.2) (это «перемешанный», но еще не зашифрованный текст сообщения). Мы видим, что данный шифр скрывает частоты появления отдельных символов, затрудняя частотный анализ. Конечно, он сохраняет частоты появления пар символов, но мы можем скрыть и их, если будем шифровать сообщения блоками по три символа и т.д.

Вообще, шифр (4.2) выглядит более сложным для Евы по сравнению с шифром (4.1), и он дает нам возможность рассмотреть еще одну ситуацию, связанную с ее действиями. До сих пор мы рассматривали атаки только по шифротексту. Но что произойдет, если Ева каким-то образом достала открытый текст, соответствующий ранее переданному зашифрованному сообщению? (Т.е. мы находимся в условиях атаки второго типа, см. главу 1, стр. 10.) Например, Ева имеет пару (ПЕРЕМЕНА, ТКУКПКРЕ) для шифра (4.1). Тогда она сразу вычисляет секретный ключ, $k_1 = \text{Т} - \text{П} = 3$, $k_2 = \text{К} - \text{Е} = 5$, и легко расшифровывает все последующие сообщения от Алисы к Бобу. При использовании шифра (4.2) пара (ПЕРЕМЕНА, ЧЮШЯФЫРТ) уже не дает такого очевидного решения, хотя и здесь оно довольно просто. Ева применяет две первые операции из (4.2) (не требующие секретного ключа) к слову ПЕРЕМЕНА, получает промежуточный результат ФЩХЪСЦНН и уже по паре (ФЩХЪСЦНН, ЧЮШЯФЫРТ), как и в первом случае, находит ключ $k = 3, 5$.

Как затруднить такие действия Евы? Идея проста. Будем при шифровании сообщения использовать шифр (4.2) два раза. Тогда получим:

$$\text{ПЕРЕМЕНА} \xrightarrow{3,5} \text{ЧЮШЯФЫРТ} \xrightarrow{3,5} \text{ШШЪЫТПЕЩ}. \qquad (4.5)$$

Теперь, имея пару (ПЕРЕМЕНА, ШШЪЫТПЕЩ), Ева не может вычислить ключ, по крайней мере алгоритм ее действий не очевиден (она не может получить промежуточное значение ЧЮШЯФЫРТ, так как при его построении был использован секретный ключ, ей не известный).

В представленной схеме (4.5) отдельная реализация алгоритма (4.2) называется *раундом* или *циклом* шифра.

Мы проиллюстрировали влияние на стойкость шифра таких параметров, как длина ключа, размер блока, количество раундов, а также показали необходимость введения «перемешивающих» преобразований. В реальных шифрах используются, в принципе, те же преобразования, но над более длинными цепочками символов и обладающие целым рядом дополнительных свойств. Это связано с наличием развитых методов криптоанализа, таких, как дифференциальный и линейный криптоанализ, описание которых выходит за рамки нашей книги, но может быть найдено в [27].

4.2. Блоковые шифры

Блоковый шифр можно определить как зависящее от ключа K обратимое преобразование слова X из n двоичных символов. Преобразованное с помощью шифра слово (или блок) будем обозначать через Y. Для всех рассматриваемых в этом разделе шифров длина слова Y равна длине слова X.

Таким образом, блоковый шифр — это обратимая функция E (другим словами, такая, для которой существует обратная функция). Конкретный вид E_K этой функции определяется ключом K,

$$\begin{aligned} Y &= E_K(X), \\ X &= E_K^{-1}(Y) \quad \text{для всех } X. \end{aligned}$$

Здесь E_K^{-1} обозначает дешифрующее преобразование и называется *обратным шифром*.

Для криптографических приложений блоковый шифр должен удовлетворять ряду требований, зависящих от ситуации, в которой он используется. В большинстве случаев достаточно потребовать, чтобы

шифр был стоек по отношению к атаке по выбранному тексту. Это автоматически подразумевает его стойкость по отношению к атакам по шифротексту и по известному тексту. Следует заметить, что при атаке по выбранному тексту шифр всегда может быть взломан путем перебора ключей. Поэтому требование стойкости шифра можно уточнить следующим образом.

Шифр *стоек* (при атаке по выбранному тексту), если для него *не существуют* алгоритмы взлома, существенно более быстрые, чем прямой перебор ключей.

Нам будет достаточно такого нестрогого определения стойкости. На самом деле, по состоянию на сегодняшний день, ни для одного используемого шифра не доказано соответствие этому определению стойкости. Реально можно говорить о следующем.

Шифр *считается стойким* (при атаке по выбранному тексту), если для него *неизвестны* алгоритмы взлома, существенно более эффективные, чем прямой перебор ключей.

Ниже мы приведем примеры некоторых практичеки используемых блоковых шифров. Наша задача будет состоять не только в том, чтобы дать достаточно подробное описание алгоритмов (их описание может быть найдено в литературе), но и в объяснении основных принципов построения блоковых шифров. Кроме того, наше описание может облегчить понимание материала, изложенного в официальных документах (ГОСТах и.т.п.). Далее, на протяжении всей главы, мы будем изучать технику использования блоковых шифров для решения различных задач криптографии.

До недавнего времени ни одна книга по криптографии не обходилась без описания шифра DES (Data Encryption Standard). Этот шифр был принят в качестве стандарта США в 1977 году. Его основные параметры: размер блока 64 бита, длина ключа 56 бит, 16 раундов. Этот шифр интенсивно использовался более двух десятков лет и еще сегодня встречается во многих работающих системах. Несмотря на многочисленные атаки против DES, он так и не был взломан. Однако

высокий уровень развития вычислительных средств позволяет сегодня вскрывать DES путем перебора ключей. Например, еще в 1993 году было опубликовано техническое описание системы стоимостью в один миллион долларов, позволяющей взламывать любой ключ DES за 7 часов. В результате DES не рекомендуется использовать во вновь создаваемых криптографических системах, и поэтому мы не описываем этот шифр. В 2001 году после специально объявленного конкурса в США был принят новый стандарт на блоковый шифр, названный AES (Advanced Encryption Standard), в основу которого был положен шифр Rijndael, разработанный бельгийскими специалистами.

Большинство современных блоковых шифров строятся по схемам, значительно отличающимся от DES. Тем не менее есть один действующий шифр, построенный на тех же принципах, что и DES, и представляющий для нас особый интерес. Это российский блоковый шифр ГОСТ 28147-89.

Шифр ГОСТ 28147-89

Шифр ГОСТ 28147-89 [5], как следует из его обозначения, был принят в качестве стандарта в 1989 году. Основные параметры ГОСТ 28147-89: длина ключа 256 бит, размер блока 64 бита, 32 раунда. ГОСТ 28147-89 более удобен для программной реализации, чем DES, имеются сведения о выигрыше по времени примерно в 1.5 раза. В отличие от DES, ГОСТ 28147-89, по-видимому, не был предметом столь глубокого анализа со стороны мирового криптологического сообщества. Тем не менее, как отмечают специалисты, консервативный дизайн и величина основных параметров (длина ключа, размер блока, количество раундов) позволяют утверждать, что шифр вряд ли может быть слабым. Никаких эффективных атак против шифра ГОСТ 28147-89 не опубликовано.

В основе ГОСТ 28147-89, так же как и DES, лежит так называемая структура Фейстела. Блок разбивается на две одинаковые части,

правую R и левую L. Правая часть объединяется с ключевым элементом и посредством некоторого алгоритма шифрует левую часть. Перед следующим раундом левая и правая части меняются местами. Такая структура позволяет использовать один и тот же алгоритм как для шифрования, так и для дешифрования блока. Это особенно важно при аппаратной реализации, так как прямой и обратный шифры формируются одним и тем же устройством (различается только порядок подачи элементов ключа).

Перейдем к непосредственному описанию шифра ГОСТ 28147-89. Введем необходимые определения и обозначения. Последовательность из 32 бит будем называть словом. Блок текста X (64 бита), также как блок шифротекста Y, состоит из двух слов — левого L и правого R, причем L — старшее слово, а R — младшее. Секретный ключ K (256 бит) рассматривается состоящим из восьми слов $K = K_0K_1 \cdots K_7$. На его основе строится так называемый раундовый ключ $W = W_0W_1 \cdots W_{31}$, состоящий из 32 слов (метод построения раундового ключа будет дан позже).

Для работы шифра нужны 8 таблиц S_0, S_1, …, S_7 (называемых также S-боксами). Каждая таблица содержит 16 четырехбитовых элементов, нумеруемых с 0 по 15. Будем обозначать через $S_i[j]$ j-й элемент i-й таблицы. ГОСТ рекомендует заполнять каждую таблицу различными числами из множества $\{0, 1, \ldots, 15\}$, переставленными случайным образом. Содержимое таблиц формирует дополнительный секретный параметр шифра, общий для большой группы пользователей. Заметим, что в DES аналогичные S-боксы фиксированы и несекретны.

В шифре используются следующие операции:

- $+$ сложение слов по модулю 2^{32};
- $\hookleftarrow$ циклический сдвиг слова влево на указанное число бит;
- $\oplus$ побитовое «исключающее или» двух слов, т.е. побитовое сложение по модулю 2.

Алгоритм 4.1. БАЗОВЫЙ ЦИКЛ ШИФРА ГОСТ 28147-89

ВХОД: Блок L, R, раундовый ключ W.
ВЫХОД: Преобразованный блок L, R.
1. FOR $i = 0, 1, \ldots, 31$ DO
2. $k \leftarrow R + W_i, \quad k = (k_7 \cdots k_0)_{16}$;
3. FOR $j = 0, 1, \ldots, 7$ DO
4. $k_j \leftarrow S_j[k_j]$;
5. $L \leftarrow L \oplus (k \leftharpoonup 11)$;
6. $L \longleftrightarrow R$;
7. RETURN L, R.

(На шаге 4 алгоритма используются отдельные четырехбитовые элементы переменной k.)

С помощью базового цикла осуществляется шифрование и дешифрование блока. Чтобы зашифровать блок, строим раундовый ключ

$$W = K_0K_1K_2K_3K_4K_5K_6K_7K_0K_1K_2K_3K_4K_5K_6K_7 \\ K_0K_1K_2K_3K_4K_5K_6K_7K_7K_6K_5K_4K_3K_2K_1K_0, \quad (4.6)$$

подаем на вход X и на выходе получаем Y.

Чтобы дешифровать блок, строим раундовый ключ

$$W = K_0K_1K_2K_3K_4K_5K_6K_7K_7K_6K_5K_4K_3K_2K_1K_0 \\ K_7K_6K_5K_4K_3K_2K_1K_0K_7K_6K_5K_4K_3K_2K_1K_0, \quad (4.7)$$

подаем на вход Y и на выходе получаем X.

Программная реализация обычно требует переработки цикла 3 алгоритма 4.1, так как работа с полубайтами неэффективна. Ясно, что то же самое преобразование может быть выполнено с использованием четырех таблиц по 256 байт или двух таблиц по 65536 полуслов. Например, при работе с байтами имеем $k = (k_3 \cdots k_0)_{256}$, и шаги 3—4 алгоритма 4.1 переписываются следующим образом:

3. FOR $j = 0, 1, 2, 3$ DO
4. $\quad k_j \leftarrow T_j[k_j]$.

Таблицы T_j, $j = 0, 1, 2, 3$, вычисляются предварительно из S-боксов:

FOR $i = 0, 1, \ldots, 255$ DO
$\quad T_j[i] \leftarrow S_{2j}[i \bmod 16] + 16 S_{2j+1}[i \operatorname{div} 16]$.

ГОСТ ничего не говорит о правилах формирования ключевой информации, кроме требования, чтобы каждый S-бокс содержал перестановку различных чисел. В то же время ясно, что, например, нулевой ключ или тривиально заданные S-боксы (отображающие k в себя) не обеспечивают секретности шифра.

4.3. Основные режимы функционирования блоковых шифров

Блоковые шифры применяются для решения многих задач криптографии. В этом разделе мы рассмотрим основные режимы их использования.

В предыдущем разделе были даны примеры реальных блоковых шифров. Теперь мы можем думать о (идеализированном) блоковом шифре, как о преобразовании входного блока X в выходной блок Y с участием секретного ключа K,

$$Y = E_K(X),$$

причем это преобразование должно иметь следующие свойства:

1) при известном Y, но неизвестном K практически невозможно узнать X;

2) при произвольных известных X и Y, но неизвестном K практически невозможно узнать K.

Вначале рассмотрим классическую задачу шифрования сообщений при помощи блоковых шифров.

Режим ECB

Название режима ECB (Electronic CodeBook) можно перевести как электронная кодовая книга.

Сообщение X разбивается на блоки $X = X_1, X_2, \ldots, X_t$. Каждый блок шифруется блоковым шифром

$$Y_i = E_K(X_i), \quad 1 \le i \le t.$$

Получаем зашифрованное сообщение $Y = Y_1, Y_2, \ldots, Y_t$. Дешифрование выполняется по правилу

$$X_i = E_K^{-1}(Y_i), \quad 1 \le i \le t.$$

Нетрудно видеть, что дешифрование сообщения можно производить, выбирая блоки шифротекста в произвольном порядке. Такой режим удобен во многих реальных ситуациях. Например, можно работать с базой данных, хранящейся в зашифрованном виде. Однако при таком использовании одинаковые записи будут зашифрованы одинаково. Говорят, что в режиме ECB шифр сохраняет «образ данных», т.е. некий «рисунок» или шаблон, характерный для данных. Это может дать некоторую информацию противнику. Например, если количество *различных* записей в базе данных невелико (что нередко случается), то противник может составить словарь шифротекстов и вскрыть базу на основе частотного анализа. Заметим, что ему в этом случае не понадобится вскрывать сам шифр.

Некоторые авторы рекомендуют использовать режим ECB только в случаях, когда размер отдельного элемента данных в сообщении,

к которому требуется осуществлять непосредственный (прямой) доступ, меньше размера блока. Остаток блока, свободный от данных, рекомендуется заполнять случайно выбираемыми битами. Тогда даже одинаковые элементы данных будут иметь разные шифротексты. При дешифровании биты заполнения просто отбрасываются.

Если размер элемента данных превышает размер блока, то часто рекомендуют использовать режим CBC.

Режим CBC

Название режима CBC (Cipher-Block Chaining) переводится как сцепление блоков шифра.

Зашифрованное сообщение получается по следующему правилу:

$$Y_i = E_K(X_i \oplus Y_{i-1}), \quad 1 \le i \le t,$$

т.е. каждый последующий блок открытого текста предварительно закрывается предыдущим зашифрованным блоком. Слово Y_0 должно быть определено заранее и известно при шифровании и дешифровании. Полученное зашифрованное сообщение можно дешифровать следующим образом:

$$X_i = Y_{i-1} \oplus E_K^{-1}(Y_i), \quad 1 \le i \le t.$$

Мы получаем шифротекст, в котором каждый следующий блок зависит от предыдущих. Данный режим разрушает «образ данных». Даже если все блоки X идентичны, шифротекст будет состоять из различных блоков Y. Этот режим предпочтителен при шифровании сообщений, размер которых превышает размер блока. Однако дешифровать сообщение можно только последовательно, начиная с первого блока.

4.4. Потоковые шифры

В главе 3 мы рассмотрели шифр Вернама и установили, что он является совершенно секретным, т.е. при его использовании противник, перехвативший зашифрованное сообщение, не получает никакой информации об исходном сообщении. В шифре Вернама зашифрованное сообщение $y_1, y_2, \ldots, y_k$ получается из исходного сообщения $x_1, x_2, \ldots, x_k$ и ключа $z_1, z_2, \ldots, z_k$ при помощи операции шифрования, задаваемой равенством

$$y_i = x_i \oplus z_i, \quad i = 1, 2, \ldots, k. \tag{4.8}$$

Мы видели, что данный шифр совершенен только в том случае, если ключ $z_1, z_2, \ldots, z_k$ образован из независимых и равновероятных символов и используется только один раз. Это приводит к необходимости генерировать случайные последовательности очень большого объема и передавать их по закрытым каналам связи, что весьма затруднительно. Поэтому давно была высказана идея использовать в качестве ключа последовательности не случайные, а порожденные при помощи генераторов псевдослучайных чисел. В этом случае в качестве секретного ключа используется начальное значение или состояние генератора. Однако надо четко осознавать, что в этом случае система уже не будет совершенно секретной. Максимум на что мы можем надеяться, это то, что для раскрытия этой системы потребуется очень много времени (например, нужно будет выполнить перебор всех возможных начальных состояний генератора). В качестве возмещения за потерю совершенности мы получаем возможность использовать короткие (обычно несколько сотен бит) секретные ключи, которые значительно проще распределять и хранить (а при использовании систем с открытым ключом их можно и вычислять).

Определение 4.1. Шифр, построенный на основе (4.8), где в качестве $z_1, z_2, \ldots, z_k$ используется псевдослучайная последовательность, называется *потоковым шифром* (stream cipher).

Как правило, исходное сообщение и ключевая последовательность представляют собой независимые потоки бит. Так как шифрующее (и дешифрующее) преобразование для всех потоковых шифров одно и то же, они различаются только способом построения генераторов псевдослучайных чисел. Действительно, чтобы из шифротекста $y_1, y_2, \ldots, y_k$, полученного по формуле (4.8), восстановить сообщение $x_1, x_2, \ldots, x_k$, необходимо сгенерировать точно такую же последовательность $z_1, z_2, \ldots, z_k$, что и при шифровании, и использовать для дешифрования формулу

$$x_i = y_i \oplus z_i, \quad i = 1, 2, \ldots, k. \tag{4.9}$$

Пример 4.1. Один из самых простых генераторов псевдослучайных чисел (линейный конгруэнтный генератор) работает по схеме

$$z_{i+1} = (az_i + b) \bmod c, \tag{4.10}$$

где a, b, c — некоторые константы, а z_{i+1} — очередное псевдослучайное число, вычисляемое по предыдущему z_i. Обязательно задается начальное значение z_0. Возьмем в качестве примера $a = 5$, $b = 12$, $c = 23$, и пусть $z_0 = 4$. Вычислим несколько элементов последовательности:

$$\begin{aligned}
z_1 &= (4 \cdot 5 + 12) \bmod 23 &&= 9,\\
z_2 &= (9 \cdot 5 + 12) \bmod 23 &&= 11,\\
z_3 &= (11 \cdot 5 + 12) \bmod 23 &&= 21,\\
z_4 &= (21 \cdot 5 + 12) \bmod 23 &&= 2,\\
z_5 &= (2 \cdot 5 + 12) \bmod 23 &&= 22,\\
z_6 &= (22 \cdot 5 + 12) \bmod 23 &&= 7,\\
z_7 &= (7 \cdot 5 + 12) \bmod 23 &&= 1.
\end{aligned}$$

Полученная последовательность внешне выглядит как довольно случайная. □

Для использования в криптографических целях генератор должен удовлетворять следующим основным требованиям:

1) период последовательности должен быть очень большой;

2) порождаемая последовательность должна быть «почти» неотличима от действительно случайной, в частности, вычисление числа z_{i+1} по известным предыдущим элементам последовательности без знания ключа должно быть трудной, практически нерешаемой задачей.

Рассмотренный выше линейный конгруэнтный генератор совершенно не годится для криптографических целей, так как известны простые алгоритмы, позволяющие полностью восстановить параметры генератора всего по нескольким элементам порождаемой им последовательности.

В качестве примеров криптостойких генераторов псевдослучайных чисел мы рассмотрим режимы OFB и CTR блоковых шифров и алгоритм RC4.

Режим OFB блокового шифра

Название режима OFB (Output FeedBack) переводится как обратная связь по выходу. В этом режиме блоковый шифр на основе секретного ключа K и некоторого инициализирующего вектора Y_0 формирует псевдослучайную последовательность r-битовых чисел $z_1, z_2, \ldots, z_k$, которая может использоваться в (4.8) и (4.9) соответственно для шифрования и дешифрования сообщения. Будем считать, как и ранее, что размер блока шифра равен n бит.

Псевдослучайная последовательность получается по схеме

$$\begin{aligned} &Y_i = E_K(Y_{i-1}), \\ &z_i = r \text{ старших бит } Y_i, \quad 1 \le i \le k \end{aligned}$$

(здесь r, $1 \le r \le n$, — параметр метода).

При использовании стойкого блокового шифра можно получить криптостойкий генератор, отвечающий приведенным выше требованиям. А именно, средняя длина периода псевдослучайной последовательности (при случайно выбранных K и Y_0) составляет примерно $r2^{n-1}$ бит [23]. Кроме того, псевдослучайная последовательность «непредсказуема» для противника, так как возможность предсказания (вычисления) z_{i+1} на основе $z_1, \ldots, z_i$ означала бы нестойкость шифра по отношению к атаке по известному тексту. Предсказание z_{i+1} становится даже более трудной задачей, чем взлом блокового шифра, если $r < n$ [23].

Обратим внимание на одну особенность, характерную для всех потоковых шифров. Для шифрования каждого отдельного сообщения необходимо использовать разные K и/или Y_0. В противном случае несколько сообщений будут шифроваться с помощью одних и тех же последовательностей z, и такой шифр может быть раскрыт. Поясним суть проблемы. Пусть два сообщения $u_1, u_2, \ldots, u_k$ и $v_1, v_2, \ldots, v_k$ зашифрованы с помощью одной и той же последовательности z. Тогда шифротексты будут иметь вид

$$u_1 \oplus z_1, u_2 \oplus z_2, \ldots, u_k \oplus z_k \quad \text{и}$$
$$v_1 \oplus z_1, v_2 \oplus z_2, \ldots, v_k \oplus z_k.$$

Сложим оба шифротекста и, с учетом равенства $z_i \oplus z_i = 0$, получим последовательность

$$u_1 \oplus v_1, u_2 \oplus v_2, \ldots, u_k \oplus v_k.$$

Мы получили аналог так называемого «шифра с бегущим ключом», когда один текст шифруется с помощью другого текста, взятого из определенного места определенной книги. Известно, что такой шифр не стоек, хотя использовался в эпоху «донаучной» криптографии [23]. Статистический анализ, основанный на избыточности текстов, позволяет в большинстве случаев достаточно точно восстановить оба сообщения.

Дешифрование сообщений для описанного режима блокового шифра может производиться только с начала, так как невозможно получить произвольный элемент последовательности z, не вычислив предыдущие. В этом смысле режим аналогичен режиму CBC. Преимущество режима OFB заключается в том, что последовательность z может быть сформирована заранее для того, чтобы быстро шифровать или дешифровать сообщения с помощью (4.8), (4.9) в момент их поступления. Это может быть актуально для систем, обрабатывающих данные в реальном масштабе времени.

Режим CTR блокового шифра

Название данного режима происходит от слова CounTeR — счетчик. Этот режим очень похож на OFB, но в нем шифруется не предыдущий выход шифра, а просто счетчик, увеличиваемый на каждом шаге на постоянное число (обычно 1). Точнее, схема выглядит следующим образом:

$$z_i = r \text{ старших бит } E_K(Y_0 + i), \quad i = 1, 2, 3, \ldots,$$

где r — параметр.

При использовании «идеального» блокового шифра режим CTR обеспечивает те же параметры стойкости, что и OFB. Преимущество режима CTR состоит в том, что любой элемент последовательности z может быть вычислен непосредственно. Это дает возможность шифровать и дешифровать любые фрагменты сообщения независимо друг от друга.

Алгоритм RC4

Алгоритм RC4, предложенный Ривестом в 1994 году, относится к классу алгоритмов, разработанных специально для потоковых шифров. Генераторы псевдослучайных чисел, построенные с помощью

таких алгоритмов, как правило, значительно быстрее генераторов, основанных на блоковых шифрах.

Алгоритм RC4 работает с n-битовыми словами (обычно $n = 8$). Все вычисления проводятся по модулю 2^n (остаток $x \bmod 2^n$ вычисляется очень быстро путем выделения n младших бит в x с помощью логической операции «и»). RC4 использует L-словный ключ $K = K_0 K_1 \dots K_{L-1}$ и генерирует последовательность слов $\bar{z} = z_1 z_2 z_3 \dots$, конкретный вид которой определяется ключом K. Состояние генератора задается таблицей S из 2^n слов и двух переменных i и j. В каждый момент времени таблица S содержит все возможные n-битовые числа в перемешанном виде. Так как каждый элемент таблицы принимает значения в промежутке $[0,\ 2^n - 1]$, то его можно трактовать двояко: либо как число, либо как номер другого элемента в таблице. Секретный ключ задает только начальное перемешивание чисел в таблице, которое формируется с помощью следующего алгоритма:

$$
\begin{array}{l}
j \leftarrow 0, \quad S \leftarrow (0, 1, \dots, 2^n - 1); \\
\text{FOR } i = 0, 1, \dots, 2^n - 1 \text{ DO} \\
\qquad j \leftarrow (j + S_i + K_{i \bmod L}) \bmod 2^n, \\
\qquad S_j \leftrightarrow S_i; \\
i \leftarrow 0, \quad j \leftarrow 0.
\end{array}
$$

После этого генератор готов к работе. Генерация очередного псевдослучайного слова z_i осуществляется следующим образом:

$$
\begin{array}{l}
i \leftarrow (i + 1) \bmod 2^n; \\
j \leftarrow (j + S_i) \bmod 2^n; \\
S_j \leftrightarrow S_i; \\
t \leftarrow (S_i + S_j) \bmod 2^n; \\
z_i \leftarrow S_t.
\end{array}
$$

Пример 4.2. Пусть $n = 3$, $K = 25$ ($L = 2$).

Сформируем начальную перестановку чисел в таблице S (все вычисления проводим по модулю 8):

$$\begin{array}{ll} & j=0, \quad S=(0,1,2,3,4,5,6,7),\\ i=0, & j=0+0+2=2, \quad S=(2,1,0,3,4,5,6,7),\\ i=1, & j=2+1+5=0, \quad S=(1,2,0,3,4,5,6,7),\\ i=2, & j=0+0+2=2, \quad S=(1,2,0,3,4,5,6,7),\\ i=3, & j=2+3+5=2, \quad S=(1,2,3,0,4,5,6,7),\\ i=4, & j=2+4+2=0, \quad S=(4,2,3,0,1,5,6,7),\\ i=5, & j=0+5+5=2, \quad S=(4,2,5,0,1,3,6,7),\\ i=6, & j=2+6+2=2, \quad S=(4,2,6,0,1,3,5,7),\\ i=7, & j=2+7+5=6, \quad S=(4,2,6,0,1,3,7,5). \end{array}$$

Теперь вычислим несколько первых элементов псевдослучайной последовательности $\bar{z}$:

$$\begin{array}{l} i=1,\ j=0+2=2,\ S=(4,6,2,0,1,3,7,5),\ t=2+6=0,\ z_1=4,\\ i=2,\ j=2+2=4,\ S=(4,6,1,0,2,3,7,5),\ t=1+2=3,\ z_2=0,\\ i=3,\ j=4+0=4,\ S=(4,6,1,2,0,3,7,5),\ t=2+0=2,\ z_3=1,\\ i=4,\ j=4+0=4,\ S=(4,6,1,2,0,3,7,5),\ t=0+0=0,\ z_4=4,\\ i=5,\ j=4+3=7,\ S=(4,6,1,2,0,5,7,3),\ t=5+3=0,\ z_5=4,\\ i=6,\ j=7+7=6,\ S=(4,6,1,2,0,5,7,3),\ t=7+7=6,\ z_6=7 \end{array}$$

и т.д. Чтобы воспользоваться формулой (4.8) для получения шифра, числа z_i записываем в двоичном виде. В рассмотренном примере каждое число z_i представляется тремя битами, и мы получаем последовательность

$$\bar{z}=1\,0\,0\,0\,0\,0\,0\,0\,1\,1\,0\,0\,1\,0\,0\,1\,1\,1\ \ldots \qquad \square$$

4.5. Криптографические хеш-функции

Хеш-функции (hash functions) находят широкое применение в криптографии, особенно при построении систем цифровой подписи и различных криптографических протоколов, которые будут рассматриваться в последующих главах. В этом разделе мы сформулируем основные требования к криптографически стойким хеш-функциям и рассмотрим один из способов их вычисления.

Определение 4.2. *Хеш-функцией* называется любая функция

$$y = h(x_1 x_2 \dots x_n),$$

которая строке (сообщению) $x_1 x_2 \dots x_n$ *произвольной* длины n ставит в соответствие целое число *фиксированной* длины.

Примером хеш-функции может служить контрольная сумма для сообщения. В этом случае

$$h(x_1 x_2 \dots x_n) = (x_1 + x_2 + \dots + x_n) \bmod 2^w,$$

где w — размер машинного слова. Длина слова, получаемого как значение этой хеш-функции, составляет w бит независимо от длины сообщения. Контрольные суммы очень часто используются для обнаружения непреднамеренных ошибок в сообщении (при изменении одного символа контрольная сумма изменится). Однако очень легко внести *преднамеренную* ошибку в сообщение и сохранить при этом значение контрольной суммы. Если бы такая хеш-функция использовалась, например, при генерации электронной подписи, то было бы очень легко изменить содержание подписанного сообщения. Поэтому рассмотренная хеш-функция не годится для криптографических применений.

Сформулируем основные требования, предъявляемые к криптографическим хеш-функциям. Пусть x — некоторая строка (сообщение). Тогда

1) для любого заданного x вычисление $h(x)$ должно выполняться относительно быстро;

2) при известном y должно быть трудно (практически невозможно) найти x, для которого $y = h(x)$;

3) при известном сообщении x должно быть трудно найти другое сообщение $x' \neq x$, такое, что $h(x') = h(x)$;

4) должно быть трудно найти какую-либо пару различных сообщений x и x', для которых $h(x') = h(x)$.

Отметим, что первое требование должно выполняться всегда, в противном случае хеш-функция теряет какое-либо практическое значение. Остальные требования важны для тех или иных приложений. Например, если пароли для входа в систему хранятся в виде значений соответствующих им хеш-функций, то хеш-функция должна удовлетворять второму требованию. В схеме электронной подписи актуально третье требование. Четвертое требование важно в некоторых криптографических протоколах. Заметим, что четвертое требование более сильное, чем третье (т.е. при выполнении четвертого автоматически выполняется и третье).

Разработка хеш-функции, удовлетворяющей всем четырем требованиям — задача непростая. В настоящее время предложены и практически используются хеш-функции (например, MD5, SHA-1, RIPEMD-160 и др., см., например, [19, 23]), которые считаются отвечающими перечисленным выше требованиям (хотя это строго не доказано). Описание этих и подобных им функций усложнено в деталях и громоздко. Мы рассмотрим универсальный способ построения хеш-функций на базе блоковых шифров, который представляет практический интерес, хотя получаемые хеш-функции и не являются очень быстро вычислимыми. Именно такой подход использован в российском стандарте на криптографическую хеш-функцию (ГОСТ Р34.11-94 [7]).

Пусть дан блоковый шифр E, который для заданного блока X и ключа K формирует шифротекст Y,

$$Y = E_K(X).$$

Мы представим два алгоритма, для которых длина слова, получаемого как значение хеш-функции, равна размеру блока в шифре, но отметим, что известны конструкции, позволяющие получать хеш-функции с длинами слов, кратными размеру блока.

В первом алгоритме сообщение вначале представляется в виде последовательности блоков $X_1, X_2, \ldots, X_n$. Последний блок при необходимости дополняется нулями, иногда в последний блок приписывают длину сообщения в виде двоичного числа. Значение хеш-функции h получается как результат выполнения следующего итерационного процесса:

$h \leftarrow 0$;
FOR $i = 1, 2, \ldots, n$ DO
    $h \leftarrow E_h(X_i) \oplus X_i$.

В качестве начального значения h можно использовать не нуль, а какое-либо «магическое» число, но это не имеет большого значения. В данном алгоритме значение h, полученное на предыдущей итерации, используется в качестве ключа шифра в следующей итерации. Поэтому неявно полагается, что длина ключа в шифре равна длине блока. Однако, как мы видели при изучении шифра RC6, длина ключа может значительно превышать размер блока (в RC6 при максимальной длине блока 256 бит длина ключа может достигать 255 байт, или 2040 бит). В таких случаях более эффективен другой алгоритм.

В этом алгоритме сообщение вначале представляется в виде последовательности $X_1, X_2, \ldots, X_m$, в которой размер каждого элемента равен длине ключа в шифре. Последний элемент заполняется так же, как и в первом алгоритме. Значение хеш-функции h вычисляется следующим образом:

$h \leftarrow 0;$
FOR $i = 1, 2, \ldots, m$ DO
$\quad h \leftarrow E_{X_i}(h) \oplus h.$

Здесь уже элементы сообщения выполняют роль ключей в шифре.

Представленные алгоритмы вычисления хеш-функций удовлетворяют всем четырем требованиям, предъявляемым к криптографическим хеш-функциям, в предположении стойкости используемых блоковых шифров (см. [22, 23]).

Глава 5. ЭЛЕКТРОННАЯ, ИЛИ ЦИФРОВАЯ ПОДПИСЬ

5.1. Электронная подпись RSA

После появления криптографии с открытым ключом произошла настоящая революция в современных компьютерных и сетевых технологиях. Появилась возможность решать задачи, которые ранее считались неразрешимыми, а теперь находят широкое применение на практике. В настоящее время при помощи этих технологий ежедневно проводятся расчеты и совершаются сделки на многие миллиарды долларов, рублей, евро и т.п. Одним из важных элементов этих технологий является электронная, или цифровая подпись. Во многих странах и, в частности, в России введены стандарты на электронную (цифровую) подпись, а само это понятие введено в гражданское законодательство. Термин «электронная подпись» более привычен для России, хотя в последнее время все чаще используется принятый в других странах (прежде всего, в США) термин «цифровая подпись», что отражено и в новых российских стандартах. Оба термина означают одно и то же.

Прежде чем начать рассмотрение криптографической цифровой подписи, сформулируем три свойства, которым (в идеале) должна удовлетворять любая, в частности, обычная рукописная подпись:

1. Подписать документ может только «законный» владелец подписи (и, следовательно, никто не может подделать подпись).

2. Автор подписи не может от нее отказаться.

3. В случае возникновения спора возможно участие третьих лиц (например, суда) для установления подлинности подписи.

Разумеется, цифровая (электронная) подпись также должна обладать всеми этими свойствами, однако лица, подписывающие документы и проверяющие их подлинность, могут находиться за тысячи километров друг от друга и взаимодействовать только через компьютерную сеть.

Кроме обычной подписи, в реальной жизни используется так называемая нотариальная подпись, когда специально выделяемое лицо (нотариус) заверяет документы своей подписью и печатью, так что любое другое лицо может удостовериться в их подлинности. Аналог этой подписи также востребован в киберпространстве. Электронная нотариальная подпись реализуется точно так же, как и подпись любого другого лица.

В этом разделе мы рассмотрим электронную подпись, базирующуюся на схеме RSA.

Если Алиса планирует подписывать документы, то она должна вначале выбрать параметры RSA точно так же, как это описано в разд. 2.6. Для этого Алиса выбирает два больших простых числа P и Q, вычисляет $N = PQ$ и $\phi = (P-1)(Q-1)$. Затем она выбирает число d, взаимно простое с ϕ, и вычисляет $c = d^{-1} \bmod \phi$. Наконец, она публикует числа N и d, например, помещает их на своем сайте, ассоциировав со своим именем, и хранит в секрете число c (остальные числа P, Q и ϕ можно забыть, они больше не потребуются). Теперь Алиса готова ставить свои подписи на документах или сообщениях.

Пусть Алиса хочет подписать сообщение $\bar{m} = m_1, \ldots, m_n$. Тогда вначале она вычисляет криптографическую хеш-функцию

$$y = h(m_1, \ldots, m_n),$$

которая ставит в соответствие сообщению $\bar{m}$ число y. Предполагается, что алгоритм вычисления хеш-функции всем известен. Отметим наиболее важное для нас свойство используемой хеш-функции:

практически невозможно изменить основной текст $m_1, \ldots, m_n$, не изменив y. Поэтому на следующем шаге Алисе достаточно снабдить подписью только число y, и эта подпись будет относиться ко всему сообщению $\bar{m}$.

Алиса вычисляет число

$$s = y^c \bmod N, \tag{5.1}$$

т.е. она возводит число y в свою секретную степень. Число s это и есть цифровая подпись. Она просто добавляется к сообщению $\bar{m}$, и тем самым Алиса имеет сформированное подписанное сообщение

$$\langle \bar{m},\ s \rangle\,. \tag{5.2}$$

Теперь каждый, кто знает открытые параметры Алисы, ассоциированные с ее именем, т.е. числа N и d, может проверить подлинность ее подписи. Для этого необходимо, взяв подписанное сообщение (5.2), вычислить значение хеш-функции $h(\bar{m})$, число

$$w = s^d \bmod N \tag{5.3}$$

и проверить выполнение равенства $w = h(\bar{m})$.

Утверждение 5.1. *Если подпись подлинная, то* $w = h(\bar{m})$.

Доказательство. Из (5.3), (5.1) и свойств схемы RSA (см. разд. 2.6) следует

$$w = s^d \bmod N = y^{cd} \bmod N = y = h(\bar{m}). \qquad \square$$

Утверждение 5.2. *Описанная электронная подпись удовлетворяет всем требованиям, предъявляемым к подписи.*

Д о к а з а т е л ь с т в о . Проверим первое свойство подписи. Никто не может разложить число N на простые множители (при больших N порядка 1024 бит по состоянию на 2005 год эта задача практически неразрешима). Поэтому, зная N и d невозможно найти c. Действительно, чтобы вычислить $c = d^{-1} \bmod \phi$, нужно знать $\phi = (P-1)(Q-1)$, а для этого нужно знать простые множители P и Q. Таким образом, первое свойство выполнено — никто, кроме Алисы, не может знать число c и поэтому не может подписать сообщение.

Второе свойство выполнено вследствие первого. Автор подписи не может от нее отказаться, так как никто другой не может «сфабриковать» подпись от его имени.

Третье свойство также очевидно — в случае спора заинтересованная сторона может предъявить судье все вычисления для их проверки и выяснения истины. □

П р и м е р 5.1. Пусть $P = 5$, $Q = 11$. Тогда $N = 5 \cdot 11 = 55$, $\phi = 4 \cdot 10 = 40$. Пусть $d = 3$. Такой выбор d возможен, так как $\gcd(40, 3) = 1$. Параметр $c = 3^{-1} \bmod 40$ вычисляем с помощью обобщенного алгоритма Евклида (см. разд. 2.3), $c = 27$.

Пусть, например, Алиса хочет подписать сообщение $\bar{m} = abbbaa$, для которого значение хеш-функции равно, скажем, 13:

$$y = h(abbbaa) = 13.$$

В этом случае Алиса вычисляет по (5.1)

$$s = 13^{27} \bmod 55 = 7$$

и формирует подписанное сообщение

$$\langle abbbaa,\ 7 \rangle .$$

Теперь тот, кто знает открытые ключи Алисы $N = 55$ и $d = 3$, может проверить подлинность подписи. Получив подписанное сообщение, он заново вычисляет значение хеш-функции

$$h(abbbaa) = 13$$

(если содержание сообщения не изменено, то значение хеш-функции совпадет с тем, которое вычисляла Алиса) и вычисляет по (5.3)

$$w = 7^3 \bmod 55 = 13.$$

Значения w и хеш-функции совпали, значит, подпись верна. □

З а м е ч а н и е. Обратим внимание на то, что одна и та же схема RSA, сгенерированная Алисой, может использоваться для решения двух задач. Во-первых, Алиса может подписывать сообщения, как это было показано в данном разделе, используя свой *секретный ключ* c. Во-вторых, кто угодно может послать Алисе зашифрованное сообщение (число), расшифровать которое, как это было показано в разд. 2.6, сможет только она, используя для шифрования ее *открытый ключ* d.

5.2. Стандарты на электронную (цифровую) подпись

Во многих странах сегодня существуют стандарты на электронную (цифровую) подпись. В этом разделе мы опишем российский государственный стандарт ГОСТ Р34.10-94 и стандарт США FIPS 186. Российский стандарт, как следует из его обозначения, был принят в 1994 году, американский — в 1991. В основе обоих стандартов лежит по сути один и тот же алгоритм, называемый DSA (Digital Signature Algorithm). Мы подробно рассмотрим российскую версию алгоритма, а затем укажем на отличия американской версии.

Вначале для некоторого сообщества пользователей выбираются общие несекретные параметры. Прежде всего необходимо найти два простых числа, q длиной 256 бит и p длиной 1024 бита, между которыми выполняется соотношение

$$p = bq + 1 \tag{5.4}$$

для некоторого целого b. Старшие биты в p и q должны быть равны единице. Затем выбирается число $a > 1$, такое, что

$$a^q \bmod p = 1. \tag{5.5}$$

В результате получаем три общих параметра — p, q и a.

Замечание. Равенство (5.5) означает, что при возведении a в степени по модулю p показатели приводятся по модулю q, т.е. $a^b \bmod p = a^{b \bmod q} \bmod p$ (мы уже проводили обоснование подобного феномена при доказательстве утверждения 2.10 на стр. 35). Такое приведение будет постоянно выполняться при генерации и проверке подписи, в результате чего длина показателей степени в рамках рассматриваемого алгоритма никогда не будет превышать 256 бит, что упрощает вычисления.

Далее, каждый пользователь выбирает случайно число x, удовлетворяющее неравенству $0 < x < q$, и вычисляет

$$y = a^x \bmod p. \tag{5.6}$$

Число x будет секретным ключом пользователя, а число y — открытым ключом. Предполагается, что открытые ключи всех пользователей указываются в некотором несекретном, но «сертифицированном» справочнике, который должен быть у всех, кто собирается проверять подписи. Отметим, что в настоящее время найти x по y практически невозможно при указанной выше длине модуля p. На этом этап выбора параметров заканчивается, и мы готовы к тому, чтобы формировать и проверять подписи.

Пусть имеется сообщение $\bar{m}$, которое необходимо подписать. Генерация подписи выполняется следующим образом:

1. Вычисляем значение хеш-функции $h = h(\bar{m})$ для сообщения m, значение хеш-функции должно лежать в пределах $0 < h < q$ (в российском варианте хеш-функция определяется ГОСТом Р34.11-94).

2. Формируем случайное число k, $0 < k < q$.

3. Вычисляем $r = \left(a^k \bmod p\right) \bmod q$. Если оказывается так, что $r = 0$, то возвращаемся к шагу 2.

4. Вычисляем $s = (kh + xr) \bmod q$. Если $s = 0$, то возвращаемся к шагу 2.

5. Получаем подписанное сообщение $\langle \bar{m};\ r,\ s \rangle$.

Для проверки подписи делаем следующее.

1. Вычисляем хеш-функцию для сообщения $h = h(\bar{m})$.
2. Проверяем выполнение неравенств $0 < r < q$, $0 < s < q$.
3. Вычисляем $u_1 = s \cdot h^{-1} \bmod q$, $u_2 = -r \cdot h^{-1} \bmod q$.
4. Вычисляем $v = (a^{u_1} y^{u_2} \bmod p) \bmod q$.
5. Проверяем выполнение равенства $v = r$.

Если хотя бы одна из проверок на шагах 2 и 5 не дает нужного результата, то подпись считается недействительной. Если же все проверки удачны, то подпись считается подлинной.

Утверждение 5.3. *Если подпись к сообщению была сформирована законно, т.е. обладателем секретного ключа x, то $v = r$.*

Доказательство. Запишем следующую цепочку равенств, которая следует непосредственно из описания метода (напомним, что показатели степени приводятся по модулю q):

$$\begin{aligned} v &= \left(a^{sh^{-1}} y^{-rh^{-1}} \bmod p\right) \bmod q = \\ &= \left(a^{(kh+xr)h^{-1}} a^{-xrh^{-1}} \bmod p\right) \bmod q = \\ &= \left(a^{k+xrh^{-1}-xrh^{-1}} \bmod p\right) \bmod q = \\ &= \left(a^{k} \bmod p\right) \bmod q = r. \end{aligned}$$

□

З а м е ч а н и е. Чтобы найти параметр a, удовлетворяющий (5.5), рекомендуется использовать следующий метод. Берем случайное число $g > 1$ и вычисляем

$$a = g^{(p-1)/q} \bmod p. \tag{5.7}$$

Если $a > 1$, то это то, что нам нужно. Действительно, на основании (5.7) и теоремы Ферма имеем

$$a^q \bmod p = g^{((p-1)/q)q} \bmod p = g^{p-1} \bmod p = 1,$$

т.е. выполняется равенство (5.5). Если при вычислении по (5.7) мы получаем $a = 1$ (крайне маловероятный случай), то нужно просто взять другое число g.

П р и м е р 5.2. Выберем общие несекретные параметры

$$q = 11, \quad p = 6q + 1 = 67,$$

возьмем $g = 10$ и вычислим

$$a = 10^6 \bmod 67 = 25.$$

Выберем секретный ключ $x = 6$ и вычислим открытый ключ

$$y = 25^6 \bmod 67 = 62.$$

Сформируем подпись для сообщения $\bar{m} = baaaab$. Пусть для хеш-функции этого сообщения $h(\bar{m}) = 3$. Возьмем случайно число $k = 8$. Вычислим

$$r = \left(25^8 \bmod 67\right) \bmod 11 = 24 \bmod 11 = 2,$$
$$s = (8 \cdot 3 + 6 \cdot 2) \bmod 11 = 36 \bmod 11 = 3.$$

Получаем подписанное сообщение

$$\langle baaaab;\ 2,\ 3\rangle.$$

Теперь выполним проверку подписи. Если сообщение не изменено, то $h = 3$. Вычислим

$$\begin{aligned}
&h^{-1} = 3^{-1} \bmod 11 = 4,\\
&u_1 = 3 \cdot 4 \bmod 11 = 1,\\
&u_2 = -2 \cdot 4 \bmod 11 = -8 \bmod 11 = 3,\\
&v = \left(25^1 \cdot 62^3 \bmod 67\right) \bmod 11 =\\
&\qquad\qquad = (25 \cdot 9 \bmod 67) \bmod 11 = 24 \bmod 11 = 2.
\end{aligned}$$

Мы видим, что $v = r$, значит подпись верна. □

Теперь остановимся на отличиях американского стандарта от российского. Они сводятся к следующему.

1. Длина числа q берется равной 160 бит.
2. В качестве хеш-функции используется алгоритм SHA-1.
3. При генерации подписи на шаге 4 параметр s вычисляется по формуле $s = k^{-1}(h + xr) \bmod q$.
4. При проверке подписи на шаге 3 u_1 и u_2 вычисляются по формулам $u_1 = h \cdot s^{-1} \bmod q$, $u_2 = r \cdot s^{-1} \bmod q$.

С учетом этих отличий нетрудно переписать всю схему подписи в «американском» стиле. Доказательство корректности алгоритма проводится совершенно аналогично.

Задачи и упражнения

Во всех задачах будем предполагать, что $h(m) = m$ для всех значений m.

5.1. Построить подпись RSA для сообщения m при следующих параметрах пользователя:

а. $P = 5$, $Q = 11$, $c = 27$, $m = 7$,

б. $P = 5$, $Q = 13$, $c = 29$, $m = 10$,

в. $P = 7$, $Q = 11$, $c = 43$, $m = 5$,

г. $P = 7$, $Q = 13$, $c = 29$, $m = 15$,

д. $P = 3$, $Q = 11$, $c = 7$, $m = 24$.

См. ответ.

5.2. Для указаннных открытых ключей пользователя RSA проверить подлинность подписанных сообщений:

а. $N = 55$, $d = 3$: $\langle 7, 28\rangle$, $\langle 22, 15\rangle$, $\langle 16, 36\rangle$,

б. $N = 65$, $d = 5$: $\langle 6, 42\rangle$, $\langle 10, 30\rangle$, $\langle 6, 41\rangle$,

в. $N = 77$, $d = 7$: $\langle 13, 41\rangle$, $\langle 11, 28\rangle$, $\langle 5, 26\rangle$,

г. $N = 91$, $d = 5$: $\langle 15, 71\rangle$, $\langle 11, 46\rangle$, $\langle 16, 74\rangle$,

д. $N = 33$, $d = 3$: $\langle 10, 14\rangle$, $\langle 24, 18\rangle$, $\langle 17, 8\rangle$.

См. ответ.

5.3. Сообщество пользователей ГОСТа Р34.10-94 имеют общие параметры $q = 11$, $p = 67$, $a = 25$. Вычислить открытый ключ (y) и построить подпись для сообщения m при следующих секретных параметрах:

а. $x = 3$, $h = m = 10$, $k = 1$,

б. $x = 8$, $h = m = 1$, $k = 3$,

в. $x = 5$, $h = m = 5$, $k = 9$,

г. $x = 2$, $h = m = 6$, $k = 7$,

д. $x = 9$, $h = m = 7$, $k = 5$.

См. ответ.

5.4. Для указаннных открытых ключей (y) пользователей ГОСТа Р34.10-94 с общими параметрами $q = 11$, $p = 67$, $a = 25$ проверить подлинность подписанных сообщений:

а. $y = 14$: $\langle 10; 4, 5\rangle$, $\langle 10; 7, 5\rangle$, $\langle 10; 3, 8\rangle$,

б. $y = 24$: $\langle 1; 3, 5\rangle$, $\langle 1; 4, 3\rangle$, $\langle 1; 4, 5\rangle$,

в. $y = 40$: $\langle 7; 7, 4\rangle$, $\langle 7; 9, 2\rangle$, $\langle 5; 9, 2\rangle$,

г. $y = 22$: $\langle 6; 9, 5\rangle$, $\langle 8; 8, 3\rangle$, $\langle 7; 4, 1\rangle$,

д. $y = 64$: $\langle 10; 7, 3\rangle$, $\langle 7; 7, 10\rangle$, $\langle 8; 7, 5\rangle$.

См. ответ.

Глава 6. КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ

Рассмотренные в предыдущих главах криптографические методы часто используются в качестве инструментов для решения других практически важных задач. Современная криптография позволяет решать проблемы, которые ранее считались в принципе неразрешимыми. Причем в настоящее время многие такие возможности криптографии используются в реальных компьютерных системах. Это и заключение коммерческих сделок в режиме удаленного взаимодействия участников, и осуществление денежных расчетов по сети, и проведение выборов по компьютерным сетям, и многое другое. Методы решения подобных задач обычно описываются в форме так называемых криптографических протоколов. Некоторые из них будут представлены в этой главе.

Обратим внимание читателя на то, что криптографические алгоритмы не просто предоставляют новые возможности пользователю (например, не нужно ходить в банк, можно произвести все необходимые операции со своего домашнего компьютера). Важно то, что они способны обеспечивать надежность значительно более высокую, чем традиционные механизмы. Например, если бумажную банкноту можно подделать, и случаи подделок весьма многочисленны, то электронную банкноту, созданную при помощи криптографических методов, подделать практически невозможно.

6.1. Доказательства с нулевым знанием

Рассмотрим следующую задачу, возникающую в некоторых криптографических приложениях. Снова участвуют Алиса и Боб. Алиса знает решение некоторой сложной задачи, она хочет убедить Боба в этом, однако так, чтобы Боб не узнал самого решения задачи. Т.е. в результате Боб должен убедиться в том, что Алиса знает решение, но не должен узнать что-нибудь о самом решении. На первый взгляд сама задача кажется абсурдной, а возможность ее решения — фантастической! Для того чтобы лучше понять ситуацию, рассмотрим случай из жизни пиратов. Пусть, например, Алиса знает карту острова, где спрятан клад, а Боб — капитан корабля, который может доставить ее на остров. Алиса хочет доказать, что карта у нее есть, не показывая ее Бобу (иначе Боб обойдется без Алисы, и весь клад достанется ему).

Такая же задача актуальна для компьютерных сетей в тех случаях, когда Боб (сервер или контроллер домена) должен принять решение о допуске Алисы к информации, хранящейся в сети, но при этом Алиса не хочет, чтобы кто-либо, прослушивающий канал передачи данных и сам сервер, получил какие-либо знания о ее пароле. Т.е. Боб получает «нулевое знание» о пароле (или карте) Алисы, но уверен, что у Алисы такой пароль (или карта) есть.

Итак, наша задача — построить протокол «доказательства с нулевым знанием». При этом мы считаем, что каждый из участников может вести «нечестную» игру и пытаться обмануть другого.

В качестве сложной задачи, решение которой известно Алисе, мы рассмотрим задачу нахождения гамильтонова цикла в графе. Отметим, что эта задача NP-полная. Мы не приводим формального определения NP-полноты, которое может быть найдено, например, в [1]. Для читателя, не знакомого с этим определением, отметим только, что NP-полнота задачи неформально означает, что время решения задачи растет экспоненциально с ростом размера задачи (объема исходных

данных).

Задача о нахождении гамильтонова цикла в графе

Рассматриваемая в данном разделе задача не просто предоставляет нам возможность описать одну схему построения протокола доказательства с нулевым знанием, но и имеет важное теоретическое значение. Блюм (Manuel Blum) показал, что, выражаясь неформально, любое математическое утверждение может быть представлено в виде графа, причем доказательство этого утверждения соответствует гамильтонову циклу в этом графе (см., например, [26]). Поэтому наличие протокола доказательства с нулевым знанием для гамильтонова цикла означает, что доказательство любого математического утверждения может быть представлено в форме доказательства с нулевым знанием.

Определение 6.1. *Гамильтоновым циклом* в графе называется непрерывный путь, проходящий через все вершины графа ровно по одному разу.

Пример 6.1. Рассмотрим граф, изображенный на рис. 6.1.

Путь, проходящий последовательно через вершины 8, 2, 4, 6, 3, 5, 7, 1, представляет собой гамильтонов цикл. Действительно, в этом пути содержатся все вершины графа, и каждая вершина посещается только один раз. □

Ясно, что если в графе G с n вершинами гамильтонов цикл существует, то при некоторой нумерации вершин он пройдет точно через вершины с последовательными номерами $1, 2, 3, \ldots, n$. Поэтому путем перебора всех возможных нумераций вершин мы обязательно найдем гамильтонов цикл. Но количество возможных нумераций равно $n!$, и поэтому уже при умеренно больших n, например, при $n = 100$, такой подход становится практически нереализуемым. Доказано, что задача нахождения гамильтонова цикла в графе является

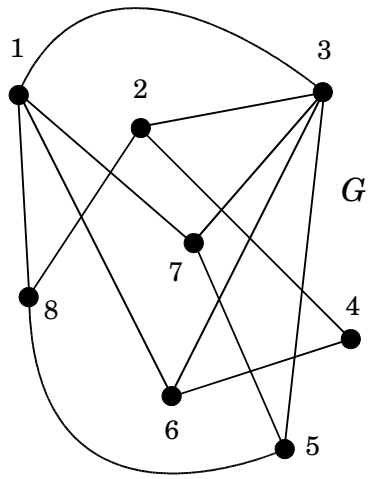

Рис. 6.1. Граф с гамильтоновым циклом (8, 2, 4, 6, 3, 5, 7, 1)

NP-полной. Мы уже говорили кратко о понятии NP-полноты. Неформально, NP-полнота *рассматриваемой* задачи означает, что для ее решения не существуют (точнее, неизвестны) алгоритмы существенно более быстрые, чем указанный метод перебора.

Нашей задачей будет построение криптографического протокола, с помощью которого Алиса будет доказывать Бобу, что она знает гамильтонов цикл в некотором графе G так, чтобы Боб не получил никаких знаний о самом этом цикле. Иными словами, Алиса будет предоставлять Бобу доказательство с нулевым знанием. Еще раз напомним читателю, что «нулевое знание» означает, что независимо от числа реализаций протокола доказательства Боб будет располагать точно такими же сведениями о гамильтоновом цикле, какие он мог бы получить, просто изучая представленный ему граф G.

Итак, допустим, что Алиса знает гамильтонов цикл в графе G. Теперь она может это доказывать Бобу и всем, кто имеет граф G, с

помощью описываемого ниже протокола. Алиса может использовать это доказательство, например, для идентификации своей личности. Но прежде чем мы перейдем к описанию протокола, договоримся о некоторых обозначениях.

Мы будем обозначать графы буквами G, H, F, понимая под этим одновременно соответствующие матрицы смежности. Элемент матрицы $H_{ij} = 1$, если в графе H есть ребро, соединяющее вершины i и j; $H_{ij} = 0$ в противном случае. Символом $\|$ будем обозначать конкатенацию (сцепление) двух чисел, точнее, двоичных слов, им соответствующих. Нам понадобится шифр с открытым ключом. Вообще говоря, это может быть любой шифр, но для определенности будем использовать шифр RSA (разд. 2.6). Будем считать, что Алиса сформировала систему RSA с открытыми параметрами N и d. Важно, что зашифрованные в этой системе сообщения может расшифровать только Алиса и больше никто.

Протокол доказательства состоит из следующих четырех шагов (пояснения будут даны ниже).

Шаг 1. Алиса строит граф H, являющийся копией исходного графа G, где у всех вершин новые, случайно выбранные номера. На языке теории графов говорят, что H изоморфен G. Иными словами, H получается путем некоторой перестановки вершин в графе G (с сохранением связей между вершинами). Алиса кодирует матрицу H, приписывая к первоначально содержащимся в ней нулям и единицам случайные числа r_{ij} по схеме $\tilde{H}_{ij} = r_{ij} \| H_{ij}$. Затем она шифрует элементы матрицы $\tilde{H}$, получая зашифрованную матрицу F, $F_{ij} = \tilde{H}_{ij}^d \bmod N$. Матрицу F Алиса передает Бобу.

Шаг 2. Боб, получив зашифрованный граф F, задает Алисе один из двух вопросов.

1. Каков гамильтонов цикл для графа H?

2. Действительно ли граф H изоморфен G?

Шаг 3. Алиса отвечает на соответствующий вопрос Боба.

1. Она расшифровывает в F ребра, образующие гамильтонов цикл.
2. Она расшифровывает F полностью (фактически передает Бобу граф $\tilde{H}$) и предъявляет перестановки, с помощью которых граф H был получен из графа G.

Шаг 4. Получив ответ, Боб проверяет правильность расшифровки путем повторного шифрования и сравнения с F и убеждается либо в том, что показанные ребра действительно образуют гамильтонов цикл, либо в том, что предъявленные перестановки действительно переводят граф G в граф H.

Весь протокол повторяется t раз.

Обсудим вначале кратко несколько вопросов по построению протокола.

1. Зачем Алиса строит изоморфный граф? Если бы она этого не делала, то Боб, получив ответ на свой вопрос номер один, узнал бы гамильтонов цикл в графе G.

2. Зачем Алиса кодирует матрицу H? С этим приемом мы уже встречались при шифровании цветов вершин графа. Дело в том, что невозможно зашифровать непосредственно нули и единицы (с помощью шифра RSA они вообще не шифруются). Даже если заменить их на какие-то произвольные числа a и b, то мы получим всего два различных шифротекста, и Бобу не составит труда понять, какой из них какому числу соответствует. Т.е. структура графа не будет скрыта. Здесь мы сталкиваемся

с типичной ситуацией, когда требуется использовать так называемый рандомизированный шифр. И такой шифр строится путем добавления случайных чисел в матрицу H перед шифрованием. Закодированная матрица $\tilde{H}$ точно также задает граф (нечетность числа означает наличие ребра, четность — его отсутствие), но после шифрования $\tilde{H}$ структура графа полностью скрывается (мы используем известное свойство шифра RSA — он полностью скрывает четность числа [22]).

3. Зачем Боб задает два вопроса? Если бы он задавал только вопрос номер один, который по смыслу протокола является основным, то Алиса, не зная в действительности гамильтонова цикла в графе G, могла бы предъявить Бобу совсем другой граф с таким же количеством вершин и искуственно заложенным в него гамильтоновым циклом. Поэтому Боб иногда просит Алису доказать изоморфизм графов H и G. Важно, что Алиса не знает заранее, какой из двух вопросов задаст Боб.

4. Почему Боб не может задать сразу двух вопросов? В этом случае он узнал бы гамильтонов цикл в G, так как ему был бы показан гамильтонов цикл в H и правило перехода от H к G.

5. Зачем Боб проверяет правильность расшифровки? Если бы он этого не делал, то Алиса на четвертом шаге могла бы предоставить ему «выгодную» для себя информацию, а не ту, которую она посылала ему на втором шаге.

Более точно основные детали протокола обосновываются в ходе доказательства двух основных утверждений.

Утверждение 6.1. *Вероятность обмана при t реализациях протокола не превосходит 2^{-t}.*

Доказательство. Вначале покажем, что вероятность обмана в одной реализации протокола равна 1/2. Заметим, что если Алиса действительно знает гамильтонов цикл в графе G, то она может правильно ответить на любой вопрос Боба. Если же она не знает гамильтонов цикл, то самое большее, что она может сделать, — это подготовиться к ответу на первый либо на второй вопрос. В ожидании первого вопроса, она создает новый граф с искусственно заложенным в него гамильтоновым циклом. Но в этом случае она не сможет доказать его изоморфизм графу G. В ожидании второго вопроса, она строит граф, изоморфный графу G. Но в этом случае она не сможет показать в нем гамильтонов цикл. Таким образом, вероятность успешности обмана равна вероятности угадывания номера вопроса. В предположении, что Боб задает оба вопроса с одинаковыми вероятностями, мы получаем, что вероятность обмана равна 1/2.

Так как Боб прекращает игру при первом же неправильном ответе, вероятность обмана при t реализациях протокола не превосходит $(1/2)^t$. □

Утверждение 6.2. *Представленный протокол реализует доказательство с нулевым знанием.*

Доказательство. Чтобы доказать, что Боб не получает никаких знаний в ходе реализации протокола, достаточно показать, что все, что он получает от Алисы, он мог бы получить сам, не вступая с ней ни в какое общение.

Рассмотрим вначале второй вопрос Боба. В ответ на этот вопрос он получает граф, изоморфный графу G. Но он сам мог строить сколько угодно изоморфных графов, и то, что присылает ему Алиса, это просто один из них.

Случай с первым вопросом не столь очевиден. В ответ на первый вопрос Боб получает гамильтонов цикл в графе, изоморфном графу G. На первый взгляд может показаться, что это дает Бобу какую-то

информацию. Однако это не так. Заметим, что если в G есть гамильтонов цикл, то при некоторой нумерации вершин существует изоморфный граф, который задается матрицей смежности вида

$$\begin{pmatrix} * & 1 & * & \cdots & * & * & * \\ * & * & 1 & \cdots & * & * & * \\ & & & \cdots & & & \\ * & * & * & \cdots & * & 1 & * \\ * & * & * & \cdots & * & * & 1 \\ 1 & * & * & \cdots & * & * & * \end{pmatrix}, \tag{6.1}$$

где $*$ означает неопределенность в наличии или отсутствии ребра. Т.е. при такой нумерации гамильтонов цикл проходит через вершины в порядке возрастания номеров. Изменяя нумерацию вершин, Боб может получать из (6.1) всевозможные изоморфные матрицы. Когда Алиса, отвечая на его первый вопрос, открывает гамильтонов цикл, Боб видит как раз одну из таких матриц.

Таким образом, Боб не получает от Алисы никакой информации, которую он не мог бы получить сам. □

Рассмотрим пример, иллюстрирующий все основные этапы описанного протокола.

Пример 6.2. Возьмем в качестве основного граф G, изобра-

женный на рис. 6.1. Его матрица смежности имеет вид

$$G = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{array} \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & \boxed{1} \\ 0 & 0 & 1 & \boxed{1} & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & \boxed{1} & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & \boxed{1} & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & \boxed{1} & 1 \\ 1 & 0 & \boxed{1} & 1 & 0 & 0 & 0 & 0 \\ \boxed{1} & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & \boxed{1} & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

В матрице с помощью $\boxed{\cdot}$ показан гамильтонов цикл. Алиса выбирает некоторую случайную нумерацию вершин, скажем, 7, 4, 5, 3, 1, 2, 8, 6, и получает изоморфный граф

$$H = \begin{array}{c} \\ 7 \\ 4 \\ 5 \\ 3 \\ 1 \\ 2 \\ 8 \\ 6 \end{array} \begin{pmatrix} 7 & 4 & 5 & 3 & 1 & 2 & 8 & 6 \\ 0 & 0 & 1 & 1 & \boxed{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & \boxed{1} \\ \boxed{1} & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & \boxed{1} & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & \boxed{1} & 1 \\ 0 & \boxed{1} & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & \boxed{1} & 0 & 0 \\ 0 & 1 & 0 & \boxed{1} & 1 & 0 & 0 & 0 \end{pmatrix}.$$

Для шифрования матрицы будем использовать систему RSA с параметрами $N = 55$, $d = 3$. Вначале закодируем матрицу H. В рамках данного примера просто припишем слева к каждому элементу матрицы выбираемую случайно с равными вероятностями цифру из мно-

жества $\{1, 2, 3, 4, 5\}$:

$$\tilde{H} = \begin{pmatrix} 50 & 20 & 11 & 31 & 21 & 40 & 20 & 10 \\ 40 & 30 & 50 & 20 & 10 & 41 & 50 & 21 \\ 41 & 30 & 50 & 11 & 30 & 20 & 51 & 40 \\ 11 & 10 & 41 & 30 & 51 & 41 & 30 & 21 \\ 31 & 20 & 40 & 11 & 50 & 10 & 41 & 31 \\ 50 & 41 & 20 & 21 & 40 & 10 & 21 & 50 \\ 40 & 30 & 31 & 50 & 41 & 21 & 30 & 40 \\ 20 & 41 & 10 & 51 & 41 & 20 & 30 & 40 \end{pmatrix}.$$

Теперь мы шифруем матрицу $\tilde{H}$, возводя каждый ее элемент в куб по модулю 55:

$$F = \begin{pmatrix} 40 & 25 & 11 & 36 & 21 & 35 & 25 & 10 \\ 35 & 50 & 40 & 25 & 10 & 06 & 40 & 21 \\ 06 & 50 & 40 & 11 & 50 & 25 & 46 & 35 \\ 11 & 10 & 06 & 50 & 46 & 06 & 50 & 21 \\ 36 & 25 & 35 & 11 & 40 & 10 & 06 & 36 \\ 40 & 06 & 25 & 21 & 35 & 10 & 21 & 40 \\ 35 & 50 & 36 & 40 & 06 & 21 & 50 & 35 \\ 25 & 06 & 10 & 46 & 06 & 25 & 50 & 35 \end{pmatrix}.$$

(При внимательном просмотре матрицы F может показаться, что использованный нами шифр плохо скрывает исходную матрицу H. Это объясняется тем, что, во-первых, модуль 55 слишком мал и, во-вторых, в матрице $\tilde{H}$ много чисел, не взаимно простых с модулем. Для реальных систем RSA, где N — большое число, такая ситуация практически исключена.)

Боб получает матрицу F и задает один из двух вопросов. Если он просит доказать изоморфизм графов, то Алиса просто посылает ему кодированную матрицу $\tilde{H}$ и использованную нумерацию 7, 4, 5, 3, 1, 2, 8, 6. Боб проверяет соответствие матрицы $\tilde{H}$ матрице F,

т.е.выполнение равенств $50^3 \bmod 55 = 40$, $20^3 \bmod 55 = 25$ и т.д. Из матрицы $\tilde{H}$ Боб получает граф H (просто отбросив старшую десятичную цифру). Затем он переставляет вершины графа G в соответствии с полученной нумерацией, как это делала Алиса, и убеждается в том, что H и G — один и тот же граф.

Если Боб просит показать ему гамильтонов цикл, то Алиса посылает ему соответствующий список (закодированных) ребер графа H: (1, 5, 21), (5, 7, 41), (7, 6, 21), ..., (3, 1, 41). Каждый элемент содержит номера вершин и код ребра. Боб проверяет соответствие указанных в списке ребер матрице F, например, $21^3 \bmod 55 = 21 = F_{1,5}$, $41^3 \bmod 55 = 06 = F_{5,7}$ и т.д. Затем он убеждается, что указанный в списке путь проходит через все вершины графа по одному разу. □

6.2. Электронные деньги

Во многих странах люди оплачивают покупки при помощи электронных карточек, заказывают авиабилеты через Интернет, покупают самые разнообразные товары в Интернет-магазинах. Сведения о покупках накапливаются в магазинах и банках. Поэтому появилась новая проблема, иногда называемая «проблема Большого Брата».

Суть проблемы состоит в том, что исчезает анонимность процесса покупки, т.е. информация о покупках любого человека может стать известной третьим лицам и использоваться против него. Например, сведения о покупке билета на поезд или самолет могут представлять интерес для преступников, информация о закупках алкогольных напитков политическим деятелем может быть использована против него его противниками и т.д., и т.п.

Поэтому возникла идея разработать такие схемы электронных платежей, которые сохраняли бы анонимность покупателя в той же степени, что и при расчете наличными деньгами. Такие протоколы называются электронными или цифровыми деньгами (digital cache), что

подчеркивает их основное свойство — они обеспечивают ту же степень анонимности, что и обычные деньги. Некоторые схемы уже используются в реальной жизни. Описываемая ниже схема была предложена Д. Чаумом (David Chaum), см. [2, 22].

Мы рассмотрим две «плохие» схемы, а затем «хорошую», чтобы было легче понять суть метода.

Вначале дадим более точную постановку задачи. Имеются три участника: банк, покупатель и магазин. Покупатель и магазин имеют соответствующие счета в банке, и покупатель хочет купить некоторый товар в магазине. Покупка осуществляется в виде трехступенчатого процесса:

1) покупатель снимает нужную сумму со своего счета в банке;

2) покупатель «пересылает» деньги в магазин;

3) магазин сообщает об этом в банк, соответствующая сумма денег зачисляется на счет магазина, а покупатель забирает товар(или последний ему доставляется).

Наша цель — разработать такую схему, чтобы

- она была надежна;

- чтобы банк не знал, кто купил товар, т.е. была сохранена анонимность обычных денег.

Опишем первую «плохую» схему (она базируется на RSA). Банк имеет следующую информацию: секретные числа P, Q, c и открытые

$$N = PQ,$$
$$d = c^{-1} \bmod (P-1)(Q-1). \tag{6.2}$$

Допустим, покупатель решил израсходовать некоторую заранее оговоренную с банком сумму (например, 100\$). (Мы сначала рассмотрим

случай, когда может использоваться «банкнота» только одного номинала (скажем, 100$).) Покупатель высылает в банк число n, которое будет номером банкноты (обычно требуется, чтобы генерировалось случайное число в промежутке $[2,\ N-1]$).

Банк вычисляет число

$$s = n^c \bmod N \tag{6.3}$$

и формирует банкноту $\langle n,\ s\rangle$, которую возвращает покупателю, предварительно уменьшив его счет на 100$. Параметр s в банкноте — это подпись банка. Никто не может подделать подпись, так как число c секретно.

Покупатель предъявляет банкноту $\langle n,\ s\rangle$ в магазине, чтобы купить товар. Магазин отправляет эту банкноту в банк для проверки. Прежде всего, банк проверяет правильность подписи (эту проверку мог бы сделать и магазин, используя открытые ключи банка). Но кроме этого банк хранит все номера возвратившихся к нему банкнот и проверяет, нет ли числа n в этом списке. Если n есть в списке, то платеж не принимается (кто-то пытается использовать банкноту повторно), и банк сообщает об этом магазину. Если же все проверки прошли успешно, то банк добавляет 100$ на счет магазина, а магазин отпускает товар покупателю.

Недостаток этой схемы — отсутствует анонимность. Банк, а также все, кто имеет доступ к открытым линиям связи, могут запомнить, какому покупателю соответствует число n, и тем самым выяснить, кто купил товар.

Рассмотрим вторую «плохую» схему, которая уже обеспечивает анонимность. Эта схема базируется на так называемой «слепой подписи».

Снова покупатель хочет купить товар. Он генерирует число n, которое теперь *не будет* посылаться в банк. Затем он генерирует слу-

чайное число r, взаимно простое с N, и вычисляет число

$$\hat{n} = (n \cdot r^d) \bmod N. \tag{6.4}$$

Число $\hat{n}$ покупатель отправляет в банк.

Банк вычисляет число

$$\hat{s} = \hat{n}^c \bmod N \tag{6.5}$$

и отправляет $\hat{s}$ обратно покупателю (не забыв при этом снять 100$ с его счета).

Покупатель находит число $r^{-1} \bmod N$ и вычисляет

$$s = (\hat{s} \cdot r^{-1}) \bmod N. \tag{6.6}$$

Заметим, что с учетом соотношений (6.5), (6.4) и (6.2) имеем

$$s = \hat{n}^c \cdot r^{-1} = (n \cdot r^d)^c \cdot r^{-1} = n^c r^{dc} \cdot r^{-1} = n^c r^1 r^{-1} = n^c \bmod N,$$

т.е. мы получили подпись банка к n (см. (6.3)), но самого числа n ни банк, ни кто либо другой не видел. Вычисление (6.5) называется «слепой подписью», так как реальное сообщение (n) подписывающий не видит и узнать не может.

Таким образом, покупатель имеет число n, которое никому не известно и никогда не передавалось по каналам связи, и подпись банка s, совпадающую с вычисленной по (6.3). Покупатель формирует банкноту $\langle n,\ s\rangle$ и действует так же, как в первой «плохой» схеме. Но теперь никто не знает, кому соответствует эта банкнота, т.е. она стала анонимной, как обычная бумажная банкнота.

Действия магазина и банка после предъявления покупателем банкноты $\langle n,\ s\rangle$ ничем не отличаются от действий, описанных в первой схеме.

Почему же данная схема плохая? Она имеет следующий недостаток: можно сфабриковать фальшивую банкноту, если известны хотя бы две настоящие. Делается это так. Путь злоумышленник (будь

то покупатель или магазин) имеет две настоящие банкноты $\langle n_1,\ s_1\rangle$ и $\langle n_2,\ s_2\rangle$. Тогда он легко сможет изготовить фальшивую банкноту $\langle n_3,\ s_3\rangle$, вычислив числа

$$n_3 = n_1 n_2 \bmod N,$$
$$s_3 = s_1 s_2 \bmod N.$$

Действительно,

$$n_3^c = (n_1 n_2)^c = n_1^c n_2^c = s_1 s_2 = s_3 \bmod N, \tag{6.7}$$

т.е. s_3 является правильной подписью для n_3, и у банка нет никаких оснований, чтобы не принять эту фальшивую банкноту (он просто не сможет отличить ее от подлинной). Это так называемое «мультипликативное свойство» системы RSA.

Опишем, наконец, «хорошую» схему, в которой устранены все недостатки первых двух. В одном варианте такой схемы используется некоторая односторонняя функция

$$f:\ \{1,\dots,N\} \to \{1,\dots,N\}$$

(f вычисляется легко, а обратная к ней функция f^{-1} — очень трудно). Функция f не секретна и известна всем (покупателю, банку и магазину).

Банкнота теперь определяется как пара чисел $\langle n,\ s_f\rangle$, где

$$s_f = (f(n))^c \bmod N,$$

т.е. подписывается не n, а значение $f(n)$.

Покупатель генерирует n (никому его не показывая), вычисляет $f(n)$, подписывает в банке при помощи «слепой подписи» число $f(n)$ и формирует банкноту $\langle n,\ s_f\rangle$. Эта банкнота обладает всеми хорошими свойствами, как и во второй схеме, в то же время подделать такую

банкноту невозможно, так как невозможно вычислить f^{-1}. Для проверки подписи (т.е. подлинности банкноты) нужно вычислить $f(n)$ и убедиться, что

$$s_f^d \bmod N = f(n).$$

Заметим, что при выборе односторонней функции нужно проявлять осторожность. Например, функция $f(n) = n^2 \bmod N$, которая действительно является односторонней, не годится для рассматриваемого протокола. Читатель может проверить, что банкноты, созданные с использованием такой функции, будут по-прежнему обладать мультипликативным свойством (6.7). На практике в качестве $f(n)$ обычно используются криптографичесие хеш-функции, описываемые в главе 4.

Все остальные действия магазина и банка остаются такими же, как и в ранее описанных схемах.

Есть еще один, более простой, способ борьбы с мультипликативным свойством системы RSA — внесение избыточности в сообщение. Допустим, что длина модуля N — 1024 бита. Такой же может быть и длина числа n. Будем записывать (случайно выбираемый) номер банкноты только в младшие 512 бит n, а в старшие 512 бит n запишем некоторое фиксированное число. Это фиксированное число может нести полезную информацию, такую, как номинал банкноты и наименование банка (с помощью 512 бит можно представить строку из 64 символов ASCII). Теперь банк при предъявлении ему банкноты будет обязательно проверять наличие фиксированного заголовка в параметре n и отвергать банкноту в случае его отсутствия. Вероятность того, что при перемножении двух чисел по модулю N результат совпадет с ними в 512 битах пренебрежимо мала. Поэтому получить фальшивую банкноту по формуле (6.7) не удастся.

П р и м е р 6.3. Пусть в качестве секретных параметров банка выбраны числа $P = 17$, $Q = 7$, $c = 77$. Соответствующие им открытые параметры будут $N = 119$, $d = 5$. Для исключения возможности

подделки банкнот их допустимыми номерами считаются только числа, состоящие из двух одинаковых десятичных цифр, например, 11, 77, 99.

Когда покупатель хочет получить банкноту, он вначале случайным образом выбирает ее номер (из числа допустимых). Предположим, он выбрал $n = 33$. Затем он находит случайное число r, взаимнопростое со 119. Допустим, $r = 67$, $\gcd(67, 119) = 1$. Далее, покупатель вычисляет

$$\hat{n} = (33 \cdot 67^5) \bmod 119 = (33 \cdot 16) \bmod 119 = 52.$$

Именно число 52 он посылает в банк.

Банк списывает со счета покупателя 100$ и отправляет ему число

$$\hat{s} = 52^{77} \bmod 119 = 103.$$

Покупатель вычисляет $r^{-1} = 67^{-1} \bmod 119 = 16$ и $s = 103 \cdot 16 \bmod 119 = 101$ и получает платежеспособную банкноту

$$\langle n, s \rangle = \langle 33, 101 \rangle.$$

Эту банкноту он приносит (или посылает) в магазин, чтобы купить товар.

Магазин предъявляет банкноту в банк. Банк делает следующие проверки:

1) номер банкноты ($n = 33$) состоит из двух одинаковых десятичных цифр (т.е. содержит требуемую избыточность);

2) ранее банкнота с таким номером не предъявлялась;

2) подпись банка верна, т.е. $33^5 \bmod 119 = 101$.

Так как все проверки прошли успешно, банк зачисляет 100$ (это фиксированный номинал банкноты) на счет магазина, о чем ему и сообщает. Магазин отпускает товар покупателю. □

В завершение разберем еще две проблемы, возникающие в связи с рассмотренной схемой электронных денег.

В представленной схеме независимо действующие покупатели или даже один покупатель, который не помнит номеров ранее использованных им банкнот, могут случайно сгенерировать две или более банкноты с одинаковыми номерами. По условиям протокола банк примет к оплате только одну из таких банкнот (ту, которая будет предъявлена первой). Однако примем во внимание размеры чисел, используемых в протоколе. Если номер банкноты — число длиной 512 бит и покупатели генерируют его действительно случайным образом, то вероятность получения когда либо двух одинаковых номеров пренебрежимо мала.

Вторая проблема состоит в том, что в рассмотренной схеме используются только банкноты одного фиксированного номинала, что, конечно, неудобно для покупателя. Решение проблемы использования банкнот разного номинала возможно следующим образом. Банк заводит несколько пар $(c_i,\ d_i)$, обладающих свойством (6.2), и объявляет, что d_1 соответствует, например, 1000 руб., d_2 — 500 руб. и т.д. Когда покупатель запрашивает слепую подпись в банке, он дополнительно сообщает, какого номинала банкноту он хочет получить. Банк снимает с его счета сумму, равную указанному номиналу, и формирует подпись, используя соответствующее секретное число c_i. Когда впоследствии банк получает подписанную банкноту, он использует для проверки подписи по очереди числа d_1, d_2 и т.д. Если подпись оказалась верна для какого-то d_i, то принимается банкнота i-го номинала. В случае, когда параметр n банкноты содержит фиксированный заголовок с указанием ее номинала, задача проверки подписи облегчается — банк сразу использует нужный ключ d_i.

6.3. Взаимная идентификация с установлением ключа

В данном разделе мы рассмотрим криптографически стойкий протокол, в результате реализации которого два абонента сети A и B взаимно идентифицируют друг друга (т.е. A убеждается в том, что взаимодействует с B, а B — в том, что он взаимодействует с A) и формируют общий секретный ключ, который может использоваться в дальнейшем для шифрования передаваемых ими сообщений. В реальной жизни в качестве A и B могут выступать пользователь и компьютерная система или две различные компьютерные системы — суть описываемого ниже протокола от этого не меняется.

В процессе описания мы рассматриваем различные, все более изощренные типы атак и средства защиты от них. Так мы уже рассматривали ранее (см. разд. 2.1 и 2.2) подходы к решению задачи идентификации и установления ключа. Однако мы исходили из того, что противник может только прослушивать информацию, передаваемую по открытому каналу. Но в современных сетях передачи данных, например в Интернете, информация от одного пользователя передается другому через множество промежуточных узлов (маршрутизаторы, шлюзы, почтовые серверы и т.д..), не контролируемых этими пользователями. В результате противник, обосновавшийся на одном таком промежуточном узле, может не только прослушивать информацию, т.е. играть чисто пассивную роль, но и осуществлять активные воздействия, например, изменять, добавлять или удалять сообщения.

Разберем, например, типичную атаку на систему Диффи–Хеллмана в сети связи с активным противником. Алиса выбирает свое секретное число X_A и посылает Бобу g^{X_A}. Боб выбирает свое секретное число X_B и посылает Алисе g^{X_B}. Однако Ева перехватывает эти числа и посылает вместо них и Алисе, и Бобу g^{X_E}, где X_E — ее число. Все эти числа выглядят как совершенно случайные, так что ни

Алиса, ни Боб ничего не подозревают. В результате Алиса формирует ключ $K_A = g^{X_E X_A}$, а Боб — ключ $K_B = g^{X_E X_B}$. Оба этих ключа могут быть легко вычислены и Евой. Теперь, когда Алиса посылает Бобу сообщение, зашифрованное с ключом K_A, Ева расшифровывает его, снова шифрует с ключом K_B и отправляет Бобу. Аналогично Ева действует и при передаче сообщений в обратном направлении. Боб и Алиса взаимодействуют, как им кажется, в защищенном режиме, но на самом деле Ева читает все их сообщения.

Такая атака становится невозможной, если Алиса и Боб не передают открытые ключи (в системе Диффи–Хеллмана это $Y_A = g^{X_A}$ и $Y_B = g^{X_B}$) по каналу связи, а выбирают их из некоторой таблицы или справочника, который был получен ими ранее из «надежного» источника (как это предполагалось в разд. 2.2).

Вообще, большинство криптосистем с открытыми ключами требуют наличия некоторой организационной структуры, занимающейся сертификацией открытых ключей. Такая структура может, например, выглядеть следующим образом. В сети, которой принадлежат Алиса и Боб, имеется «честный» пользователь Трент (абонент T), который заинтересован только в том, чтобы сеть работала надежно (скорее всего это не человек, а хорошо охраняемый компьютер, работающий по жестко заложенной программе). Трент располагает какой-либо надежной криптосистемой (например, RSA с длиной модуля порядка 10000 бит) с соответствующими открытыми ключами и выполняет всего две функции:

1) он добавляет в свою базу данных информацию об открытом ключе пользователя, присылаемую в виде сообщения, зашифрованного с использованием открытого ключа Трента;

2) он сообщает информацию о чьем-либо открытом ключе, снабженную своей подписью.

Открытые ключи Трента доводятся до сведения всех пользователей

каким-либо способом, исключающим вмешательство Евы. Например, они публикуются в виде рекламного сообщения в газете. Теперь Алиса, вычислив свой открытый ключ, формирует сообщение из своего имени и этого ключа, шифрует его с использованием открытого ключа Трента и посылает Тренту (никто кроме Трента не может расшифровать это сообщение). Боб, когда ему нужен открытый ключ Алисы, посылает запрос Тренту, и Трент присылает ему подписанный ключ Алисы (никто не может подделать подпись Трента). Боб проверяет подпись Трента, используя его открытый ключ, и принимает ключ Алисы как достоверный. Таким образом, каждый пользователь сети получает достоверную информацию об открытых ключах других пользователей, и Ева никак не может вмешаться в этот процесс.

Итак, если Алиса и Боб пользуются достоверными открытыми ключами, то схема Диффи–Хеллмана решает задачу установления секретного ключа. Однако она непосредственно не обеспечивает идентификацию пользователей. Действительно, если вместо Алисы выступала Ева, которая не знает секретного ключа Алисы, то у них с Бобом будут сформированы различные секретные ключи, но это может выясниться только позднее, на стадии обмена данными, когда Боб, например, не сможет расшифровать переданное ему сообщение или обнаружит, что «Алиса» не понимает того, что он посылает ей. Часто требуется обеспечить явную идентификацию, чтобы по завершении протокола стороны точно знали, кто есть кто.

У схемы Диффи–Хеллмана есть и другой недостаток: секретный ключ, который формируют Алиса и Боб, будет всегда один и тот же, пока они не поменяют открытые ключи. Но смена открытых ключей — это относительно долгий процесс (например, обычно требуется оповестить всех пользователей сети об изменении какого-то открытого ключа, чтобы они могли скорректировать информацию в своих справочниках). Хотелось бы иметь протокол, обеспечивающий оперативное создание каждый раз различных, случайно выбираемых секретных ключей.

Решение состоит в использовании какого-либо шифра с открытым ключом для передачи секретных ключей. Обозначим шифр сообщения x, построенный с использованием открытого ключа пользователя A, через $P_A(x)$. (Например, $P_A(x)$ может быть шифром RSA или шифром Эль-Гамаля. В случае RSA $P_A(x) = x^{d_A} \bmod N_A$, где пара чисел d_A и N_A представляет собой открытый ключ пользователя A.) Все, кто знает открытый ключ A, могут вычислить $P_A(x)$ для сообщения x. В то же время только A, знающий соответствующий секретный ключ, может получить x из $y = P_A(x)$. Аналогично ($P_B(x)$) будем обозначать шифр, построенный с помощью открытого ключа пользователя B. Символом $\|$, как и ранее, будем обозначать конкатенацию чисел. Мы опишем протокол поэтапно, чтобы не «утопить» читателя в деталях.

Напомним, что мы решаем следующую задачу: Алиса и Боб хотят взаимно идентифицировать друг-друга и установить общий секретный ключ. Рассмотрим вначале следующий (плохой) протокол, состоящий из трех шагов, чтобы обсудить несколько важных вопросов.

Шаг 1. Алиса придумывает секретный ключ k_1, шифрует его, используя открытый ключ Боба, и посылает Бобу:

$$A \longrightarrow B: \quad P_B(k_1). \tag{6.8}$$

Шаг 2. Боб расшифровывает k_1, снова шифрует его, используя открытый ключ Алисы, и посылает Алисе:

$$A \longleftarrow B: \quad P_A(k_1). \tag{6.9}$$

Шаг 3. Алиса расшифровывает k_1 и сравнивает его с тем, который она придумала на шаге 1.

Что мы имеем в результате реализации этого протокола? Во-первых, Алиса и Боб получили общий секретный ключ k_1, неизвестный Еве (Ева не может расшифровать ни $P_B(k_1)$, ни $P_A(k_1)$). Во-вторых, Алиса получила криптографически стойкую идентификацию

Боба, так как никто кроме него не смог бы расшифровать k_1. Очевидно, что в данном протоколе Боб не получает никакой идентификации Алисы (сообщение (6.8) мог послать кто угодно). Он мог бы провести симметричный протокол со своей стороны:

$$A \longleftarrow B: \quad P_A(k_2), \tag{6.10}$$

$$A \longrightarrow B: \quad P_B(k_2) \tag{6.11}$$

и получить такую идентификацию. Проблема, однако, здесь состоит в логической независимости двух протоколов, в результате чего нет гарантии, что оба протокола проводятся одними и теми же участниками.

Но есть и более тонкая проблема. Алиса может использовать описанный протокол для вскрытия криптосистемы Боба! Делается это так. Допустим, Алиса перехватила какое-то сообщение y, предназначенное для Боба, т. е. $y = P_B(x)$. Она притворяется, что хочет войти в систему Боба, и запускает протокол (6.8), (6.9). Однако вместо $P_B(k_1)$ она передает Бобу сообщение y. Так как k_1 — произвольно выбранное число, то Боб не может ничего заподозрить. Он честно выполняет свой шаг в протоколе и расшифровывает для Алисы x!

Урок, который следует отсюда извлечь, следующий: никогда ни для кого не следует расшифровывать случайные числа. Это может повредить вашей безопасности. Средство борьбы с такой «опасной» случайностью — внесение избыточности в сообщения, например, введение какого-либо элемента, известного получателю и ожидаемого им. В частности, в (6.8) Алиса могла бы послать свое имя. Она могла бы построить сообщение, отведя 512 бит под случайное число k_1 и 512 бит под свое имя, адрес, фрагмент открытого ключа и другую легко проверяемую информацию (будем обозначать все это вместе через $\hat{A}$), и послать Бобу $P_B(k_1 \| \hat{A})$. В этом случае Боб не стал бы посылать Алисе сообщение x, так как его соответствующие 512 бит наверняка не содержали бы $\hat{A}$.

Все вышеизложенное приводит нас к следующему протоколу Нидхама–Шредера (Needham, Schroeder, см., например, [23]), который полностью решает посталенную в начале раздела задачу.

Шаг 1. Алиса выбирает случайное число k_1, объединяет его со своей открытой информацией $\hat{A}$ и посылает Бобу

$$A \longrightarrow B: \quad P_B(k_1 \| \hat{A}). \tag{6.12}$$

Шаг 2. Боб расшифровывает (6.12) и убеждается в том, что полученное сообщение содержит открытую информацию Алисы $\hat{A}$. Затем он выбирает случайное число k_2, объединяет его с k_1 и посылает Алисе

$$A \longleftarrow B: \quad P_A(k_1 \| k_2). \tag{6.13}$$

Шаг 3. Алиса расшифровывает (6.13) и убеждается в том, что полученное сообщение содержит k_1. Это является для нее надежным признаком идентификации Боба, так как никто другой не мог бы извлечь k_1 из (6.12). Алиса посылает Бобу

$$A \longrightarrow B: \quad P_B(k_2). \tag{6.14}$$

Шаг 4. Боб расшифровывает (6.14) и убеждается в том, что он получил k_2. Это является для него надежным признаком идентификации Алисы, так как никто другой не мог бы извлечь k_2 из (6.13).

Теперь Алиса и Боб могут сформировать из k_1, k_2 общий ключ, например, $k = k_1 \oplus k_2$, где $\oplus$ — побитовая сумма по модулю 2, или использовать k_1 и k_2 по-отдельности для шифрования входящих и исходящих сообщений.

П р и м е р 6.4. Пусть в некоторой сети используется шифр Эль-Гамаля с открытыми параметрами $p = 107$, $g = 2$. Пользователи A и B имеют открытые ключи $d_A = 58$, $d_B = 28$, которым соответствуют секретные $c_A = 33$, $c_B = 45$. Рассмотрим реализацию протокола Нидхама–Шредера для взаимной идентификации пользователей A и B и установления общего секретного ключа. Учитывая небольшую величину модуля p в нашем примере, будем использовать в качестве идентификаторов пользователей одну цифру в десятичной записи, пусть $\hat{A} = 1$, $\hat{B} = 2$, и секретный ключ будем получать также в виде одной десятичной цифры.

На первом шаге протокола A выбирает секретный ключ, пусть $k_1 = 3$, и формирует сообщение $m = k_1 \| \hat{A} = 31$. Это сообщение шифруется шифром Эль-Гамаля на открытом ключе пользователя B:

$$k = 15, \quad r = g^k \bmod p = 2^{15} \bmod 107 = 26,$$
$$e = m \cdot {d_B}^k \bmod p = 31 \cdot 28^{15} \bmod 107 = 47.$$

Пара чисел $(26, 47)$ и есть тот шифротекст, который необходимо послать B. В использованных при описании протокола обозначениях $P_B(k_1 \| \hat{A}) = (26, 47)$ и

$$A \longrightarrow B : \quad (26, 47).$$

На втором шаге протокола B расшифровывает $(26, 47)$, используя свой секретный ключ:

$$m' = e \cdot r^{p-1-c_B} \bmod p = 47 \cdot 26^{106-45} \bmod 107 = 31.$$

B убеждается, что младшая цифра содержит идентификационный номер пользователя A и извлекает $k_1 = 3$. Затем он выбирает свое секретное число, пусть $k_2 = 7$, формирует сообщение $m = k_1 \| k_2 = 37$ и шифрует его на открытом ключе A:

$$k = 77, \quad r = g^k \bmod p = 2^{77} \bmod 107 = 63,$$
$$e = m \cdot {d_A}^k \bmod p = 37 \cdot 58^{77} \bmod 107 = 18.$$

Пара чисел $(63, 18)$ — это то, что нужно послать A. Т.е. $P_A(k_1 \| k_2) = (63, 18)$ и

$$A \longleftarrow B: \quad (63, 18).$$

На третьем шаге A расшифровывает $(63, 18)$:

$$m' = e \cdot r^{p-1-c_A} \bmod p = 18 \cdot 63^{106-33} \bmod 107 = 37.$$

A убеждается в том, что старшая цифра содержит $k_1 = 3$ и извлекает $k_2 = 7$. Теперь A шифрует k_2 для B:

$$k = 41, \quad r = g^k \bmod p = 2^{41} \bmod 107 = 82,$$
$$e = m \cdot {d_B}^k \bmod p = 7 \cdot 28^{41} \bmod 107 = 49,$$

и посылает B

$$A \longrightarrow B: \quad (82, 49).$$

На четвертом шаге B расшифровывает $(82, 49)$:

$$m' = e \cdot r^{p-1-c_B} \bmod p = 49 \cdot 82^{106-45} \bmod 107 = 7.$$

B убеждается в том, что он получил свое число $k_2 = 7$.

Теперь A и B могут сформировать общий ключ по заранее оговоренной схеме, например,

$$k = k_1 \oplus k_2 = 3 \oplus 7 = (011)_2 \oplus (111)_2 = (100)_2 = 4. \qquad \square$$

Задачи и упражнения

6.1. В системе электронных денег выбраны секретные параметры банка $P = 17$, $Q = 7$, $c = 77$, а соответствующие им открытые параметры $N = 119$, $d = 5$. Сформировать электронные банкноты со следующими номерами:

а. $n = 11$ при $r = 5$,

б. $n = 99$ при $r = 6$,

в. $n = 55$ при $r = 10$,

г. $n = 44$ при $r = 15$,

д. $n = 77$ при $r = 30$.

См. ответ.

ОТВЕТЫ К ЗАДАЧАМ И УПРАЖНЕНИЯМ

1.1. **а.** $k = 17$. **б.** $k = 27$.

1.2. **а.** ПРИВЕТ ($k = 5$). **б.** ВЕСНА ($k = 20$).

2.1. **а.** $5 = 5$, $16 = 6$, $27 = 7$, $-4 = 6$, $-13 = -3 = 7$, $3 + 8 = 1$, $3 - 8 = 5$, $3 \cdot 8 = 4$, $3 \cdot 8 \cdot 5 = 4 \cdot 5 = 0 \pmod{10}$. **б.** $5 = 5$, $16 = 5$, $27 = 5$, $-4 = 7$, $-13 = -2 = 9$, $3 + 8 = 0$, $3 - 8 = 6$, $3 \cdot 8 = 2$, $3 \cdot 8 \cdot 5 = 2 \cdot 5 = 10 \pmod{11}$.

2.2. $2^8 \bmod 10 = 6$, $3^7 \bmod 10 = 7$, $7^{19} \bmod 100 = 43$, $7^{57} \bmod 100 = 7$.

2.3. $108 = 2 \cdot 2 \cdot 3 \cdot 3 \cdot 3$, $77 = 7 \cdot 11$, $65 = 5 \cdot 13$, $30 = 3 \cdot 3 \cdot 5$, $159 = 3 \cdot 53$.

2.4. пары $(25, 12)$ и $(40, 27)$ взаимно просты, другие — нет (числа $(25, 15)$ делятся на 5, $(13, 39)$ делятся на 13).

2.5. $\varphi(14) = 6$, $\varphi(20) = 8$.

2.6. $\varphi(53) = 52$, $\varphi(21) = \varphi(7) \cdot \varphi(3) = 6 \cdot 2 = 12$, $\varphi(159) = 2 \cdot 52 = 104$.

2.7. $3^{13} \bmod 13 = 3 \cdot 3^{12} \bmod 13 = 3$, $5^{22} \bmod 11 = 5^2 \cdot 5^{10} \cdot 5^{10} \bmod 11 = 25 \bmod 11 = 3$, $3^{17} \bmod 5 = 3$.

2.8. $3^9 \bmod 20 = 3 \cdot 3^8 \bmod 20 = 3$, $2^{14} \bmod 21 = 2^2 \cdot 2^{12} \bmod 21 = 4$, $2^{107} \bmod 159 = 2^3 \cdot 2^{104} \bmod 159 = 8$.

2.9. $\gcd(21, 12) = 3$, $\gcd(30, 12) = 6$, $\gcd(24, 40) = \gcd(40, 24) = 8$, $\gcd(33, 16) = 1$.

2.10. **а.** $x = -1$, $y = 2$. **б.** $x = 1$, $y = -2$. **в.** $x = 2$, $y = -1$. **г.** $x = 1$, $y = -2$.

2.11. $3^{-1} \bmod 7 = 5$, $5^{-1} \bmod 8 = 5$, $3^{-1} \bmod 53 = 18$, $10^{-1} \bmod 53 = 16$.

2.12. Простые числа, меньшие 100: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 73, 79, 83, 89, 97. Из них числа 5, 7, 11, 23, 47, 59 и 83 соответствуют виду $p = 2q + 1$.

2.13. При $p = 11$ в качестве параметра g могут быть выбраны числа 2, 6, 7 и 8.

2.14. **а.** $Y_A = 20$, $Y_B = 17$, $Z_{AB} = 21$. **б.** $Y_A = 13$, $Y_B = 14$, $Z_{AB} = 10$. **в.** $Y_A = 21$, $Y_B = 9$, $Z_{AB} = 16$. **г.** $Y_A = 8$, $Y_B = 5$, $Z_{AB} = 9$. **д.** $Y_A = 6$, $Y_B = 17$, $Z_{AB} = 16$.

2.15. **а.** $d_A = 11$, $d_B = 13$, $x_1 = 17$, $x_2 = 5$, $x_3 = 6$, $x_4 = 4$. **б.** $d_A = 3$, $d_B = 19$, $x_1 = 8$, $x_2 = 12$, $x_3 = 3$, $x_4 = 6$. **в.** $d_A = 5$, $d_B = 11$, $x_1 = 14$, $x_2 = 10$, $x_3 = 3$, $x_4 = 10$. **г.** $d_A = 5$, $d_B = 15$, $x_1 = 7$, $x_2 = 21$, $x_3 = 14$, $x_4 = 17$. **д.** $d_A = 11$, $d_B = 5$, $x_1 = 15$, $x_2 = 2$, $x_3 = 8$, $x_4 = 9$.

2.16. **а.** $d_B = 13$, $r = 14$, $e = 12$, $m' = 5$. **б.** $d_B = 16$, $r = 9$, $e = 15$, $m' = 10$. **в.** $d_B = 15$, $r = 16$, $e = 14$, $m' = 10$. **г.** $d_B = 21$, $r = 14$, $e = 12$, $m' = 5$. **д.** $d_B = 8$, $r = 5$, $e = 5$, $m' = 10$.

2.17. **а.** $N_A = 55$, $\varphi(N_A) = 40$, $c_A = 27$, $e = 23$, $m' = 12$. **б.** $N_A = 65$, $\varphi(N_A) = 48$, $c_A = 29$, $e = 50$, $m' = 20$. **в.** $N_A = 77$, $\varphi(N_A) = 60$, $c_A = 43$, $e = 52$, $m' = 17$. **г.** $N_A = 91$, $\varphi(N_A) = 72$, $c_A = 29$, $e = 88$, $m' = 30$. **д.** $N_A = 33$, $\varphi(N_A) = 20$, $c_A = 7$, $e = 9$, $m' = 15$.

2.18. $m = 111$.

3.1. **а.** $\bar{e} = 1111001110$. **б.** $\bar{e} = 1111110101$. **в.** $\bar{e} = 0001000110$. **г.** $\bar{e} = 0101011011$. **д.** $\bar{e} = 0001010001$.

3.2. **а.** $P_1 \approx 0.002$, $P_2 \approx 0.006$, $P_3 \approx 0.623$, $P_4 \approx 0.051$, $P_5 \approx 0.311$, $P_6 \approx 0.007$. **б.** $P_1 \approx 0.000$, $P_2 \approx 0.009$, $P_3 \approx 0.000$, $P_4 \approx 0.000$, $P_5 \approx 0.892$, $P_6 \approx 0.099$. **в.** $P_1 \approx 0.000$, $P_2 \approx 0.697$, $P_3 \approx 0.000$, $P_4 \approx 0.004$, $P_5 \approx 0.299$, $P_6 \approx 0.000$. **г.** $P_1 \approx 0.003$, $P_2 \approx 0.000$, $P_3 \approx 0.036$, $P_4 \approx 0.000$, $P_5 \approx 0.801$, $P_6 \approx 0.160$. **д.** $P_1 \approx 0.196$, $P_2 \approx 0.000$, $P_3 \approx 0.001$, $P_4 \approx 0.000$, $P_5 \approx 0.018$, $P_6 \approx 0.785$.

3.3. **а.** $H \approx 1.16$, $n \approx 6.04$. **б.** $H \approx 0.52$, $n \approx 2.42$. **в.** $H \approx 0.9$, $n \approx 3.76$. **г.** $H \approx 1.08$, $n \approx 5.08$. **д.** $H \approx 1.16$, $n \approx 6.04$.

3.4. **а.** $P_1 \approx 0.7$ ($\bar{m} = bcacbcacc$), $P_2 = 0$, $P_3 \approx 0.3$ ($\bar{m} = acbcacbcc$), $P_4 = 0$, $P_5 = 0$, $P_6 = 0$. **б.** $P_1 = 0$, $P_2 = 0$, $P_3 = 0$, $P_4 \approx 0.21$ ($\bar{m} = bcccaccac$), $P_5 \approx 0.20$ ($\bar{m} = abbbcbbcb$), $P_6 \approx 0.59$ ($\bar{m} = acccbccbc$). **в.** $P_1 = 0$, $P_2 = 0$, $P_3 = 0$, $P_4 = 1$ ($\bar{m} = ccbcabccb$), $P_5 = 0$, $P_6 = 0$. **г.** $P_1 = 0$, $P_2 = 0$, $P_3 \approx 0.000$ ($\bar{m} = acbbbbcbb$), $P_4 \approx 1.000$ ($\bar{m} = abccccbcc$), $P_5 = 0$, $P_6 = 0$. **д.** $P_1 = 0$, $P_2 = 0$, $P_3 \approx 0.009$ ($\bar{m} = bbbcbbbcb$), $P_4 \approx 0.970$ ($\bar{m} = cccbcccbc$), $P_5 = 0$, $P_6 \approx 0.021$ ($\bar{m} = cccacccac$).

5.1. **а.** $s = 28$. **б.** $s = 30$. **в.** $s = 26$. **г.** $s = 71$. **д.** $s = 18$.

5.2. **а.** $\langle 7, 28\rangle$ подлинно, $\langle 22, 15\rangle$ нет, $\langle 16, 36\rangle$ подлинно. **б.** $\langle 6, 42\rangle$ нет, $\langle 10, 30\rangle$ да, $\langle 6, 41\rangle$ да. **в.** $\langle 13, 41\rangle$ да, $\langle 11, 28\rangle$ нет, $\langle 5, 26\rangle$ да. **г.** $\langle 15, 71\rangle$ да, $\langle 11, 46\rangle$ нет, $\langle 16, 74\rangle$ да. **д.** $\langle 10, 14\rangle$ нет, $\langle 24, 18\rangle$ да, $\langle 17, 8\rangle$ да.

5.3. **а.** $y = 14$, $r = 3$, $s = 8$. **б.** $y = 24$, $r = 3$, $s = 5$. **в.** $y = 40$, $r = 9$, $s = 2$. **г.** $y = 22$, $r = 9$, $s = 5$. **д.** $y = 64$, $r = 7$, $s = 10$.

5.4. **а.** $\langle 10; 4, 5\rangle$ нет ($h^{-1} = 10$, $u_1 = 6$, $u_2 = 4$, $a^{u_1} = 62$, $y^{u_2} = 25$, $v = 9 \neq 4$), $\langle 10; 7, 5\rangle$ да ($h^{-1} = 10$, $u_1 = 6$, $u_2 = 7$, $a^{u_1} = 62$, $y^{u_2} = 59$, $v = 7$), $\langle 10; 3, 8\rangle$ да ($h^{-1} = 10$, $u_1 = 3$, $u_2 = 3$, $a^{u_1} = 14$, $y^{u_2} = 64$, $v = 3$). **б.** $\langle 1; 3, 5\rangle$ да ($h^{-1} = 1$, $u_1 = 5$, $u_2 = 8$, $a^{u_1} = 40$, $y^{u_2} = 64$, $v = 3$), $\langle 1; 4, 3\rangle$ да ($h^{-1} = 1$, $u_1 = 3$, $u_2 = 7$, $a^{u_1} = 14$, $y^{u_2} = 25$, $v = 4$), $\langle 1; 4, 5\rangle$ нет ($h^{-1} = 1$, $u_1 = 5$, $u_2 = 7$, $a^{u_1} = 40$, $y^{u_2} = 25$, $v = 7 \neq 4$). **в.** $\langle 7; 7, 4\rangle$ да ($h^{-1} = 8$, $u_1 = 10$, $u_2 = 10$, $a^{u_1} = 59$, $y^{u_2} = 62$, $v = 7$), $\langle 7; 9, 2\rangle$ нет ($h^{-1} = 8$, $u_1 = 5$, $u_2 = 5$, $a^{u_1} = 40$, $y^{u_2} = 14$, $v = 2 \neq 9$), $\langle 5; 9, 2\rangle$ да ($h^{-1} = 9$, $u_1 = 7$, $u_2 = 7$, $a^{u_1} = 9$, $y^{u_2} = 22$, $v = 9$). **г.** $\langle 6; 9, 5\rangle$ да ($h^{-1} = 2$, $u_1 = 10$, $u_2 = 4$, $a^{u_1} = 59$, $y^{u_2} = 24$, $v = 9$), $\langle 8; 8, 3\rangle$ нет ($h^{-1} = 7$, $u_1 = 10$, $u_2 = 10$, $a^{u_1} = 59$, $y^{u_2} = 64$, $v = 2 \neq 8$), $\langle 7; 4, 1\rangle$ да ($h^{-1} = 8$, $u_1 = 8$, $u_2 = 1$, $a^{u_1} = 24$, $y^{u_2} = 22$, $v = 4$). **д.** $\langle 10; 7, 3\rangle$ да ($h^{-1} = 10$, $u_1 = 8$, $u_2 = 7$, $a^{u_1} = 24$, $y^{u_2} = 24$, $v = 7$), $\langle 7; 7, 10\rangle$ да ($h^{-1} = 8$, $u_1 = 3$, $u_2 = 10$, $a^{u_1} = 14$, $y^{u_2} = 22$, $v = 7$), $\langle 8; 7, 5\rangle$ нет ($h^{-1} = 7$, $u_1 = 2$, $u_2 = 6$, $a^{u_1} = 22$, $y^{u_2} = 59$, $v = 3 \neq 7$).

6.1. **а.** $\hat{n} = 103$, $\hat{s} = 52$, $r^{-1} = 24$, банкнота $\langle 11, 58\rangle$. **б.** $\hat{n} = 13$, $\hat{s} = 13$, $r^{-1} = 20$, банкнота $\langle 99, 22\rangle$. **в.** $\hat{n} = 58$, $\hat{s} = 74$, $r^{-1} = 12$, банкнота $\langle 55, 55\rangle$. **г.** $\hat{n} = 37$, $\hat{s} = 46$, $r^{-1} = 8$, банкнота $\langle 44, 11\rangle$. **д.** $\hat{n} = 49$, $\hat{s} = 70$, $r^{-1} = 4$, банкнота $\langle 77, 42\rangle$.

СПИСОК ЛИТЕРАТУРЫ

1. **Ахо А., Хопкрофт Дж., Ульман Дж.** Построение и анализ вычислительных алгоритмов. – М.: Мир, 1979. – 383 с.

2. **Введение** в криптографию / Под общ. ред. В. В. Ященко. – М.: МЦНМО: «ЧеРо», 2000. – 287 с.

3. **Виноградов И. М.** Основы теории чисел. – М.: Наука, 1972. – 402 с.

4. **Галлагер Р.** Теория информации и надежная связь. – М.: Советское радио, 1974. – 425 с.

5. **ГОСТ 28147-89.** Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования данных.

6. **ГОСТ Р34.10-94.** Информационная технология. Криптографическая защита информации. Процедуры выработки и проверки электронной цифровой подписи на базе асимметричного криптографического алгоритма.

7. **ГОСТ Р34.11-94.** Информационная технология. Криптографическая защита информации. Функция хэширования.

8. **ГОСТ Р34.10-2001.** Процессы формирования и проверки электронной цифровой подписи.

9. **Кнут Д.** Искусство программирования для ЭВМ. В 3-х томах. Т. 2. Получисленные алгоритмы. – М.: Мир, 1977. – 724 с.

10. **Рябко Б. Я.** Просто реализуемая идеальная криптографическая система // Проблемы передачи информации. – 2000. – Т. 36, № 1. – С. 90–104.

11. **Рябко Б. Я.** Быстрая нумерация комбинаторных объектов // Дискретная математика. – 1998. – Т. 10, № 2. – С. 101–119.

12. **Рябко Б. Я., Фионов А. Н.** Криптографические методы защиты информации: Учебное пособие для вузов. – М.: Горячая линия–Телеком, 2005. – 229 с.

13. **Рябко Б. Я., Фионов А. Н.** Быстрый метод полной рандомизации сообщений // Проблемы передачи информации. – 1997. – Т. 33, № 3. – С. 3–14.

14. **Рябко Б. Я., Фионов А. Н.** Эффективный метод адаптивного арифметического кодирования для источников с большими алфавитами // Проблемы передачи информации. – 1999. – Т. 35, № 4. – С. 1–14.

15. **Феллер В.** Введение в теорию вероятностей и ее приложения. В 2-х томах. – М.: Мир, 1984. – Т 1. – 527 с.

16. **Фионов А. Н.** Эффективный метод рандомизации сообщений на основе арифметического кодирования // Дискретный анализ и исследование операций. – 1997. – Т. 4, № 2. – С. 51–74.

17. **Шеннон К.** Работы по теории информации и кибернетике. – М.: ИЛ, 1963. – С. 333–369 (Теория связи в секретных системах).

18. **Diffie W., Hellman M. E.** New directions in cryptography // IEEE Transactions on Information Theory. – 1976. – V. 22. – P. 644–654.

19. **FIPS 180-2.** Secure hash standard.
см. http://csrc.nist.gov/.../fips180-2....pdf.

20. **FIPS 186-2.** Digital signature standard.
см. http://csrc.nist.gov/.../fips186-2-change1.pdf.

21. **FIPS 197.** Advanced encryption standard.
см. http://csrc.nist.gov/.../fips-197.pdf.

22. **Goldwasser S., Bellare M.** Lecture notes on cryptography.
см. http://www-cse.ucsd.edu/.../crypto-lecnotes.html.

23. **Menezes A., van Oorschot P., Vanstone S.** Handbook of Applied Cryptography. – CRC Press, 1996. – 661 p.
см. http://www.cacr.math.uwaterloo.ca/hac/.

24. **Ryabko B., Fionov A.** Efficient homophonic coding // IEEE Transactions on Information Theory. – 1999. – V. 45, N. 6. – P. 2083–2091.

25. **Ryabko B., Fionov A.** Fast and space-efficient adaptive arithmetic coding // Cryptography and Coding. – Berlin: Springer, 1999. – P. 270–279 (Lecture Notes in Computer Science; V. 1746).

26. **Schneier B.** Applied Cryptography, Second Edition: Protocols, Algorthms, and Source Code in C. – Wiley, 1996.

27. **Schneier B.** Self-study course in block cipher cryptanalysis // Cryptologia. – 2000. – V. 24, N. 1. – P. 18–34.
см. http://www.counterpane.com/self-study.html.