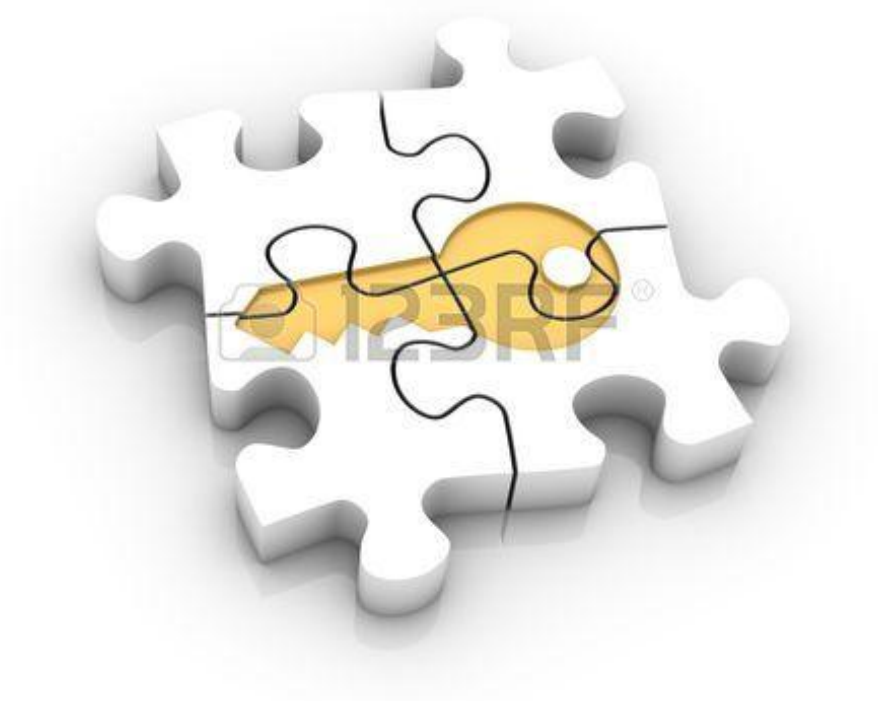
УНИВЕРСИТЕТ ИТМО

А.А. Ожиганов

КРИПТОГРАФИЯ



Санкт-Петербург
2016

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

УНИВЕРСИТЕТ ИТМО

А.А. Ожиганов

КРИПТОГРАФИЯ

Учебное пособие

Санкт-Петербург

2016

Ожиганов А.А. Криптография: учебное пособие. – СПб: Университет ИТМО, 2016. – 140 с.

Целью данного учебного пособия является ознакомление студентов с теоретическими и практическими основами криптографии. В первом разделе пособия рассматриваются математические основы криптографии. Второй раздел включает в себя теоретические и практические вопросы построения криптографических систем с секретным ключом, а также все наиболее известные методы криптографического анализа как блоковых, так и потоковых шифров. В третьем разделе пособия рассматриваются принципы построения криптографических систем с открытым ключом; достаточно подробно анализируются широко используемые в настоящее время криптосистемы, такие как RSA, криптографическая система Рабина, метод распределения ключей Диффи-Хеллмана, криптографическая система Эль-Гамаля, а также самая перспективная криптографическая система с открытым ключом на основе использования эллиптических кривых. В конце третьего раздела освещаются вопросы, связанные с применением электронных цифровых подписей на основе различных криптографических систем с открытым ключом, хеш-функции, а также криптографические протоколы.

Пособие предназначено для студентов, обучающихся в рамках направления 09.04.01 - "Информатика и вычислительная техника" при подготовке магистров по учебной программе «Безопасность вычислительных систем и сетей».

Рекомендовано Советом факультета Программной инженерии и компьютерной техники 28 апреля 2016 г., протокол № 3

УНИВЕРСИТЕТ ИТМО

Университет ИТМО – ведущий вуз России в области информационных и фотонных технологий, один из немногих российских вузов, получивших в 2009 году статус национального исследовательского университета. С 2013 года Университет ИТМО – участник программы повышения конкурентоспособности российских университетов среди ведущих мировых научно-образовательных центров, известной как проект «5 в 100». Цель Университета ИТМО – становление исследовательского университета мирового уровня, предпринимательского по типу, ориентированного на интернационализацию всех направлений деятельности.

© Университет ИТМО, 2016

©А.А.Ожиганов, 2016

Содержание

	Стр.
Предисловие	5
1. Математические основы криптографии	6
1.1. Элементы теории конечных полей	6
1.2. Основы теории чисел	15
1.3. Квадратичные вычеты и тестирование простых чисел	23
2. Криптографические системы с секретным ключом	27
2.1. Теоретические основы построения криптографических систем с секретным ключом	27
2.1.1. Модель криптографической системы	27
2.1.2. Принцип Керхгоффа	27
2.1.3. Типы криптографических систем	27
2.1.4. Абсолютно стойкие криптографические системы	28
2.1.5. Вычислительно стойкие шифры	29
2.1.6. Основные методы криптоанализа блоковых шифров	32
2.1.7. Разработка блоковых шифров, доказуемо стойких к линейному и разностному криптоанализу	34
2.1.8. Другие методы криптоанализа	36
2.2. Модификации блоковых шифров	39
2.2.1. Модификация в виде электронной кодовой книги (ECB)	39
2.2.2. Модификация с зацеплением блоков (CBC-мода)	40
2.2.3. Модификация с обратной связью по криптограмме (CFB-мода)	42
2.2.4. Модификация с обратной связью по шифрующей гамме (OFB)	43
2.3. Многократное шифрование	44
2.3.1. Метод криптоанализа двухкратного шифрования – «встреча в середине»	45
2.3.2. Трехкратное шифрование с двумя различными ключами	46
2.4. Примеры практически используемых блоковых шифров	47
2.4.1. DES	47
2.4.2. ГОСТ 28147–89	48
2.4.3. AES (Advanced Encryption Standard)	49
2.5. Способы построения и криптоанализ потоковых шифров на основе использования линейных рекуррентных регистров сдвига	52
2.5.1. Линейный рекуррентный регистр и его основные свойства	53
2.5.2. Основные способы криптоанализа потоковых шифров	58
2.6. Аутентификация сообщений	63
2.6.1. Общая структура (техника) аутентификации	64
2.6.2. Классификация систем аутентификации и характеристики их эффективности	64
2.6.3. Безусловно стойкие системы аутентификации	65
2.6.4. Вычислительно стойкие системы аутентификации	70

3. Криптографические системы с открытым ключом 74
3.1. Принципы построения криптографических систем с открытым ключом (КС ОК) 74
3.2. Построение практических криптографических систем с открытым ключом 77
3.2.1. Криптографическая система RSA 77
3.2.2. Криптографическая система Рабина 85
3.2.3. Метод распределения ключей Диффи–Хеллмана 87
3.2.4. Криптографическая система Эль-Гамаля 89
3.2.5. Построение криптографических систем на основе использования эллиптических кривых 90
3.2.6. Криптографическая система с открытым ключом Мак-Элис 95
3.3. Цифровые подписи с использованием криптографических систем с открытым ключом 100
3.3.1. Основные требования, предъявляемые к цифровым подписям 100
3.3.2. Цифровые подписи на основе различных криптографических систем с открытым ключом 101
3.3.3. Бесключевые хеш-функции 105
3.3.4. Выводы о возможности построения стойких цифровых подписей 109
3.4. Криптографические протоколы 110
3.4.1. Обзор основных криптографических протоколов 110
3.4.2. Описание процедур выполнения некоторых криптографических протоколов 113
Заключение 136
Литература 137

Предисловие

Проблема защиты информации путем ее преобразования, исключающего ее прочтение посторонним лицом, возникла в глубокой древности. При раскопках в Месопотамии был найден один из самых древних зашифрованных текстов. Он был написан клинописью на глиняной табличке и содержал рецепт глазури для покрытия гончарных изделий. Древние греки использовали для шифрования длинную узкую ленту, намотанную на посох - скиталь, и писали вдоль посоха. Прочесть текст можно было только после намотки ленты на цилиндр такого же диаметра.

Метод дешифрования придумал Аристотель, который наматывал ленту на конус, и то место, где появлялось читаемое слово или его часть, определяло неизвестный диаметр цилиндра. Юлий Цезарь использовал замену одних букв на другие, но такой шифр считается нестойким.

Грек Полибий придумал другой метод - "квадрат Полибия". Буквы были вписаны в квадрат, и каждую букву текста заменяли на пару чисел - номер строки и номер столбца. Идея Полибия была реализована в более сложных шифрах, применявшихся во время Первой мировой войны. Более сложный шифр замены создал К.Гаусс.

Итальянский математик XVI века Джироламо Кардано изобрел систему шифрования, которая применялась в военном флоте Великобритании во время Второй мировой войны: в куске картона с размеченной решеткой случайным образом прорезали отверстия, решетку клали на бумагу и писали по отверстиям текст. После снятия картона промежутки бессмысленного набора букв дописывали до псевдосмысловых фраз, что позволяло скрыть сам факт передачи секретного сообщения. Первую в мире шифрслужбу создал в XVII веке во Франции кардинал Ришелье.

Шифрование и дешифрование оставались уделом узкого круга умельцев до 1949 года, когда появилась работа Клода Шеннона "Теория связи в секретных системах", в которой проведено фундаментальное научное исследование шифров и важнейших вопросов их стойкости.

Использование криптографических методов стала особенно актуальной в настоящее время в связи с передачей по открытой сети Интернет больших объемов информации государственного, военного, коммерческого и частного характера. В связи с высокой стоимостью ущерба от потерь, разглашения и искажения информации, хранящейся в базах данных и передаваемых по локальным сетям, в современных ИС рекомендуется хранить и передавать информацию в зашифрованном виде.

1. Математические основы криптографии

Математические методы, используемые в криптографии, невозможно успешно освоить без знания таких алгебраических структур, как группы, кольца и поля. Поэтому знание и умение работать с этими объектами является необходимым условием для подготовки специалистов в области защиты информации.

Для того чтобы понять работу криптографических систем с открытым ключом и доказать их стойкость, необходимо понимать некоторый математический базис. Этот базис состоит из основ теории чисел и теории квадратичных вычетов.

В силу присущей методам криптографии специфики, большой интерес представляет множество целых чисел и различные алгебраические структуры на его базе. Поэтому основное внимание будет уделено работе с целыми числами.

1.1. Элементы теории конечных полей [1]

Определение 1.1. *Конечным полем* $GF(q)$ (или *полем Галуа*) называют конечное произвольное множество элементов с заданными между ними операциями сложения, умножения и деления. Эти операции должны обладать следующими свойствами:

1) $\forall a, b \in GF(q)$, $a + b \in GF(q)$;
2) $\forall a, b \in GF(q)$, $a \cdot b \in GF(q)$;
3) $a + b = b + a$;
4) $a \cdot b = b \cdot a$;
5) $(a + b) + c = a + (b + c) = a + b + c$;
6) $a \cdot (b + c) = a \cdot b + a \cdot c$;
7) $\exists$ элемент «O» $\in GF(q)$, $a + O = a$, $\forall a \in GF(q)$;
8) $\exists$ элемент «$-a$» $\in GF(q)$ такой, что $a + (-a) = O$, $\forall a \in GF(q)$;
9) $\exists$ элемент «e» $\in GF(q)$, $a \cdot e = a$, $\forall a \in GF(q)$;
10) $\forall a \in GF(q)$, $a \neq 0$, $\exists a^{-1}$: $a \cdot a^{-1} = e$.

Определение 1.2. *Характеристикой* «p» конечного поля $GF(q)$ называют наименьшее натуральное число, такое что

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

Утверждение 1.1. Характеристика любого конечного поля всегда является простым числом.

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

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

где $p = p' \cdot p$". Тогда

$$\underbrace{e + e + \overset{1}{\overset{\uparrow}{e}} + \cdots + e}_{p'} + \underbrace{e + e + \overset{2}{\overset{\uparrow}{e}} + \cdots + e}_{p'} + \cdots + \underbrace{e + e + \overset{p"}{\overset{\uparrow}{e}} + \cdots + e}_{p'} = 0.$$

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

$$\underbrace{a + a + a + \cdots + a}_{p"} = 0. \qquad (1.1)$$

Так как $a \neq 0$, существует элемент a^{-1}. Умножив обе части (1.1) на a^{-1}, получаем

$$\underbrace{e + e + e + \cdots + e}_{p"} = 0,$$

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

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

Действительно, рассмотрим множество $0,\ 1,\ 1+1,\ \ldots,\ \underbrace{1+1+1+\cdots+1}_{p-1}$. Это множество образует поле, удовлетворяющее всем определенным выше свойствам $GF(p)$. Выберем в качестве *e* обычную единицу (1), в качестве элементов поля – числа: $0,\ 1,\ 2,\ 3,\ \ldots,\ (p-1)$, а все действия будем рассматривать как математические операции по $\mathrm{mod}\ p$.

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

Рассмотрим некоторые примеры простых полей.

Пример 1.1. Пусть $p = 2$, тогда $GF(2)$= {0,1}, причем все операции выполняются по $\mathrm{mod}\ 2$:

$$\begin{pmatrix} 0+0=0 & 0+1=1 & 1+0=1 & 1+1=0 \\ 0\cdot 0=0 & 0\cdot 1=0 & 1\cdot 0=0 & 1\cdot 1=1 \end{pmatrix}$$

Пример 1.2. Пусть $p = 5$, тогда $GF(5)$={0, 1, 2, 3, 4}, причем все операции выполняются по $\mathrm{mod}\ 5$ (табл. 1.1а, 1.1б).

Таблица 1.1а
Сложение в поле *GF*(5)

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

Таблица 1.1б
Умножение в поле *GF*(5)

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

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

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

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

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

Так, например, число элементов поля может быть: $q = 2,\ 3,\ 4,\ 5,\ 7,\ 8,\ 9,\ 11,\ 13, \ldots$ и не может быть: $q = 6,\ 10,\ 12,\ 15, \ldots$

Конструкция конечного поля

Рассмотрим множество всех последовательностей длины n, каждая позиция которых принимает любое значение из множества $0, \ldots, p-1$. Тогда общее число последовательностей будет $q = p^n$.

Пример 1.3. Рассмотрим поле $GF(2^3)$. Тогда $n = 3$ и получаем следующие элементы поля $GF(2^3)$ в виде 8 двоичных последовательностей:

$$000,\ 001,\ 010 = \alpha,\ 011,\ 100,\ 101 = \beta,\ 110,\ 111 = \gamma.$$

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

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

Нуль в таком поле – это нулевая последовательность – 000.

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

Определение 1.3. Многочлен $f(x)$ с коэффициентами из поля $GF(p)$ называется неприводимым, если он не может быть представлен как произведение двух и более многочленов с коэффициентами из этого же поля.

Пример 1.4. Многочлен $x^2 + 1 = (x+1)\cdot(x+1) = x^2 + \underbrace{1x + 1x}_{0} + 1$ – приводимый, а $x^3 + x + 1$ – неприводимый, что легко проверяется по определению 1.3.

(Заметим, что при перемножении многочленов, действия над их коэффициентами должны выполняться по правилам действий в поле *GF*(*p*).)

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

$$00000 \to 0;\ \ 00\ldots1 \to 1;\ \ 00\ldots10 \to x;\ \ 11\ldots1 \to x^{n-1} + x^{n-2} + \cdots + 1.$$

Так, для поля $GF(2^3)$ получаем табл. 1.2.

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

Пример 1.5. Рассмотрим поле $GF(2^3)$ и неприводимый многочлен $f(x) = x^3 + x + 1$ и перемножим элементы поля:

$$\alpha = 110 \Rightarrow x^2 + x;\ \beta = 111 \Rightarrow x^2 + x + 1;$$

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

$$= \begin{array}{l|l} x^4 + x & x^3 + x = 1 \\ \hline x^4 + x^2 + x & x \\ \hline x^2 - \text{остаток} & \\ \hline \gamma = \alpha\beta = x^2 = 100 & \end{array}.$$

Таблица 1.2

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

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

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

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

$$\gcd(a(x), b(x)) = u(x) \cdot a(x) + v(x) \cdot b(x), \qquad (1.2)$$

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

Полином gcd(*a*(*x*), *b*(*x*)) = *d*(*x*) и полиномы *u*(*x*), *v*(*x*) могут быть найдены по следующему *расширенному алгоритму Евклида*, где *a*(*x*)*divb*(*x*) означает частное от деления полинома *a*(*x*) на полином *b*(*x*) (без учета остатка).

Вход: *a*(*x*), *b*(*x*).

Выход: *d*(*x*), *u*(*x*), *v*(*x*).

1. Если *b*(*x*) = 0, то установить $d(x) \leftarrow a(x), u(x) \leftarrow 1, v(x) \leftarrow 0$ и выдать *d*(*x*), *u*(*x*), *v*(*x*).

2. Установить $u_2(x) \leftarrow 1, u_1(x) \leftarrow 0, v_2(x) \leftarrow 0, v_1(x) \leftarrow 1$.

3. Если только $b(x) \neq 0$, выполнить шаги 3.1–3.4.

3.1. $g(x) \leftarrow a(x)divb(x), r(x) \leftarrow a(x) - b(x)q(x)$.

3.2. $u(x) \leftarrow u_2(x) - q(x)u_1(x), v(x) \leftarrow v_2(x) - q(x)v_1(x)$.

3.3. $a(x) \leftarrow b(x), b(x) \leftarrow r(x)$.

3.4. $u_2(x) \leftarrow u_1(x), u_1(x) \leftarrow u(x), v_2(x) \leftarrow v_1(x), v_1(x) \leftarrow v(x)$.

4. $d(x) \leftarrow a(x), u(x) \leftarrow u_2(x), v(x) \leftarrow v_2(x)$.

5. Выдать $d(x), u(x), v(x)$.

Из этого утверждения вытекает следствие, аналогичное тому, которое доказывается в теории чисел. Пусть многочлены $a(x),\ b(x) = f(x)$ над $GF(p)$ такие, что $\gcd(a(x),\ f(x)) = 1$. Тогда из (1.2) следует, что существует единственный многочлен

$$u(x)\colon a(x) \cdot u(x) = 1, \bmod (f(x)). \qquad (1.3)$$

Из (1.2) видно, что $u(x) = a(x)^{-1}$, т. е. $u(x)$ – это обратный элемент к элементу поля $a(x)$. Поскольку в конечном поле многочлен $f(x)$ всегда выбирается неприводимым, то $\gcd(a(x), f(x)) = 1$ и все ненулевые элементы поля имеют обратные элементы, которые находятся по расширенному алгоритму Евклида. Деление тогда будет выполняться следующим образом:

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

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

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

Изоморфизм – это соответствие между элементами двух множеств A и B, которое сохраняет основные операции (рис. 1.1). Изоморфные поля (множества) можно считать совпадающими с точностью до порядка обозначения их элементов.

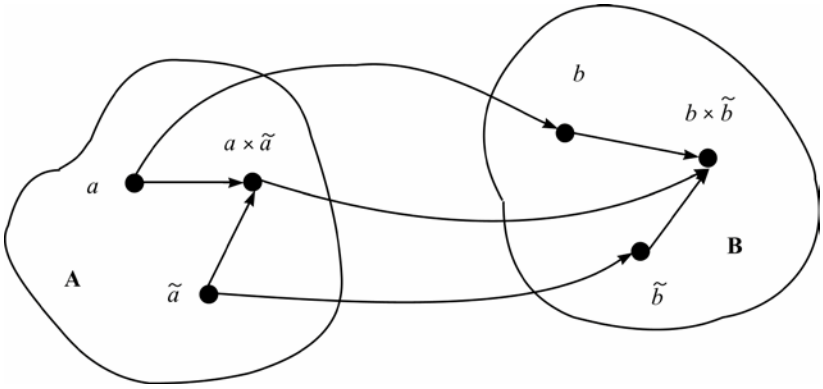


Рис. 1.1. Изоморфизм конечных полей

Основные свойства конечных полей

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

Доказательство. Рассмотрим равенство

$$(a+b)^p = a^p + \mathrm{C}_p{}^1 a^{p-1} b + C_p^2 p^{p-2} b^2 + \cdots + b^p,$$

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

Определение 1.4. *Порядком e* элемента α конечного поля $GF(q)$ называется наименьшее целое положительное число, такое что $\alpha^e = 1$. Очевидно, что порядок любого элемента конечного поля всегда будет конечен.

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

Доказательство. Докажем сначала, что всегда $\alpha^{q-1} = 1$. Для доказательства рассмотрим произведение всех ненулевых элементов поля $\alpha_1 \cdot \alpha_2 \cdot \ldots \cdot \alpha_{q-1}$. Умножим каждый из этих элементов на α. Получаем: $(\alpha_1 \cdot \alpha) \cdot (\alpha_2 \cdot \alpha) \cdot \ldots \cdot$

$(\alpha_{q-1} \cdot \alpha)$. Очевидно, что данная операция будет лишь перестановкой строки $\alpha_1 \cdot \alpha_2 \cdots \alpha_{q-1}$, т. е.:

$$\alpha_1 \cdot \alpha_2 \cdot \ldots \cdot \alpha_{q-1} = (\alpha_1 \cdot \alpha) \cdot (\alpha_2 \cdot \alpha) \cdot \ldots \cdot (\alpha_{q-1} \cdot \alpha).$$

Отсюда получаем, что

$$\alpha_1 \cdot \alpha_2 \cdot \ldots \cdot \alpha_{q-1} = \alpha^{q-1}(\alpha_1 \cdot \alpha_2 \cdot \ldots \cdot \alpha_{q-1}) \Rightarrow \alpha^{q-1} = 1,$$

а это и требовалось доказать.

Пусть e – порядок элемента α, т. е. $\alpha^e = 1$. Предположим, что $e \overline{|} q-1$ (e не делит $q-1$). Тогда $q-1 = be + r$, где $r < e$.

Рассмотрим теперь $\alpha^r = \alpha^{q-1-be} = \alpha^{q-1} \cdot (\alpha^{be})^{-1} = 1 \cdot 1$, а это приводит к противоречию, так как r не может быть порядком, поскольку $r < e$, а e – это, по определению, такое минимальное число, что $\alpha^e = 1$. Следовательно, утверждение не верно и $e|q-1$.

Можно доказать следующие свойства, относящиеся к порядкам элементов поля.

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

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

Определение 1.5. Элемент α, принадлежащий конечному полю $GF(q)$, называется *примитивным*, если его порядок равен $q-1$.

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

5. Каждое конечное поле $GF(q)$ содержит хотя бы один примитивный элемент [2].

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

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

Пример 1.6. Рассмотрим поле $GF(2^3)$. Пусть $f(x) = x^3 + x + 1$ – неприводимый многочлен, образующий это поле.

Тогда если α – корень уравнения $\alpha^3 + \alpha + 1 = 0$, который является примитивным элементом, то все элементы поля можно представить в виде степеней α (табл. 1.3).

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

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

Можно показать, что все операции в конечном поле $GF(q)$ требуют $O(\log^3 q)$ битовых операций. Возведение в k-ю степень любого элемента $a \in GF(q)$ требует

$O(\log k \log^3 q)$ битовых операций при использовании *быстрого алгоритма* с представлением числа k в двоичном виде.

Определение 1.6. Неприводимый многочлен $p(x)$ над полем $GF(p)$ называется *примитивным многочленом*, если его корень (принадлежащий $GF(p^n)$), является примитивным элементом этого поля.

Таблица 1.3

Представление элементов поля $GF(2^3)$ степенями примитивного элемента

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

Свойства многочленов над конечными полями [2]

1. Многочлен $x^{p^n} - x$ равен произведению всех нормированных неприводимых над $GF(p)$ многочленов, степени k которых делят n.

(*Нормализованный многочлен* – это многочлен, у которого коэффициент при наибольшей степени равен 1.)

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

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

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

Эквивалентное определение примитивного многочлена

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

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

Оценим число всех неприводимых и примитивных многочленов степени n над полем $GF(p)$. Обозначим через $I_p(n)$ число неприводимых нормированных многочленов степени n над полем $GF(q)$.

Тогда

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

где $\mu(d)$ – *функция Мёбиуса*:

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

Пример 1.7. Если $d = 20$, то $\mu(20) = 0$, так как $20 = 2^2 \cdot 5$, а если $d = 10$, то $\mu(10) = 1$, так как $10 = 2 \cdot 5$.

При больших величинах n существует следующая асимптотическая оценка для $I_p(n)$:

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

т. е. при больших n имеется достаточно много неприводимых многочленов.

Найдем теперь число примитивных многочленов степени n над полем $GF(p)$, обозначив его через $N_p(n)$. Тогда

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

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

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

Для чисел Мерсенна (согласно свойству функции Эйлера) справедливо равенство

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

Очевидно, что для больших n число примитивных многочленов также весьма велико.

Тесты на нахождение неприводимых и примитивных многочленов

Для нахождения неприводимых и примитивных многочленов можно использовать таблицы из [2], однако в некоторых криптосистемах неприводимые и примитивные многочлены выбираются в качестве ключей, и тогда общий метод генерирования такого ключа состоит в том, чтобы случайным образом сгенерировать многочлен $f(x)$ степени n над полем $GF(p)$, а затем проверить его неприводимость или примитивность. Если требуемое свойство не выполнилось то генерируется следующий многочлен, и так до тех пор, пока не получится положительный результат. Так как доля неприводимых и примитивных многочленов достаточно велика, число попыток будет конечно (и не слишком велико).

Для проверки многочленов $f(x)$ степени n над полем $GF(p)$ на неприводимость используют следующее их свойство. Для того чтобы данный многочлен был неприводимым, необходимо и достаточно выполнение условия

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

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

Алгоритм тестирования для больших величин 2^n выглядит тогда следующим образом [3]:

1. Установить $u(x) \leftarrow x$.
2. Для i от 1 до $[m/2]$ выполнить шаги 2.1–2.3.
   2.1. Вычислить $u(x) \leftarrow u(x)^{2^i} \bmod f(x)$.
   2.2. Вычислить $d(x) = \gcd(f(x), u(x) - x)$.
   2.3. Если $d(x) \neq 1$, то $f(x)$ – приводимый многочлен.
3. Если $d(x) = 1$ для всех i, то $f(x)$ –неприводимый многочлен.

Чтобы выполнить шаг 2.2 алгоритма, используется следующая модификация *алгоритма Евклида* для нахождения общего наибольшего делителя целых чисел. Если $g(x),\ h(x)$ – два многочлена над полем $GF(2)$, то их наибольший общий делитель $\gcd((g(x),\ h(x))$ находится повторением следующих шагов:

1. Пока $h(x) \neq 0$, выполнить шаг 1.1.
   1.1. Установить $r(x) \leftarrow g(x) \bmod h(x), g(x) \leftarrow h(x), h(x) \leftarrow r(x)$.
2. Выдать $\gcd = ((g(x))$.

Для нахождения примитивных многочленов используют следующий тест на примитивность [3].

Пусть задан неприводимый многочлен $f(x)$ степени n над полем $GF(p)$. Допустим, что существует разложение $p^n - 1$ на простые множители. Обозначим их через $r_1,\ r_2,\ \dots,\ r_t$. Если выполняется условие

$$x^{\frac{(p^n-1)}{r_i}} \bmod f(x) \neq 1 \text{ при } i = 1,\ 2,\ \dots,\ t,$$

то $f(x)$ будет примитивным. (Если n – число Мерсенна, то любой неприводимый полином будет и примитивным). Для возведения x в большую степень по модулю $f(x)$используется алгоритм, который является обобщением алгоритма быстрого возведения в степень по модулю заданного числа, изучаемого в третьей части пособия. Пусть необходимо вычислить $g(x)^k \bmod f(x)$, где показатель степени k имеет представление в виде двоичного разложения $k = \sum_{i=0}^{t} k_i 2^i$. Тогда выполняются следующие шаги:

1. Установить $s(t) \leftarrow 1$. Если $k = 0$, то выдать как результат $s(x)$.
2. Установить $G(x) \leftarrow g(x)$.
3. Если $k_0 = 1$, то установить $s(x) \leftarrow g(x)$.
4. Для i от 1 до t выполнить пп. 4.1–4.2.
   4.1. Установить $G(x) \leftarrow G(x)^2 \bmod f(x)$.
   4.2. Если $k_i = 1$, то установить $s(x) \leftarrow G(x) \cdot s(x) \bmod f(x)$.
5. Выдать как результат $s(x)$.

1.2. Основы теории чисел

Представление чисел в различных позиционных системах

Пусть имеется целое число n. Рассмотрим его представление в b-ичной системе:

$$n = (d_{k-1}d_{k-2}\ldots d_1 d_0),$$

где $0 < d_i < b-1$, $i = 0,\ 1,\ 2,\ \ldots,\ k-1$.

Тогда

$$n = d_{k-1}b^{k-1} + d_{k-1}b^{k-2} + d_1 b + d_0.$$

Пример 1.8. Для целого числа 201 получаем представление: $201 = (11001001)_2$. Число разрядов k числа n по основанию b следующее:

$$k = [\log_b n] + 1 = \left[\frac{\ln n}{\ln b}\right] + 1,$$

где $[x]$ означает нахождение целой части x. Если $x = 3{,}9$, то $[3{,}9] = 3$, если $x = 3{,}1$, то $[3{,}1] = 3$.

Битовые операции

Рассмотрим сначала *суммирование* двух чисел, заданных в двоичной форме:

	111	перенос
+	1111000	= 120
	0011110	= 30
	10010110	= 150

Вывод. Для сложения двух k-битовых чисел требуется выполнить k битовых операций (включая сложение с переносом).

Процесс *умножения* в двоичном представлении:

×	1 1 1 0 1	= 29
	0 1 1 0 1	= 13
	1 1 1 0 1	
	0 0 0 0 0	
	1 1 1 0 1	
	1 1 1 0 1	
	0 0 0 0 0	
	1 0 1 1 1 1 0 0 1	= 377

Вывод. Число битовых операций будет не более k^2, где $k = [\log_2 n] + 1$, а n – наибольшее из двух слагаемых. Тогда

$$k^2 = \left[[\log_2 n] + 1\right]^2.$$

Введем важное обозначение. Пусть имеется две функции от натуральных аргументов: $f(n)$ и $g(n)$. Тогда будем писать, что $f(n) = O\big(g(n)\big)$, или кратко $f = O(g)$, если существует такая константа C, что $f(n) \le C \cdot g(n)$ при любом n.

Пример 1.9. Пусть $f(n) = 2n^2 + 3n - 3$ и $g(n) = n^2 \Rightarrow f = O(n^2)$, так как легко показать, что $2n^2 + 3n - 3 \le 3n^2$. Используя это обозначение, можно записать, что сложность выполнения сложения двух k-битовых чисел будет $O(k)$, а умножения – $O(k^2)$. Иногда сложность выполнения операции называют *временем* ее

выполнения, полагая, что одна элементарная операция требует какого-то фиксированного машинного времени.

Считается, что задача требует *полиноминального времени* для своего решения, если существует такое целое число s, что необходимое для решения задачи число операций равно $O(k^s)$, где k – число бит, представляющих исходные данные этой задачи.

Делимость. Алгоритм Евклида

Говорят, что a *делит* b, если существует такое целое число d, что $b = a \cdot d$. Можно также сказать, что a является *делителем* b.

Каждое число a имеет не менее двух делителей: a и 1. Если других делителей у него нет, то такое число называется *простым* (p), а если они есть, то число называется *составным*.

Основная теорема арифметики гласит, что каждое натуральное (т. е. неотрицательное целое) число n может быть представлено как произведение степеней простых чисел, причем это представление единственно, т. е.

$$n = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdots p_i^{\alpha_i}, \tag{1.4}$$

где α_1, α_2, ... , α_i – целые числа, а p_1, p_2, …, p_i – простые числа.

Пример 1.10. Для числа 4200 разложение имеет вид $4200 = 2^3 \cdot 3 \cdot 5^2 \cdot 7$. Легко видеть, что при разложении вида (1.4) общее число делителей числа n равно: $(\alpha_1 + 1) \cdot (\alpha_2 + 1) \cdots (\alpha_i + 1)$.

Определение 1.7. *Наибольшим общим делителем* двух чисел a и b называется такое наибольшее число d, которое делит оба этих числа. Наибольший общий делитель двух чисел обозначается обычно как $\gcd(a, b)$.

Пример 1.11. Можно проверить, что $\gcd(12,\ 15) = 3$. Легко найти наибольший общий делитель, если известно представление чисел в виде степеней простых чисел (3.1).

Пример 1.12. Так, для чисел 4200 и 10780 получим: $4200 = 2^3 \cdot 3 \cdot 5^2 \cdot 7$, $10780 = 2^2 \cdot 5 \cdot 7^2 \cdot 11$, $\gcd(4200,\ 10780) = 2^2 \cdot 5 \cdot 7 = 140$.

Казалось бы, данный метод просто решает задачу нахождения gcd, однако разложение чисел на простые множители, т. е. так называемая *задача факторизации*, является весьма сложной и неполиномиальной задачей. Означает ли это, что нахождение gcd также является неполиномиальной задачей? К счастью (для КС ОК), это не так.

Эта задача решается в полиномиальное время с использованием известного еще с глубокой древности *алгоритма Евклида.* Пусть требуется определить $\gcd(a, b) = ?$ при $a > b$. Выполним следующие шаги:

1. Разделим a на b с остатком: $a = b \cdot q_1 + r_1$, где $r_1 < b$.
2. Разделим b на r_1 с остатком: $b = r_1 \cdot q_2 + r_2$, где $r_2 < r_1$.
3. Разделим r_1 на r_2 с остатком: $r_1 = r_2 \cdot q_3 + r_3$, где $r_3 < r_2$.

Продолжим выполнять подобные вычисления до тех пор, пока не получим на некотором этапе следующие выражения:

$$r_{n-2} = r_{n-1} \cdot q_{n-1} + r_n;\ r_{n-1} = r_n \cdot q_n.$$

Тогда получаем, что $\gcd(a, b) = r_n$.

Пример 1.13. Найдем $\gcd(1547,\ 560)$, выполняя шаги алгоритма Евклида:

$$1547 = 2 \cdot 560 + 427;\ \ 560 = 1 \cdot 427 + 133;$$
$$427 = 3 \cdot 133 + 28;\ \ 133 = 4 \cdot 28 + 21;$$
$$28 = 21 + 7;\ \ 21 = 7 \cdot 3;$$
$$\gcd(1547,\ 560) = 7.$$

Докажем, что алгоритм Евклида всегда приводит к нахождению gcd за определенное конечное число шагов.

Легко заметить, что остатки строго уменьшаются с каждым шагом. Таким образом, через конечное число шагов остаток должен стать равным 0, а следовательно, алгоритм всегда заканчивается после конечного числа шагов, причем последний остаток будет равен gcd. Действительно, если какое-то число делит как a, так и b, то оно должно делить r_1. Поскольку это число делит b и r_1, то оно должно делить и r_2, и так далее вплоть до r_n.

С другой стороны, проходя по этому алгоритму снизу вверх, легко видеть, что последний остаток должен делить все остатки. Тогда, по определению, $r_n = \gcd(a, b)$, поскольку r_n, с одной стороны, делит a и b, а с другой стороны, он делится на любой общий делитель a и b. *Найдем сложность вычисления (время вычисления) $gcd(a, b)$.*

Докажем сначала, что $r_{j+2} < \frac{1}{2} r_j$. Предположим, что $r_{j+1} \leq \frac{1}{2} r_j$. Тогда

$$r_{j+2} < r_{j+1} \leq \frac{1}{2} r_j. \qquad (1.5)$$

Если же $r_{j+1} > \frac{1}{2} r_j$, то после следующего деления получаем $r_j = 1 \cdot r_{j+1} + r_{j+2}$, отсюда следует

$$r_{j+2} = r_j - r_{j+1} < r_j - \frac{1}{2} r_j \leq \frac{1}{2} r_j,$$

что и требовалось доказать.

Возвращаясь к оценке сложности алгоритма Евклида, видим, что за каждые два шага остаток уменьшается не менее чем в два раза, и он не может быть меньше 1. Отсюда следует, что алгоритм нахождения gcd потребует не более $2[\log_2 a]$ делений, что дает оценку сложности $2(\log_2 a)_{\text{дел}} \approx O\big(\log(a)\big)$.

При выполнении каждого деления числа, участвующие в делении, не могут быть больше a, и поэтому число операций каждого деления равно $O(\log^2 a)$. В итоге получаем, что сложность алгоритма Евклида равна $O(\log^3 a)$.

Замечание. В действительности, если также учесть факт уменьшения чисел на каждом шаге, то сложность можно оценить точнее, а именно как $O(\log^2 a)$. В любом случае задача нахождения gcd является полиномиально сложной.

Утверждение 1.2. Пусть $d = \gcd(a, b)$, $a > b$. Тогда существуют такие целые числа u и v, что

$$d = u \cdot a + v \cdot b, \qquad (1.6)$$

причем сложность нахождения u и v оценивается как $O(\log^3 a)$.

Доказательство основано на использовании алгоритма Евклида для нахождения наибольшего общего делителя чисел a и b. Затем по строкам этого алгоритма поднимаемся вверх и получаем необходимое представление (1.6).

Рассмотрим идею доказательства на числовом примере. Представим $\gcd(1547,\ 560) = 7$ как линейную комбинацию чисел 1547 и 560. Для этого сначала выполним алгоритм Евклида:

$$\gcd(1547,\quad 560)) = 7;$$
$$1547 = 2 \cdot 560 + 427;$$
$$560 = 1 \cdot 427 + 133;$$
$$427 = 3 \cdot 133 + 28;$$
$$133 = 4 \cdot 28 + 21;$$
$$28 = 21 + 7;$$
$$21 = 7 \cdot 3;$$

$\gcd(1547,560)) = 7$.

Далее, следуя идее алгоритма для нахождения множителей u и v, получаем:

$$7 = 28 - 1 \cdot 21 = 28 - 1 \cdot (133 - 4 \cdot 28) = 5 \cdot 28 - 1 \cdot 133 = 5 \cdot (427 - 3 \cdot 133) - 1 \cdot 133 =$$
$$= 5 \cdot 427 - 16 \cdot 133 = 5 \cdot 427 - 16 \cdot (560 - 427 \cdot 1) = 21 \cdot 247 - 16 \cdot 560 =$$
$$= 21 \cdot (1547 - 2 \cdot 560) - 16 \cdot 560 = \underbrace{21}_{\widetilde{u}} \cdot 1547 - \underbrace{58}_{\widetilde{v}} \cdot 560.$$

Определение 1.8. Целые числа a и b называются *взаимно простыми*, если их наибольший общий делитель равен 1. Из доказанного ранее утверждения следует, что для двух взаимно простых чисел a и b можно представить их $\gcd = 1$ в следующем виде: $1 = u \cdot a + v \cdot b$, где $a > b$, причем сложность нахождения u и v равна $O(\log^3 a)$. (Данное свойство будет эффективно использоваться далее для нахождения так называемых обратных элементов.)

Определение 1.9. Пусть n – целое натуральное число, тогда *функцией Эйлера* $\varphi(n)$ называется число целых неотрицательных чисел, меньших n и взаимно простых с n, т. е.:

$$\varphi(n) = \#\{0 \le b < n; \gcd(b, n) = 1\},$$

где $\#\{X\}$ означает число элементов множества X.

Легко проверить, что $\varphi(1) = 1$, $\varphi(p) = p - 1$, если p – простое число.

Операции по числовому модулю (сравнения, конгруэнтность)

Будем полагать, что $a = b \cdot \bmod m \left(a \overset{m}{\equiv} b\right)$, или говорить, что a *сравнимо* с b по модулю m, если $(a - b)$ делится на m без остатка.

Пример 1.14. Легко проверить, что $5 = 2 \bmod 3$ (также говорят, что a *конгруэнтно* b).

Над сравнениями можно производить обычные операции: сложение, вычитание, умножение, – для которых выполняются соотношения

$$\left.\begin{matrix} a_1 \overset{m}{\equiv} b_1 \\ a_2 \overset{m}{\equiv} b_2 \end{matrix}\right\} \begin{matrix} a_1 + a_2 \overset{m}{\equiv} b_1 + b_2; \\ a_1 \cdot a_2 \overset{m}{\equiv} b_1 \cdot b_2. \end{matrix}$$

Докажем данное свойство для умножения:

$$\begin{matrix} a_1 = b_1 + m \cdot t_1; \\ a_2 = b_2 + m \cdot t_2, \end{matrix} \text{ где } \begin{cases} t_1 - \text{целое}; \\ t_2 - \text{целое}. \end{cases}$$

Тогда

$$a_1 \cdot a_2 = b_1 \cdot b_2 + b_1 \cdot m \cdot t_2 + b_2 \cdot m \cdot t_1 + m^2 \cdot t_1 \cdot t_2^2 =$$
$$= b_1 \cdot b_2 + m \cdot \underbrace{(b_1 \cdot t_2 + b_2 \cdot t_1 + m \cdot t_1 \cdot t_2)}_{\widetilde{N}} = b_1 \cdot b_2 + m \cdot N,$$

где N – целое.

Таким образом, в модульной арифметике можно рассматривать только наименьшее из чисел, входящих в сравнение, т. е. принадлежащее множеству $(0,\ 1,\ 2,\ \ldots,\ m-1)$, которое будем обозначать через Z_m.

Тогда

$$a+b=\mathrm{res}_m(a+b),\ a\cdot b=\mathrm{res}_m(a\cdot b),$$

где $\mathrm{res}_m(x)$ – остаток от деления x на m.

Пример 1.15. По приведенным выше правилам,

$$(10+13)\bmod 12=11;\ \ (5\cdot 4)\bmod 3=2.$$

Легко проверить следующие *свойства* модульной арифметики:

$$(a+b)\bmod m=(b+a)\bmod m;$$
$$(a\cdot b)\bmod m=(b\cdot a)\bmod m;$$
$$-b\ \bmod m=m-b.$$

Справедливы также следующие *свойства*:

$$\big(a+(b+c)\big)\bmod m=\big((a+b)+c\big)\bmod m;$$
$$a\cdot(b+c)\bmod m=(a\cdot b+a\cdot c)\bmod m.$$

Заметим, что в отличие от обычной арифметики, используемой в вычислительной технике, вычисления в модульной арифметике должны быть выполнены абсолютно точно, и если эти числа большие, то необходимо использовать специальные методы точных вычислений.

Рассмотрим вопрос обращения элементов в модульной арифметике. Пусть $a\in Z_m$, тогда *обратным* к a называется $x\in Z_m$, которое удовлетворяет уравнению

$$x\cdot a=1\cdot\ \bmod m. \qquad (1.7)$$

Введем для обратного элемента x обозначение $x=a^{-1}\bmod m$. Оказывается, что обратные элементы существуют не для всех чисел при заданном модуле.

Утверждение 1.3. Элемент $a\in Z_m$ имеет обратный элемент $a^{-1}\bmod m$ тогда и только тогда, когда $\gcd(a,m)=1$.

Доказательство. Предположим противное, т. е. $d=\gcd(a,m)>1$. Тогда покажем, что решение уравнения (3.4) не существует ни для одного целого $x\in Z_m$.

Действительно, если все же существует $x=a^{-1}\bmod m$, то это означает, что $ax-1=mt$. Так как d делит a и m, то d должно делить и «1», что невозможно при $d>1$.

Предположим теперь, что $\gcd(a,m)=1$.

Тогда согласно утверждению 3.1 существуют такие целые числа u и v, что $1=u\cdot a+v\cdot m$. Покажем, что $a^{-1}\bmod m=u$. Действительно, $u\cdot a-v\cdot m=1$, где $v\geq 0$, но тогда

$$u\cdot a=v\cdot m+1\ \ \Rightarrow\ \ res_m ua=1\ \ \Rightarrow\ \ u=a^{-1}\bmod m.$$

Таким образом, в случае выполнения необходимого условия ($\gcd(a,m)=1$) обратный элемент существует, и он может быть найден при помощи алгоритма для нахождения чисел u и v (со сложностью $O(\log^3 a)$).

Пример 1.16. Можно проверить самостоятельно, что справедливо выражение $160^{-1}\bmod 841=205$. Очевидно, что если модуль $m=p$ (простое число), то для

любого элемента $a \in Z_m$ существует обратный элемент, так как всегда $\gcd(a, p) = 1$. В этом случае говорят, что это множество образует конечное поле.

Легко видеть, что деление двух чисел по модулю заданного числа выполняется следующим образом: $a: b \bmod m = a \cdot b^{-1} \bmod m$. Таким образом, если m простое число, то для всех чисел из множества Z_m существует полный набор действий, т. е. сложение, вычитание, умножение и деление.

Возведение в степень по модулю

Эта операция определяется следующим образом: $c = b^n \bmod m$, $b \in Z_m$, $n \geq 0$. Простейший способ выполнения возведения в степень состоит в выполнении последовательных умножений: $c = \underbrace{(b \cdot b \cdots b)}_{n} \bmod m$. Однако если n достаточно велико, то данный способ не эффективен.

Быстрый способ возведения в степень. Представим n в двоичной форме: $n = n_0 + 2 \cdot n_1 + 4 \cdot n_2 + \cdots + 2^{k-1} \cdot n_{k-1}$, где $n_i = 0,1$, k – число двоичных разрядов в представлении n. Тогда

$$c = a^n \bmod m = a^{\sum_{i=0}^{k-1} n_i 2^i} \bmod m = \prod_{i=1}^{k-1} \left(a^{2^i}\right)^{n_i} \bmod m.$$

Откуда следует, что данный метод требует вычисления произведения, состоящего из k сомножителей, где каждый сомножитель представляет собой степени квадрата a.

В случае выполнения такого *быстрого возведения в степень* оценка сложности (в числе битовых операций) может быть выражена как

$$O\left(\underbrace{\log_2 n}_{k} \cdot \log^2 m\right).$$

Таким образом, можно полагать, что даже для весьма больших показателей степени (т. е. для n, имеющих большое число разрядов) абсолютно точное возведение числа в степень по модулю вполне возможно на обычном ПК.

Вычисление дискретного логарифма по модулю

Дискретный логарифм целого числа c по основанию b и по модулю целого числа m определяется следующим образом: это такое $x \in Z_m$, что $b^x = c \bmod m$. Тогда будем писать: $x = \log_b c \bmod m$. Такой логарифм называется также *логарифмом Зеча.*

Оказывается, что дискретный логарифм существует не всегда, например: $\log_3 15 \bmod 17 = 6$, а $\log_3 7 \bmod 13$ не существует.

Для решения задачи нахождения дискретного логарифма можно применить метод перебора, т. е. перебирать целые числа, меньшие модуля, проверяя выполнение уравнения, приведенного выше, однако при больших величинах модуля эта задача является необозримой.

Ниже приводится оценка числа операций, необходимых для вычисления дискретного логарифма, из которой следует, что такая задача не является полиномиальной, и поэтому она относится к вычислительно трудным задачам, и этот факт, как будет показано далее, является основой для построения ряда криптосистем с открытым ключом.

Малая теорема Ферма

Если p – простое число и p не делит a, то $a^{p-1} = 1\ mod\ p$.

Доказательство. Заметим, что $0,\ a,\ 2a,\ \dots,\ (p-1)\ a$ различны по $\bmod p$. В противном случае, если предположить, что $i \cdot a = j \cdot a \bmod p$, при $i \neq j$, то $(i-j)a = 0 \bmod p$ и поэтому $p|(i-j)a$. Но поскольку p не делит a и $i, j < p$, сделано неверное предположение, и тогда числа $0,\ a,\ 2a,\ \dots,\ (p-1)a$ составляют всего лишь перестановку чисел $1,\ 2,\ \dots,\ p-1$. Следовательно, справедливо следующее равенство:

$$a \cdot 2a \cdots (p-1)a = a^{p-1}(p-1)\,!\ \bmod p = (p-1)\,!\ \bmod p.$$

Отсюда следует, что $(p-1)\,!\left(a^{p-1}-1\right) = O \bmod p$. Это приводит к условию $p|a^{p-1}-1$ и поэтому $a^{p-1}-1 = O \bmod p$. Последнее равенство равносильно утверждению малой теоремы Ферма: $a^{p-1} = 1 \bmod p$.

Теорема Эйлера (обобщение теоремы Ферма)

Если $gcd(a,m) = 1$, *то* $a^{\varphi(m)} = 1 mod m$, *где* $\varphi(m)$ *– функция Эйлера.*

Теорема Ферма – это частный случай теоремы Эйлера. Действительно, если $m = p$ – простое число, то по теореме Эйлера $\varphi(p) = p-1$, что и дает утверждение теоремы Ферма: $a^{p-1} = 1 \bmod p$.

Свойство мультипликативности функции Эйлера

Если $gcd(m,n) = 1$, *то* $\varphi(m \cdot n) = \varphi(m) \cdot \varphi(n)$.

Так как целое число n может быть представлено однозначно в виде степеней простых чисел, т. е.: $n = p_1^{n_1} \cdot p_2^{n_2} \cdots p_r^{n_r}$ (где $p_1,\ p_2,\ \dots,\ p_r$ – простые числа; $n_1,\ n_2,\ \dots,\ n_r$ – целые числа), из свойства мультипликативности и из того, что

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

(это нетрудно проверить), следует

$$\varphi(n) = p_1^{\alpha_1}\left(1-\tfrac{1}{p_1}\right)p_2^{\alpha_2}\left(1-\tfrac{1}{p_2}\right)\dots p_r^{\alpha_r}\left(1-\tfrac{1}{p_r}\right) = n\prod_{p|n}\left(1-\tfrac{1}{p}\right). \qquad (1.8)$$

Утверждение 1.4. (Полезное для ускорения вычисления степени по модулю.) *Если* $gcd(a,m) = 1, n' = n\ mod\ \varphi(m)$, *то* $a^{n'} mod\ m = a^n mod\ m$.

Утверждение 1.5. (Полезное для анализа стойкости криптосистем с открытым ключом.) *Пусть* $n = p \cdot q$, *где* $p \cdot q$ *– простые числа. Тогда числа* p *и* q *можно найти, если известно* n *и* $\varphi(n) = (p-1) \cdot (q-1)$.

Доказательство. Будем рассматривать p, q как пару неизвестных целых чисел, для которых задано их произведение $p \cdot q = n$ и известна сумма, поскольку $p + q = n + 1 - \varphi(n) = p \cdot q + 1 - (p-1) \cdot (q-1) = 2b$, где b – некоторое целое число.

Два числа, сумма которых равна $2b$, а произведение равно n, являются очевидно корнями уравнения $x^2 - 2bx + n = 0$ (теорема Виета). Тогда корни квадратного уравнения и есть необходимые числа p и q:

$$p = b + \sqrt{b^2 + n^2};\quad q = b - \sqrt{b^2 + n^2}.$$

Сложность решения этого уравнения – $O(\log^3 n)$.

Китайская теорема об остатках

Пусть $gcd(m_i, m_j) = 1$ *для* $i \neq j$. *Тогда система уравнений*

$$\left.\begin{array}{l} x = a_1 \bmod m_1; \\ x = a_2 \bmod m_2; \\ \vdots \\ x = a_r \bmod m_r \end{array}\right\} \tag{1.9}$$

имеет решение, и при этом если два числа x' *и* x" *– это решения данной системы, то они удовлетворяют уравнению*

$$x' = x'' \bmod \mathrm{M}, \tag{1.10}$$

где $M = m_1 \cdot m_2 \cdots m_r$.

Доказательство. Докажем однозначность решения по $\bmod M = m_1 \cdot m_2 \cdots m_r$. Предположим, что есть два решения системы (3.6) x' и x". Обозначим $y = x' - x$", тогда *y* удовлетворяет системе

$$\left.\begin{array}{l} y = 0 \bmod m_1; \\ y = 0 \bmod m_2; \\ \vdots \\ y = 0 \bmod m_r \end{array}\right\} \Rightarrow y = 0 \bmod M,$$

так как $m_1, m_2, \ldots, m_r$ – взаимно простые. Отсюда и следует, что $x' = x'' \bmod M$.

Покажем теперь, как сконструировать хотя бы одно решение *x*.

Обозначим $M_i = \frac{M}{m_i}$. Очевидно, что $\gcd(m_i, M_i) = 1$, поэтому существует обратный элемент N_i к M_i по $\bmod m_i$, т. е. $M_i^{-1} = N_i$, $M_i \cdot N_i = 1 \bmod m_i$, который может быть найден по алгоритму Евклида для нахождения обратных элементов. Положим теперь

$$x = \sum_{i=1}^{r} a_i M_i \cdot N_i \bmod M.$$

Данное решение будет решением системы (3.6). Действительно, так как m_i делит Mj, $i \neq j$, видно, что все слагаемые будут равны нулю по $\bmod m_i$, за исключением *i*-го слагаемого.

Тогда получаем $x = a_i \underbrace{M_i N_i}_{1} \bmod m_i$, и поэтому $x = a_i \bmod m_i$, при $i = 1, 2, \ldots, r$, т. е. *x* – решение системы (3.6).

Свойства делимости для некоторых представлений чисел

1. Пусть *n*, *b* – произвольные целые числа, тогда

$$\left(b^n - 1\right) : (b - 1) = b^{n-1} + b^{n-2} + \cdots + b + 1.$$

2. Пусть $b,\ m,\ n$ – целые числа. Тогда

$$b^{mn} - 1 = \left(b^m - 1\right)\left(b^{m(n-1)} + b^{m(n-2)} + \cdots + b^m + 1\right).$$

Эти утверждения легко доказываются, что может быть предложено в качестве упражнения для закрепления материала данного раздела.

1.3. Квадратичные вычеты и тестирование простых чисел

Квадратичные вычеты

Корни из единицы. Пусть задано конечное поле $GF(q)$, $q = p^n$, определение и свойства которого описаны выше. Возникает вопрос: сколько корней m-й степени из 1 существует в этом поле? То есть нужно решить уравнение: $x^m = 1$, где m – любое целое число. Ответ дает следующее утверждение.

Утверждение 1.6. Число корней m-й степени из 1 в поле $GF(q)$, $q = p^n$, равно $\gcd(m, q-1)$. Отсюда следует, что если $\gcd(m, q-1) = 1$, то 1 является единственным корнем n-й степени из 1 в этом поле.

Квадратичные вычеты. Рассмотрим простые поля $GF(p)$, где p – простое число, а $GF(p)$ состоит из элементов: $0, 1, 2, 3, \ldots, p-1$. Предположим, что $p > 2$. Ставится вопрос: какие из элементов этого поля являются квадратами этих или других элементов этого поля?

Определение 1.10. Если $a \in GF(p)$ является квадратом некоторого элемента $b \in GF(p)$, т. е. $a = b^2$, $b \in GF(p)$, то такой элемент поля a называется *квадратичным вычетом*. Остальные элементы поля, не представимые в таком виде, называются *квадратичными невычетами*.

Пример 1.17. Если $p = 11$, то вычетами в таком поле являются 1, 4, 9, 5, 3, так как $1^2 = 1$, $2^2 = 4$, $3^2 = 9$, $4^2 = 5$, $5^2 = 3$. Элементы 2, 6, 7, 8, 10 (как легко проверить) будут невычетами.

Если записать ненулевые элементы поля $GF(p)$ как степени примитивного элемента α, $\alpha^1, \alpha^2, \alpha^3, \ldots, \alpha^{p-1} = 1$, то в этом случае квадратичные вычеты имеют вид: α^j, где j – четное число.

Чтобы определить, является ли элемент $a \in GF(p)$ квадратичным вычетом, используются *символы Лежандра*.

Определение 1.11. Символом Лежандра числа a и простого числа p называется

$$\left(\frac{a}{p}\right) = \begin{cases} 0, & \text{если } p|a; \\ +1, & \text{если } a \text{ квадратичный вычет в } GF(p); \\ -1, & \text{если } a \text{ невычет в } GF(p). \end{cases}$$

Утверждение 1.7. Символ Лежандра может быть вычислен по формуле

$$\left(\frac{a}{p}\right) = a^{(p-1)/2} \bmod p\,.$$

Однако данный метод не позволяет найти квадратный корень из a по $\bmod p$, даже если известно, что a – вычет. Известен алгоритм решения задачи $\sqrt{a} \bmod p$, если найдено некоторое другое число $b \in GF(p)$, которое дает $\left(\frac{b}{p}\right) = -1$, т. е. b – невычет. Хотя сейчас не известен полиномиально сложный алгоритм, решающий задачу нахождения невычета, однако с вероятностью 50% при случайном выборе элемента $b \in GF(p)$ будем попадать на невычет. Следовательно, несколько попыток случайного выбора b с высокой вероятностью даст невычет.

Имея в своем распоряжении метод генерирования невычетов b, можно использовать следующую конструкцию для нахождения $\sqrt{a} \bmod p$:

1) генерировать случайные числа $b \in Z_p$, $Z_p = \{0, 1, 2, 3, \ldots, p-1\}$, до тех пор, пока $b^2 - 4a$ не окажется квадратичным невычетом по $\bmod p$, т. е.

$$\left(\frac{b^2-4a}{p}\right) = -1;$$

2) найти $r = x^{(p+1)/2} \bmod (x^2 - bx + a)$ в поле $GF(p)$;

3) выдать ответ: $r, -r$ – как решение задачи $\sqrt{a} \bmod p$. Сложность нахождения $\sqrt{a} \bmod p$ составляет $O((\lg p)^3)$ битовых операций. Когда n составное число, нахождение $\sqrt{a} \bmod p$ является весьма трудной задачей, и до сих пор не известно ни одного полиномиально сложного алгоритма ее решения. Доказано, что по сложности эта задача эквивалентна задаче факторизации чисел. Если же $n = p \cdot q$, причем p и q известны, то задача извлечения $\sqrt{a} \bmod n$ решается довольно просто по алгоритму, аналогичному рассмотренному выше. Данный факт эффективно используется в криптосистемах с открытым ключом.

Генерирование простых чисел

При генерировании простых чисел можно говорить о генерировании простых и неслучайных чисел, либо о генерировании простых чисел, являющихся частью секретного ключа, и тогда их необходимо каждый раз генерировать заново и случайным образом. Типичным для решения этих задач является подход, состоящий из двух этапов:

1) генерирование случайного или псевдослучайного (если число не является секретным ключом) числа, которое по размерности удовлетворяет предъявленным требованиям;

2) проверка, является ли выбранное нечетное число простым. Если является, то оно принимается. Если же это число не является простым, тогда нужно повторять эти этапы до успешного результата.

Возникает вопрос: сколько потребуется сделать попыток (в среднем) для генерирования простого числа заданной размерности?

Ответом на данный вопрос является следующая теорема.

Теорема [4]. Пусть $\Pi(n)$ – число простых чисел, которые $\leq n$, тогда

$$\lim_{n\to\infty} \frac{\Pi(n)}{n/\ln n} = 1.$$

Из этой теоремы можно получить аппроксимацию доли нечетных l-разрядных простых чисел в виде $\frac{2}{l\ln 10}$, т. е. среднее число попыток $\bar{s}$ для генерирования l-разрядного простого числа равно $\frac{l\cdot\ln 10}{2}$. (Для доказательства последнего факта достаточно лишь заметить, что число в точности l-разрядных нечетных чисел равно $(10^l - 10^{l-1})/2$.)

Пример 1.18. Пусть $l = 100$, тогда

$$\bar{s} = \frac{l\cdot\ln(10)}{2} = \frac{100\cdot\ln(10)}{2} = 115.$$

Важнейшие тесты по проверке простоты чисел

Все тесты делятся на *детерминированные* и *вероятностные*. Детерминированные тесты дают определенный ответ, является ли данное число простым или составным. Случайные (вероятностные) тесты дают такой же ответ, но с некоторой вероятностью (обычно близкой к 1) того, что он будет правильным.

До недавнего времени (до 2002 г.) не было известно ни одного детерминированного алгоритма с полиномиальной сложностью. В 2002 г. три индийских математика нашли такой метод. Его сложность оказывается равной $O((\log n)^{12})$, хотя для специальных чисел вида $2p+1$ сложность будет значительно меньше, а именно: $O((\log n)^6)$. Ввиду значительной сложности этого алгоритма предпочтение, однако, отдается вероятностным алгоритмам, удовлетворяющим следующему условию. Если n простое, то оно всегда *проходит тест* (т. е. то, что оно простое, определяется однозначно), если же оно составное, то может случиться, что оно пройдет тест, однако вероятность такого события может быть сделана сколь угодно малой. Рассмотрим далее два важнейших примера подобных алгоритмов тестирования чисел на простоту.

Тест Ферма

Вспомним малую теорему Ферма, которая гласит, что если p простое число и p не делит a, то $a^{p-1} = 1 \bmod p$. Поэтому необходимо выполнить следующие шаги:

1. Сгенерировать тестируемое число n и выбрать параметр «секретности» t.
2. Сгенерировать случайное число a_1: $2 \le a_1 \le n-1$.
3. Вычислить $r = a_1^{n-1} \bmod n$.
4. Если $r \neq 1$, тогда n – составное число.

Если $r = 1$, то перейти к шагу 2 и повторить все то же самое с числом a_2 и так далее, вплоть до повторения t шагов. При получении $a_1^{n-1} = 1 \bmod n$, $a_2^{n-1} = 1 \bmod n$, …, $a_t^{n-1} = 1 \bmod n$, считать n простым числом.

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

Пример 1.19. Если $n = 341 = 11 \cdot 31$, то легко проверить, что $2^{340} \bmod 341 = 1$.

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

Утверждение 1.8. При использовании теста Ферма, если число n не является числом Кармайкла, вероятность ошибки тестирования будет равна $1/2^t$, где t – число шагов. Таким образом, выбирая параметр «секретности» t достаточно большим, можно обеспечить высокую надежность тестирования простых чисел.

Тест Миллера–Рабина

Пусть заданы тестируемое нечетное число n и параметр «секретности» t. Данный тест базируется на утверждении, доказываемом в теории чисел.

Утверждение 1.9. *Пусть $n = p$ нечетное простое число и пусть для него справедливо представление: $n - 1 = 2^s \cdot r$, где s, r – числа, причем r – нечетное. Пусть a – такое, что $gcd(a,n) = 1$, тогда: или $a^r = 1 \bmod n$, или $a^{2^j \cdot r} = -1 \bmod n$, где $0 \le j \le s-1$.*

С учетом данного утверждения, тест Миллера–Рабина выполняется следующими шагами:

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

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

3. Вычислить $y = a^r \bmod n$:

а) если $y = \pm 1$, то n прошло тест и возможно является простым (повторяем этот тест для другого случайно выбранного числа a);

б) если $y \neq \pm 1$, то вычисляются $y^2 \bmod n$, $y^4 \bmod n$, y^{2^j} для $j < s$ до тех пор, пока не получится -1 для некоторого j. Если такое событие происходит, повторить тест для следующего a.

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

Известно, что вероятность ошибки при использовании теста Миллера–Рабина аппроксимируется величиной $\frac{1}{4^t}$ Видно, что этот показатель значительно лучше, чем для теста Ферма, и все операции, необходимые для проведения этого теста, имеют полиномиальную сложность.

2. Криптографические системы с секретным ключом

2.1. Теоретические основы построения криптографических систем с секретным ключом

2.1.1. Модель криптографической системы

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

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

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

где M – шифруемое сообщение; E – результат шифрования (*криптограмма*); $k_{\text{ш}}$ – ключ шифрования; $k_{\text{д}}$ – ключ дешифрования; $f(\cdot)$, $g(\cdot)$ – функции, определяющие алгоритмы шифрования и дешифрования соответственно.

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

2.1.2. Принцип Керхгоффа

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

Основные типы криптографических атак:

1) только при известной криптограмме E;

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

3) при специально выбранном сообщении M и соответствующей ему криптограмме E.

2.1.3. Типы криптосистем

По типу ключа криптосистемы делятся на симметричные ($k_{\text{ш}} = k_{\text{д}} = k$) и несимметричные ($k_{\text{ш}} \neq k_{\text{д}}$). Несимметричные криптосистемы называют также криптосистемами с открытым ключом. В этой части пособия будут рассматриваться только симметричные криптосистемы.

По способу формирования криптограммы из сообщения различают блоковые криптосистемы и потоковые криптосистемы.

Криптосистема называется *блоковой*, если каждый блок сообщения определенной длины шифруется по одному и тому же правилу; блок криптограммы имеет обычно ту же длину, что и блок сообщения (рис. 2.1).

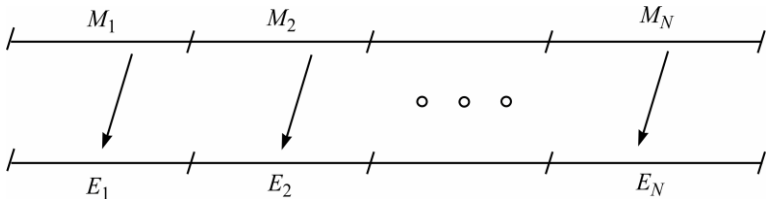


Рис. 2.1. Блоковое шифрование

Криптосистема называется *потоковой*, если каждый очередной символ криптограммы вырабатывается независимо по очередному символу сообщения и различным правилам, зависящим от номера символа и ключа (рис. 2.2).

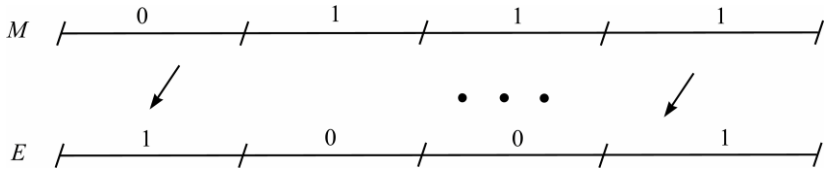


Рис. 2.2. Потоковое шифрование

Частным случаем потокового шифра является *шифр гаммирования*:

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

где E_i, M_i – i-е символы криптограммы и сообщения; γ_i – функция, зависящая от ключа k; $\oplus$ означает суммирование по mod 2 (рис. 2.3).

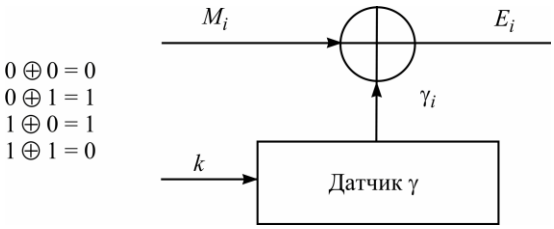


Рис. 2.3. Шифрование гаммированием

В действительности скорее можно говорить не о типе шифра, а о модификации в виде блокового или потокового шифров, поскольку, как можно увидеть далее, один и тот же алгоритм шифрования просто преобразуется как в блоковый, так и в потоковый шифр.

По стойкости шифры условно делят на два класса:

абсолютно стойкие (совершенные);

вычислительно стойкие.

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

2. 1.4. Абсолютно стойкие криптографические системы

Необходимые и достаточные условия для построения абсолютно стойких КС

Как было отмечено ранее, *абсолютно стойкие* криптосистемы (АСКС) обладают тем свойством, что если ключ не известен, то знание криптограммы E не дает

никаких преимуществ, и лучшим способом их дешифрования без знания ключа является простое угадывание сообщения.

Строгое математическое определение АСКС для любых (i, j) имеет вид

$$P\left(M_i \middle| E_j\right) = P(M_i),$$

где $P\left(M_i \middle| E_j\right)$ – условная вероятность того, что именно сообщение M_i было зашифровано криптограммой E_j; $P(M_i)$ – априорная (при неизвестной криптограмме) вероятность присутствия сообщения M_i.

Эквивалентное определение можно записать так: $I\left(E_j, M_i\right) = 0$ (где $I(E_j, M_i)$ – Шенноновская информация в криптограмме E_j о сообщении M_i).

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

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

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

Вывод. Ни при каких способах построения АСКС нельзя получить длину ключа, которая не возрастала бы пропорционально длине сообщения. Используя сжатие сообщений, можно добиться лишь уменьшения коэффициента пропорциональности. Таким образом, наилучшая АСКС имеет вид, приведенный на рис. 2.4.

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

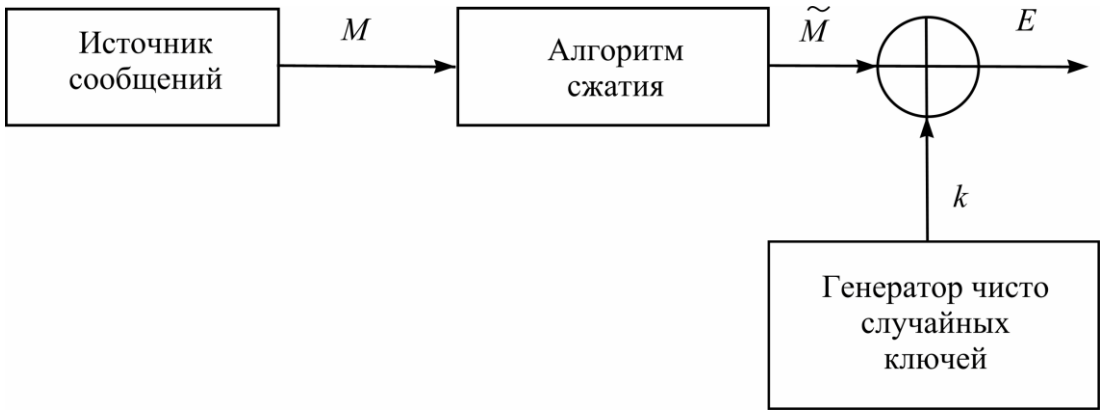


Рис. 2.4. Реализация АСКС

2.1.5. Вычислительно стойкие шифры

Определение 2.1. КС называется *вычислительно стойкой*, если наилучший возможный алгоритм дешифрования без знания ключа требует значительно большего времени или оборудования, чем может быть в распоряжении криптоаналитика.

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

Определение 2.2. КС называется *доказуемо стойкой* относительно определенного метода криптоанализа, если ее дешифрование *данным методом* требует необозримо большого времени или объема оборудования. (Такие КС, как будет показано далее, существуют).

Способы построения вычислительно стойких блоковых шифров и их криптоанализ

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

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

Одинаковые блоки сообщений M_i и M_j всегда преобразуются в одинаковые блоки криптограмм E_i и E_j. Если это свойство является нежелательным, то используют другую модификацию того же самого блокового алгоритма шифрования (см. далее «Модификации блоковых шифров»).

Принцип построения блоковых шифров

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

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

б) *перестановки* символов в блоках;

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

Рассмотрим эти процедуры в отдельности.

Подстановки

Это нелинейные преобразования блоков в блоки такой же длины.

Пример 2.1. Блок длины $n = 3$, что соответствует отображению $x_1\, x_2\, x_3 \to y_1\, y_2\, y_3$ (рис. 2.5).

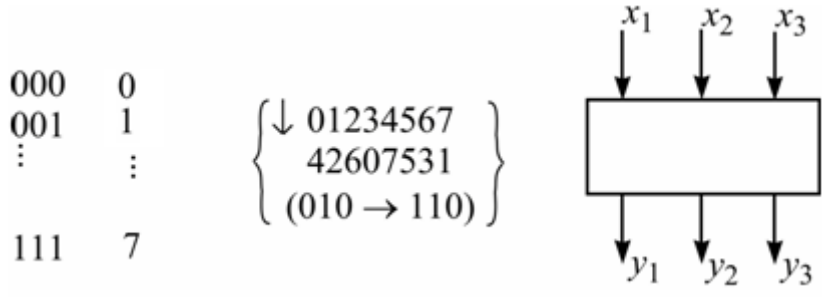


Рис. 2.5. Подстановка в подблоках длины $n = 3$

Нелинейные преобразования подстановок часто задаются фиксированной таблицей, которая может зависеть от части ключа. (Очевидно, что для шифров нельзя ограничиться только линейными преобразованиями, так как иначе был бы возможен

простой способ нахождения ключа при помощи решения линейных систем уравнений.) Такая таблица должна иметь размерность 2×2^n, что при больших величинах n оказывается нереализуемым. Именно поэтому подстановки применяются к подблокам с длиной значительно меньшей, чем длина n блока шифра, а для устранения зависимости между символами различных подблоков, которая может присутствовать в исходном сообщении, используются преобразования перестановок.

Перестановки

Это преобразование, которое показывает, на какую позицию в выходном блоке должен попасть стоящий на определенной позиции символ входной последовательности (рис. 2.6).

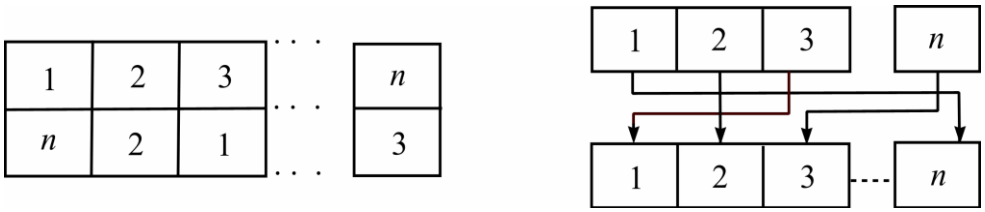


Рис. 2.6. Перестановка элементов блока

Схема Фейстеля

Для упрощения процедур шифрования и дешифрования во многих блоковых шифрах (например, в DES, см. подразд. 2.4.1) используется *схема Фейстеля*, основанная на следующих принципах:

а) каждый текущий блок делится на две равные части – левую L_i и правую R_i, где i – параметр итерации (раунда);

б) способ формирования следующих «половинок» блоков из предыдущих, определяется, как показано на рис. 2.7, где k_i – ключ i-го раунда.

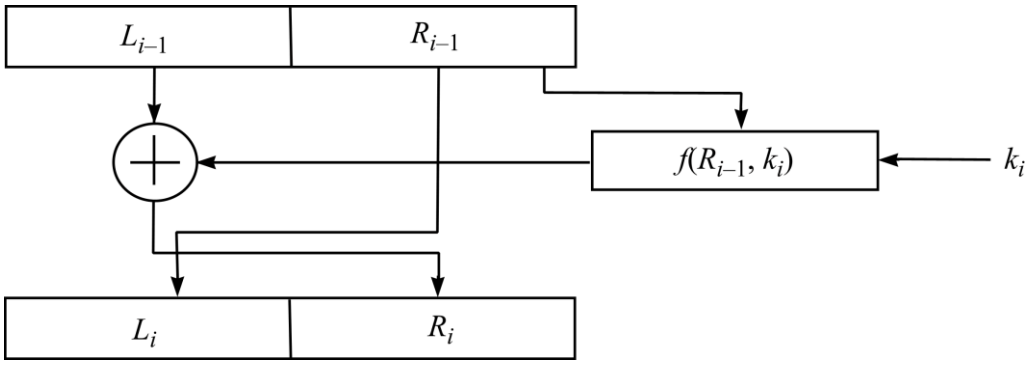


Рис. 2.7. Один раунд схемы Фейстеля

Представим это преобразование в аналитической форме:

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

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

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

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

Раундовые ключи формируются из одного секретного ключа k при помощи детерминированного алгоритма.

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

Первое преимущество схемы Фейстеля

Обратимость процедуры шифрования оказывается возможной, когда функция $f(\cdot)$ в схеме не обязательно является обратимой.

Рассмотрим выполнение процедуры шифрования по схеме Фейстеля. Один раунд схемы Фейстеля может быть представлен в следующей символической форме: $E_i = \Phi_i(M_i), \Phi_i = T \cdot V_i$, где T – перестановка половин блоков: $T(L, R) = (R, L)$, $V_i(L, R) = \big(L, L \oplus f(R, k_i)\big)$. Тогда для полного шифра получаем $E = \Phi_r, \Phi_{r-1}, \dots, \Phi_2, \Phi_1(M)$, где r – число раундов.

Дешифрование выполняется обратными операциями:

$$M = (\Phi_r \Phi_{r-1} \dots \Phi_1)^{-1}(E) = \Phi_1^{-1}\Phi_2^{-2}\dots\Phi_r^{-1}(E) = V_1 T V_2 T \dots V_r T(E) =$$
$$= V_1 \Phi_2 \Phi_3 \dots \Phi_r T(E)\,,$$

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

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

Второе преимущество схемы Фейстеля

Обе половины блока постоянно меняются местами и поэтому, несмотря на кажущуюся несимметричность, они шифруются с одинаковой стойкостью.

Существует множество вариантов шифров, построенных на основе схемы Фейстеля, которые отличаются видом нелинейной функции, числом раундов, размерами блока шифрования и способом выработки раундовых ключей по исходному ключу. В дальнейшем будет изучаться алгоритм DES, построенный по этому принципу.

Однако современные блоковые шифры предпочитают строить на несколько других принципах, чем те, на которых построена схема Фейстеля, сохраняя при этом основные принципы Шеннона. Такой класс шифров получил название *подстановочно-перестановочных* шифров (ППШ).

2.1.6. Основные методы криптоанализа блоковых шифров

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

1) полный *(тотальный)* перебор ключей;

2) *линейный*;

3) *дифференциальный (разностный)*;

4) *максимального правдоподобия*;

5) на основе *решения алгебраических (булевых) уравнений*;

6) на основе *внесения ошибок* в процедуру дешифрования.

Полный перебор ключей

При данном методе криптоанализа криптоаналитик перебирает все допустимые ключи, пытаясь дешифровать криптограмму достаточно большой длины L, равной или превосходящей расстояние единственности n_{pe}, т. е. при условии, что $L \geq n_{\text{pe}}$. Тогда первый ключ, который даст смысловой результат при дешифровании, и принимается за истинный. Однако возможности данного метода ограничены временем перебора всех или хотя бы в среднем половины ключей. Если N – длина двоичного ключа, то при $N > 128$ такой криптоанализ оказывается невозможным ни при каких вычислительных ресурсах. Действительно, полное число ключей тогда оказывается равным $2^{128} \cong 3{,}4 \cdot 10^{38}$.

Линейный криптоанализ

Линейный криптоанализ блокового шифра был предложен в 1984 г. Основная идея его состоит в использовании (существующих для любых нелинейных преобразований) *скрытых линейных уравнений*, связывающих некоторые биты входа и выхода.

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

Общее уравнение линейных связей имеет вид

$$X_{i_1} \oplus X_{i_2} \oplus \ldots \oplus X_{i_s}, \oplus Y_{j_{\ 1}} \oplus Y_{j_2} \oplus \ldots \oplus Y_{j_l} = 0, \qquad (2.1)$$

где X_i – i-й бит входа; Y_j – j-й бит выхода; $\oplus$ – сложение по mod 2.

Однако такие уравнения будут существовать не всегда, а лишь для некоторой части входных сообщений. Следовательно, можно говорить лишь о некоторой *вероятности* их выполнения. Для определения номеров $i_1, \ldots, i_s, j_1, \ldots, j_l$ в линейном уравнении (2.1) необходимо выбрать такие из них, которые обеспечивают наибольшую вероятность выполнения (2.1) по всем возможным входам.

При решении этой задачи для полного шифра будет использован следующий подход, состоящий из двух этапов:

1) исследование отдельных S-блоков (так как только они имеют нелинейность) и определение их наилучшей аппроксимации линейными уравнениями;

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

Дифференциальный (разностный) криптоанализ блоковых шифров

Дифференциальный криптоанализ (ДК) использует аномально повышенные вероятности появления некоторых разностей криптограмм для определенных разностей между открытыми сообщениями.

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

$$\Delta\bar{X} = \bar{X}'' \oplus \bar{X}'; \Delta\bar{Y} = \bar{Y}'' \oplus \bar{Y}'.$$

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

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

1) *для каждого S-блока;*

2) *для всего шифра.*

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

В работах по криптографии [9, 10] было показано, что если в качестве входов x и выходов S(x) для S-блоков, имеющих одинаковую длину m, рассматривать элементы конечного поля $GF(2^m)$ и выбрать преобразование S(x)=x^{-1}, где x – *обратный элемент* к элементу x в конечном поле $GF(2^m)$, то вероятности линейной аппроксимации P_L и дифференциальной аппроксимации P_D минимизируются и принимают значения:

$$P_L = 2^{-\frac{m}{2}}; \tag{2.2}$$

$$P_D = 2^{2-m}. \tag{2.3}$$

Доказывается также, что нелинейность такого S-блока может быть найдена так:

$$N(f) = m - 1. \tag{2.4}$$

Однако для того чтобы показать устойчивость блокового шифра к линейному и дифференциальному криптоанализу, необходимо учесть и структуру всего шифра. В частности, использование простых перестановок, встречающихся во многих типах реальных блоковых шифров, не является наилучшим способом построения блоковых шифров, устойчивых к линейному и дифференциальному криптоанализу.

Действительно, вероятности линейной и разностной аппроксимации для всего шифра подчиняются следующему неравенству [10]:

$$P_{\text{ш}(L,D)} \le P_{L(D)}^{\frac{(r\beta+2)}{2}}, \tag{2.5}$$

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

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

Рассмотрим для этого блоковый шифр, построенный на повторении так называемых SD-*преобразований* и показанный на рис. 2.8. (В аббревиатуре SD буква S означает *подстановочный* (substitution) уровень, а буква D – *рассредоточивающий* (diffusion) уровень.) В этом случае длины входов и выходов SD-преобразований, ко-

торые равны длине блока шифра, разбиваются на n подблоков длины m каждый. (Заметим, что здесь буква n используется не для обозначения длины блокового шифра, как раньше (теперь эта длина будет равна nm). Подблоки длиной m рассматриваются теперь как элементы поля $GF(2^m)$, над которыми и производится преобразование D.

Для увеличения коэффициента ветвления β необходимо вместо простых перестановок использовать более общее *линейное преобразование*, задаваемое при помощи некоторой матрицы $B_{n\times n}$, и специальным образом выбрать саму эту матрицу. Можно показать, что максимальное значение коэффициента ветвления $\beta = n + 1$, где n – число S_i-блоков, достигается в том случае, когда эта матрица $B_{n\times n}$ задается следующим образом.

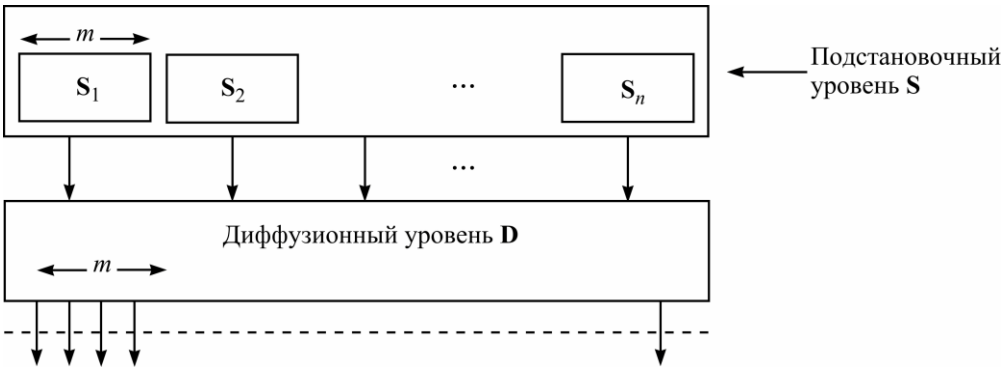


Рис. 2.8. SD-преобразование

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

$$I_n = \begin{pmatrix} 1 & 0 & 0 & \dots \\ 0 & 1 & 0 & \dots \\ 0 & 0 & 1 & \dots \\ \vdots & \vdots & \vdots & 1 \end{pmatrix};$$

$B_{n\times n}$ – нетривиальная $n \times n$ подматрица порождающей матрицы. Тогда в качестве наилучшего преобразования для диффузного уровня необходимо выбрать именно матрицу $B_{n\times n}$.

Итак, достаточно просто построить блоковый шифр, который будет устойчив как к линейному, так и к дифференциальному криптоанализу. (Такой шифр называется *доказуемо стойким* по отношению к этим видам криптоанализа). Для этого можно выбрать структуру шифра, которая показана на рис. 2.8 с неопределенными пока параметрами: m – размерность S-блоков; n – число S-блоков; r – число раундов шифра. Если после этого в качестве преобразований S взять обращение элементов в поле $GF(2^m)$, а в качестве преобразований D выбрать умножение на матрицу, взятую как нетривиальную часть порождающей матрицы кода Рида–Соломона, то для такого блокового шифра получаем из (2.5) следующие границы для вероятностей успешного осуществления линейного и дифференциального криптоанализа:

$$P_{\text{ш}(L)} \le 2^{\frac{-m(r(n+1)+2)}{4}}; \quad (2.6)$$

$$P_{\text{ш}(D)} \le 2^{\frac{(2-m)(r(n+1)+2)}{2}}. \quad (2.7)$$

Тогда число пар открытый текст/криптограмма, требуемых для анализа, будет примерно равно $P_{\text{ш}(L)}^{-2}$ для линейного и $P_{\text{ш}(D)}^{-1}$ для дифференциального криптоанализа.

Задаваясь теперь максимально допустимым числом таких пар (с точки зрения как организационной, так и вычислительной сложности), можно подобрать параметры блокового шифра *m*, *n*, *r*, которые будут удовлетворять этим требованиям.

Пример 2.2. Выберем $m = 8$, $n = 8$, $r = 5$.

Тогда по (2.6), (2.7) получаем: $P_{\text{ш}(L)} \le 2^{-94}$, $P_{\text{ш}(D)} \le 2^{-141}$, что обеспечивает достаточно высокую стойкость по отношению как к линейному, так и к дифференциальному криптоанализу.

При очевидных достоинствах общий метод построения хороших блоковых шифров обладает существенными недостатками:

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

2) структура шифра с выбором параметров по (2.6), (2.7) не гарантирует его стойкости относительно других методов криптоанализа, которые будут кратко рассмотрены далее.

2.1.8. Другие методы криптоанализа

Криптоанализ по методу максимального правдоподобия

Рассмотрим последовательно две возможные атаки:

- при неизвестном открытом сообщении;
- при известном открытом сообщении.

Первая атака

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

Предполагается, что криптоаналитик знает распределение вероятностей на сообщении $P(M)$, поэтому он может, зная структуру шифра, определить условное распределение вероятностей $P\big(E|K,P(M)\big)$ как функцию неизвестного ключа. Тогда анализ по методу максимального правдоподобия будет состоять в нахождении такого ключа $\widetilde{K}$, который обеспечит максимум этой условной вероятности, т. е.

$$\widetilde{K} = Arg\max_{K} P\big(E|K,P(M)\big).$$

Очевидно, что при нахождении max невозможно использовать переборный метод, поскольку невозможно в обозримое время перебрать все ключи. По этой причине в ряде работ предлагается использовать более эффективные методы нахождения максимумов, например градиентный метод.

Вторая атака

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

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

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

Теперь можно попытаться найти это распределение по методу максимального правдоподобия:

$$\tilde{P}(K) = Arg\max_{P(K)} P(E|P(K), M). \quad (2.8)$$

Наконец, по найденному из (2.8) вектору вероятности ключа $\tilde{P}(K) = \left(\tilde{P}_1,\ \tilde{P}_2,\ \dots,\ \tilde{P}_N\right)$ производится его квантование для нахождения бит ключа:

$$K_i = \begin{cases} 1, \text{ если } \tilde{P}(K_i = 1) \geq {}^1/_2; \\ 0, \text{ если } \tilde{P}(K_i = 1) < {}^1/_2 \end{cases}.$$

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

1) как аналитически записать функцию под аргументом максимума в (2.8) для конкретного заданного блокового шифра;

2) как найти экстремум функции в (2.8).

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

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

После получения такого представления необходимо вместо булевой переменной на входе подставить вероятность p, рассчитанную для символа 1 в этой переменной, а операции «и», «или», «нет» заменить соответственно на операции умножения, сложения и вычисления обратной вероятности $(1 - p)$. Далее, производя вычисления аналогичным образом для всех последующих раундов, можно дойти до вычисления вероятности криптограммы в (2.8). Таким образом, первая задача может быть успешно решена даже для достаточно сложных шифров.

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

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

Криптоанализ по методу решения систем нелинейных уравнений

Как было отмечено ранее, блоковый шифр может быть описан при помощи булевых функций, связывающих биты открытого сообщения, биты криптограммы и биты неизвестного ключа. Эти булевы функции можно представить в виде полиномов Жегалкина, т. е. в виде уравнений, содержащих произведения и суммы этих переменных по модулю 2. Тогда, составив систему уравнений для полного шифра при

нескольких известных блоках открытых сообщений и соответствующих им криптограмм, полученных с использованием одного и того же ключа, можно попытаться решить эту систему относительно значений бит ключа. Однако в отличие от линейной системы уравнений, для которой существуют регулярные методы решения с полиномиальной сложностью относительно ее размеров, общих методов решения нелинейных систем уравнений (даже в том случае, когда это решение заведомо существует) не известно. (Заметим также, что для усложнения описания системы линейных уравнений в современных шифрах стараются избегать задания нелинейных преобразований в «явном виде», т. е. при помощи аналитических соотношений, заменяя их или дополняя табличными описаниями.

В [11] рассматривались различные частные подходы к решению такой нелинейной системы уравнений, однако для больших длин ключа и сложных шифров не удается решить задачу по нахождению ключа в обозримое время. (Заметим, что линейный и дифференциальный криптоанализ, по существу, представляют собой специальные вероятностные методы решения такой системы уравнений.) В ряде последних исследований предлагается использовать специально разработанный алгоритм, который эффективно работает в том случае, когда число уравнений значительно больше числа переменных и каждое уравнение является «редким», т. е. в него входит относительно мало слагаемых.

Чтобы выполнить первое из этих условий, используется тот факт, что иногда уравнение, описывающее каждый S-блок, порождает несколько дополнительных уравнений относительно координат этого S-блока. Так, например, в шифре AES, который будет рассматриваться подробно в следующем разделе, в качестве преобразований, выполняемых S-блоками, используется обращение элементов в конечном поле $GF(2^m)$, где m – длина входа и выхода для S-блока. (Напомним, что такой выбор S-блока обеспечивает наилучшую стойкость шифра по отношению к линейному и дифференциальному криптоанализу.) Тогда если обозначить через x и y вход и выход S-блока соответственно, то дополнительные уравнения, описывающее этот блок, появляются при умножении обеих частей основного уравнения на x и y:

$$x^{-1} = y \Rightarrow x \cdot y = 1 \Rightarrow x = x^2 y,\ y = x \cdot y^2.$$

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

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

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

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

Криптоанализ, основанный на физической атаке

Такая атака возможна только при аппаратной реализации шифра (в том числе и при выполнении шифра на «чиповой» карте).

Сущность этой атаки состоит в том, что в алгоритм шифрования или дешифрования принудительно вносятся ошибки, приводящие к инвертированию бит на определенных местах, например когда ошибка вводится в один из битов на входе S-блока последнего раунда перед сложением с раундовым ключом.

Если теперь наблюдать результаты шифрования до и после внесения ошибок, то иногда удается определить некоторые биты ключа последнего раунда. Повторяя эту процедуру по отношению к другим битам алгоритма, можно получить постепенное вскрытие всего секретного ключа. Хотя эта процедура весьма трудоемка и требует значительных технологических ухищрений, ее нельзя полностью исключать из рассмотрения. Современные шифры строятся с учетом и подобной атаки.

Метод, основанный на использовании «слабых» ключей

Различают две основные разновидности понятия «слабый» ключ:

- неудачно сгенерированный случайный ключ;
- плохо разработанная система формирования раундовых ключей из основного ключа.

В первом случае неудачный выбор ключа может позволить криптоаналитику провести тот или иной вид криптоанализа значительно быстрее, чем это делается в общем случае. Так, в частности, для алгоритма шифрования DES (он будет рассмотрен в подразд. 2. 4.1) существует 4 слабых ключа $\widetilde{K}$, которые для 2^{32} из 2^{64} возможных входных блоков M дают вообще отсутствие всякого шифрования, т. е. выполнение условия $f(\widetilde{K}, M) = M$ [3]. Конечно, при случайном и равновероятном выборе ключей вероятность попадания на такие слабые ключи будет ничтожна мала, однако при отклонении от этого условия такие ситуации становятся возможными.

Во втором случае неудачный выбор алгоритма генерирования раундовых ключей может намного облегчить криптоаналитику задачу нахождения основного ключа.

2.2. Модификации блоковых шифров

2.2.1. Модификация в виде электронной кодовой книги (ECB)

До сих пор рассматривалась лишь одна из *модификаций* (или, иначе говоря, *мод* блоковых шифров) в виде так называемой *электронной кодовой книги*, при которой каждый блок сообщения шифровался независимо от других блоков в блок криптограммы на одном и том же ключе (рис. 2.9).

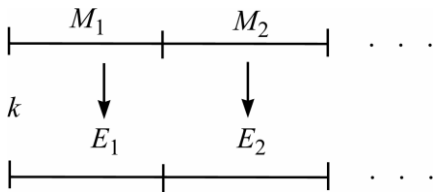


Рис. 2.9. ECB-мода блокового шифра

Данный способ шифрования имеет следующие свойства:

1) одинаковые блоки сообщения будут всегда шифроваться одинаковыми криптограммами (при условии, конечно, сохранения прежних ключей). Это, на самом де-

ле, отрицательное свойство, поскольку оно дает некоторую информацию перехватчику. Действительно, если уже было известно, что некоторым блоком криптограммы зашифровано определенное сообщение (скажем, имя пользователя), то, встретив ту же криптограмму в следующий раз, перехватчик знает, что ее отправил тот же пользователь;

2) блоки шифруются независимо, и это позволяет осуществить подмену информации без факта обнаружения такой подмены законным пользователем, имеющим правильный ключ. Для пояснения последнего отрицательного свойства рассмотрим следующий пример.

Пусть, как это иногда бывает принято, выполняется денежный перевод из одного банка в другой в зашифрованном виде. Формат этого перевода (не обязательно в точности соответствующий действительности), но отражающий суть проблемы, представлен ниже.

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

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

2.2.2. Модификация с зацеплением блоков (CBC-мода)

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

$$E_i = f_K(E_{i-1} \oplus M_i),\ i = 1,\ 2,\ \ldots, \tag{2.9}$$

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

$$M_i = E_{i-1} \oplus g_K(E_i), \tag{2.10}$$

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

Схематически шифрование представлено на рис. 2.10.

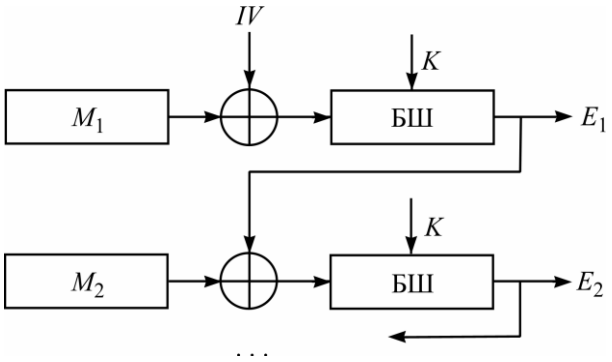


Рис. 2.10. CBC-мода блокового шифра:
БШ – блоковый шифратор

Свойства CBC-модификации:

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

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

3) единичная ошибка в блоке криптограммы E_i, как видно из алгоритма дешифрования (2.10), приводит к искажениям текущего M_i и последующего M_{i+1} блоков сообщения. В блоке M_i это может привести к большому *размножению ошибок*, и фактически блок M_i оказывается недешифруемым даже при единичной ошибке в блоке криптограммы E_i. (Заметим, что такое же свойство справедливо и для модификации в виде кодовой книги.) В следующем блоке сообщения M_{i+1} ошибки произойдут в тех же местах, где они произошли в блоке M_i, и, очевидно, без их размножения. В дальнейшем при отсутствии ошибок в блоках криптограммы все последующие блоки сообщения M_{i+2}, ... восстановятся верно;

4) для правильного дешифрования сообщения необходимо знать границы блоков. (Это требование *синхронизации по блокам*, очевидно, справедливо и для модификации в виде кодовой книги.)

Атака с подменой блоков, продемонстрированная ранее для модификации ECB, в этом случае не проходит, однако существуют более изощренные атаки, которые могут нарушить *целостность данных* в широком понимании этого термина [3].

Кроме того, если какие-либо два блока криптограмм E_i и E'_j совпадают, то в соответствии с алгоритмом дешифрования (2.10) получаем

$$E_{i-1} \oplus E'_{j-1} = M_i \oplus M_j. \qquad (2.11)$$

Поскольку сообщения являются, как правило, избыточными, то уравнение (2.11) позволяет иногда восстановить M_i или M_j. Для предотвращения такой неожиданной атаки необходимо обеспечить маловероятность совпадения криптограмм, а для этого на каждом сеансе связи следует использовать чисто случайные векторы инициализации *IV*, которые могут затем быть переданы в открытом виде для выполнения процедуры дешифрования.

2.2.3. Модификация с обратной связью по криптограмме (CFB-мода)

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

Алгоритм шифрования и дешифрования для CFB-модификации удобнее представить в виде схемы, показанной на рис. 2.11.

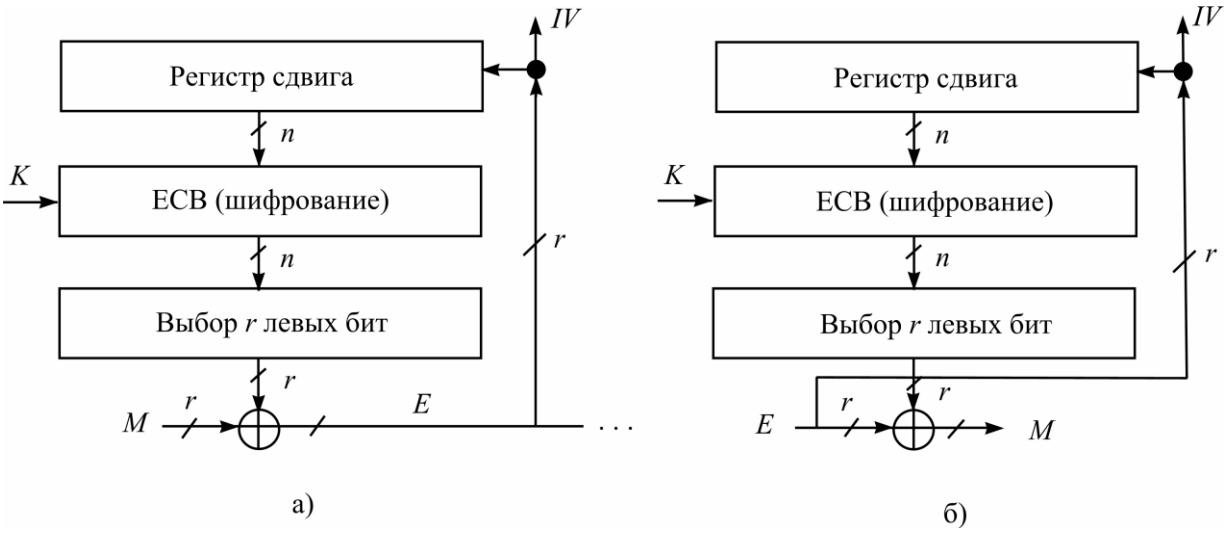


Рис. 2.11. CFB-мода блокового шифра:
а) схема шифрования; б) схема дешифрования

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

Свойства CFB-модификации:

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

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

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

4) если при шифровании оказываются неизвестными границы блоков, например за счет вставок или выпадений двоичных символов, возникающих в процессе передачи их по каналам связи, то это приведет к неправильному дешифрованию не более

чем в течение времени передачи *n*-бит (т. е. пока ошибка не покинет регистр сдвига дешифрования). Данная модификация иногда называется модификацией с *самосинхронизацией*, поскольку после ошибки синхронизации правильное дешифрование восстанавливается через определенное время без проведения специальной процедуры *ресинхронизации*;

5) алгоритм блокового шифрования используется только в *функции шифрования*. Это важно, например, для случая, когда алгоритм дешифрования является секретным, а алгоритм шифрования открытым, как это может оказаться в криптосистемах с открытым ключом.

2.2.4. Модификация с обратной связью по шифрующей гамме (OFB)

Алгоритмы шифрования и дешифрования для данной модификации показаны на рис. 2.12, где используются такие же преобразования, как и для модификации CFB, с той лишь разницей, что обратная связь (обновляющая вход блокового шифратора) поступает не как выходная криптограмма полного алгоритма, а как выход внутреннего блокового шифра. Видно, что такая модификация блокового шифра, по существу, образует *потоковый шифр.* Однако в отличие от потоковых шифров на основе линейных рекуррентных регистров, которые будут далее рассматриваться далее, такой потоковый шифр можно назвать *потоковым шифром на основе блокового шифра.*

Свойства OFB модификации:

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

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

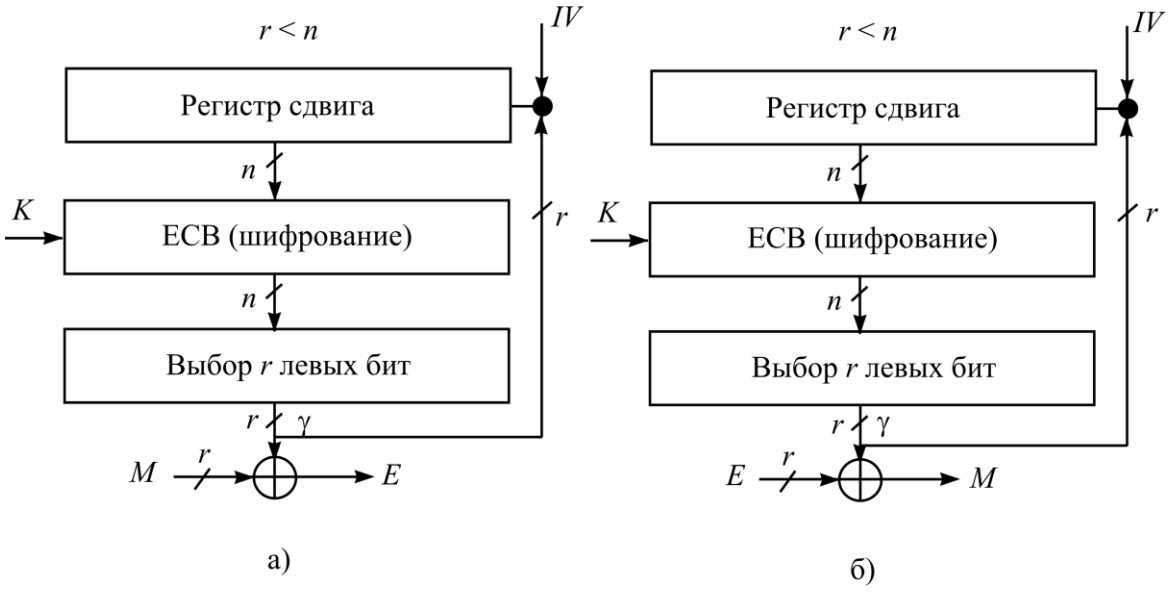


Рис. 2.12. OFB-мода блокового шифра:
а) схема шифрования; б) схема дешифрования

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

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

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

Отметим, что известна еще одна модификация блокового шифрования – *режим счетчика* [3], однако она не имеет особых преимуществ и практически редко используется.

Рекомендации по выбору модификаций блокового шифрования:

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

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

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

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

2.3. Многократное шифрование

В подразд. 2.4.1 будет рассмотрен шифр DES, который имеет всего 56 бит ключа и поэтому допускает криптоанализ в реальном времени путем перебора всех ключей с использованием специализированных устройств. Однако DES достаточно популярен в различных прикладных программах и имеет дешевую аппаратную реализацию. Именно поэтому желательно было бы повысить его стойкость, не изменяя существенно алгоритмы шифрования и дешифрования. Вполне естественной является попытка увеличения стойкости слабого алгоритма за счет применения многократного

шифрования. Попробуем начать с *двухкратного* шифрования, т. е. повторного шифрования полученных криптограмм на новых ключах, схема которого показана на рис. 2.13.

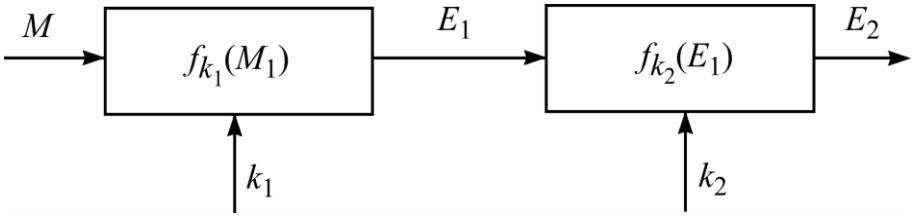


Рис. 2.13. Двухкратное шифрование

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

2.3.1. Метод криптоанализа двухкратного шифрования – «встреча в середине»

Пусть известны две пары сообщения/криптограмма: $M_1,\ M_2$ и $E_1,\ E_2$. Тогда для двухкратного шифрования будут, очевидно, справедливы следующие соотношения:

$$E_1 = f_{k_2}\left(f_{k_1}(M_1)\right); \tag{2.12}$$

$$E_2 = f_{k_2}\left(f_{k_1}(M_2)\right), \tag{2.13}$$

где f – функция шифрования; $k_1,\ k_2$ – различные ключи.

Обозначим через A результат внутреннего шифрования, т. е. $A = f_{k_1}(M_1)$. Тогда из (2.12) следует, что $g_{k_2}(E_1) = A$, где $g(\cdot)$ – функция дешифрования. Поместим в память множество значений криптограмм $f_{k_1}(M_1)$, а также множество расшифровок $g_{k_2}(E_1)$ для всех возможных ключей алгоритма шифрования DES (табл. 2.1).

Таблица 2.1

Дешифрование по методу «встречи в середине»

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

Получаем объем требуемой для этого памяти: $V_{\text{пам}} = 2 \cdot 2^{56} = 2^{57}$. Всегда найдется хотя бы одна такая пара ключей i и j (для верхней и нижней строки соответственно), что $f_i(M_1)=g_j(E_1)$. (Действительно, как следует из (2.12), это будет заведомо верно для ключей $k_1,\ k_2$, выбранных для получения криптограммы E_1 по сообщению M_1.) Далее надо в построенной выше таблице выбрать ключ $k_1 = i$, а ключ $k_2 = j$. Для того чтобы убедиться, что данная пара ключей не является ложной, проверяется выполнение равенства (2.13) на данных ключах, и если оно не выполняется, то находится следующая пара ключей $k_1,\ k_2$, которая обеспечивает равенство значений в верхней и нижней строках таблицы, и т. д.

В [3] доказывается, что для шифра DES вероятность попадания на ложные ключи, при выполнении для них (2.12), (2.13) оказывается равной $2^{-16} \cong 0{,}000015$, что говорит о практической достаточности использования всего лишь двух пар сообщение/криптограмма для правильного нахождения ключей шифрования. Действительной проблемой может показаться поиск совпадающих значений в верхней и нижней строках табл. 2.1, однако эта задача может быть решена при запоминании ее верхней строки с *сортировкой* по значениям $f_i(M_1)$. Тогда по мере получения каждого из значений $g_j(E_1)$ их можно быстро сравнивать с сортированными значениями первой строки.

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

2.3.2. Трехкратное шифрование с двумя различными ключами

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

$$E = f_{k_1}(g_{k_2}(f_{k_1}(M))). \tag{2.14}$$

Дешифрование при известных ключах $k_1,\ k_2$ выполняется очевидным образом:

$$M = g_{k_1}(f_{k_2}(g_{k_1}(M))). \tag{2.15}$$

В [3] доказывается, что применительно к шифру DES криптоанализ методом перебора ключей и атаки с t известными парами сообщение/криптограмма требует порядка $2^{120-\log t}$ операций при используемом для криптоанализа объеме памяти порядка t. Легко видеть, что даже при мало реальном выборе числа элементов памяти $t = 2^{40}$ требуемое число операций оказывается 2^{80}, т. е. практически нереализуемым.

Использование более изощренного криптоанализа со *специально выбранными* 2^{56} открытыми сообщениями и соответствующими им криптограммами позволяет найти ключи $k_{1,}\ k_2$ после выполнения порядка 2^{56} операций и задействования такого же примерно объема памяти. Хотя требуемый вычислительный ресурс в последнем случае значительно меньше, чем в предыдущем, однако выполнение условия по получению 2^{56} пар сообщение/криптограмма для сообщений, *специально выбранных* криптоаналитиком, представляется совершенно нереальным. Исходя из этого можно сделать общий вывод, что алгоритм трехкратного шифрования с двойными ключами для DES является стойким относительно наилучших переборных методов криптоанализа.

Заметим, что переход к трехкратному алгоритму шифрования DES с использованием трех различных ключей повышает, безусловно, его стойкость по отношению

к переборному методу криптоанализа [3], однако это вряд ли целесообразно, поскольку приводит и к значительному увеличению расхода ключевого материала.

2.4. Примеры практически используемых блоковых шифров

2.4.1. DES

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

Структура шифра DES

DES – это блоковый шифр с одинаковой длиной входных и выходных двоичных блоков, равной 64 бита. Он основан на структуре Фейстеля с использованием 16 раундов преобразований, в каждом из которых выполняется нелинейное преобразование, зависящее от раундового ключа. Нелинейное преобразование выполняется с использованием восьми различных S-блоков, заданных таблицами. Перестановки бит в промежуточных блоках раундов задаются одинаковыми и также определяются соответствующей таблицей. Раундовые ключи формируются из основного секретного ключа длиной 56 бит с использованием алгоритма, выполняющего его циклические сдвиги на число бит, зависящее от номера раунда. Дешифрование (согласно свойству схемы Фейстеля) выполняется тем же алгоритмом, что и шифрование, но при использовании раундовых ключей в обратном порядке.

Структурная схема алгоритма шифрования подробно описана в [12].

Стойкость шифра DES

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

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

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

го и дифференциального криптоанализа обладали бы значительным преимуществом перед полным перебором.

Таким образом, хотя известна «народная теорема» о том, что шифр DES является нестойким, но это не совсем так. В действительности, его вполне можно использовать при шифровании сообщений не слишком высокого уровня секретности, когда разумно надеяться, что к их дешифрованию не подключатся государственные структуры или корпорации, обладающие большим финансовым потенциалом. В тех же случаях, когда эти условия отсутствуют, целесообразно использовать так называемые «усиленные» алгоритмы DES, например тройной DES с двойным ключом.

Скорость шифрования DES

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

2.4.2. ГОСТ 28147–89

(Бывший советский (ныне российский) федеральный стандарт, рекомендованный к обязательному использованию в организациях, взаимодействующих с государственными учреждениями)

Структура шифра ГОСТ 28147–89

ГОСТ 28147–89 – это блоковый шифр с длиной входных и выходных блоков, равной 64 бита, основанный на структуре Фейстеля. Число раундов равно 32, а длина ключа составляет типично 256 бит, однако имеется дополнительный *долговременный ключ* длиной 512 бит.

В каждом раунде выполняются нелинейные преобразования с использованием S-блоков, задаваемых таблицами (структура таблиц не задана самим стандартом) и сложением с раундовыми ключами по $\mathrm{mod}\ 2^{32}$. (Последняя операция аналогична той, которая используется в известном шифре IDEA [3]). Раундовые ключи длиной 32 бита формируются по заданному стандартом алгоритму из основного ключа длиной 256 бит, который для этого делится на 8 блоков по 32 бита каждый. (Полное описание этого шифра легко найти в Интернете.)

Стойкость шифра ГОСТ 28147–89

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

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

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

Скорость шифрования ГОСТ 28147–89

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

2.4.3. AES (Advanced Encryption Standard) [13]

Новый американский стандарт шифрования FIPS-197

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

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

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

При оценке экспертами конкурсных проектов они подвергались изучению по следующим свойствам: стойкости, стоимости, гибкости (выполнимости в различных средах), использованию для выполнения других криптографических функций, реализуемости в смарт-картах.

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

Структура шифра AES

Данный шифр основан на принципе итерирования SD-преобразований и использует так называемую архитектуру «квадрат», т. е. все преобразования производятся в рамках одного квадрата.

Текущие данные (в том числе исходное сообщение и получаемая криптограмма) записываются по одному байту (8 бит) в каждую из 16 клеток, что дает общую длину блока шифрования, равную $8 \times 16 = 128$ бит.

Первое преобразование данного алгоритма выполняется как вычисление обратного элемента в поле $GF(2^8)$ по модулю неприводимого полинома $x^8 + x^4 + x^3 + x + 1$, что, как было показано выше, обеспечивает доказуемую устойчивость шифра по отношению к линейному и дифференциальному криптоанализу, при этом нулевой элемент поля сохраняется без преобразования (рис. 2.14).

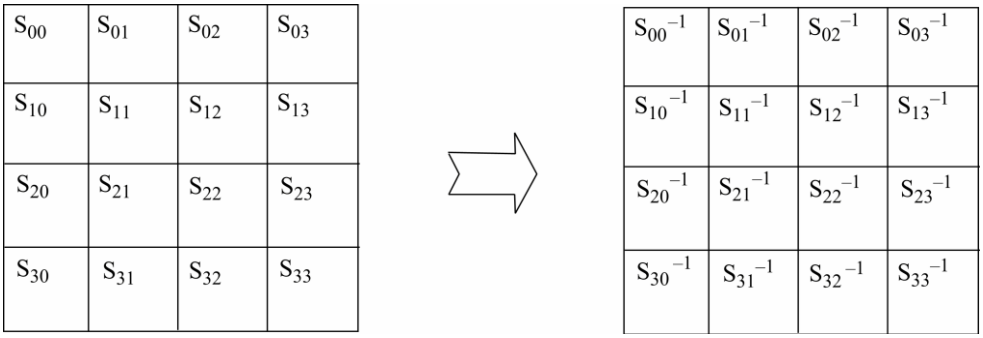


Рис. 2.14. Начальное преобразование в шифре AES

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

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

Используемая в этом преобразовании матрица и вектор-столбец сохраняются одинаковыми на всех раундах и не зависят от ключа.

Заметим, что умножение на матрицу и добавление вектора улучшают криптографические свойства шифра для случая, когда в клетках квадрата появляются нулевые элементы.

В качестве очередного преобразования используется побайтовый циклический сдвиг массива сообщений на различное число байт (клеток), показанный на рис. 2.15.

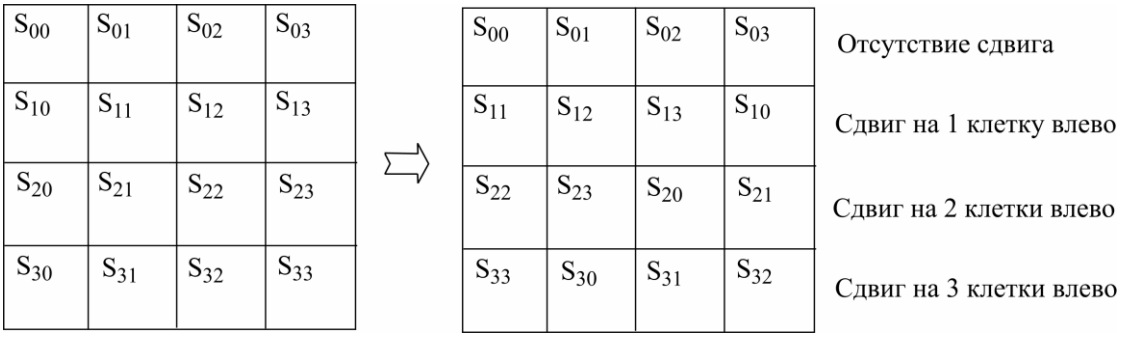


Рис. 2.15. Преобразование побайтового циклического сдвига

Следующее преобразование называется *перемешиванием столбцов.* На этом шаге каждый C-й столбец квадратной матрицы представляется как 4-мерный вектор над полем $GF(2^8)$, и далее производится умножение в этом поле, заданном непри-

водимым полиномом $x^8 + x^4 + x^3 + x + 1$, на определенную матрицу с элементами из этого же поля:

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

где элементы, показанные в этой матрице, задаются как элементы поля $GF(2^8)$ (т. е. как двоичные последовательности длины 8), что иллюстрируется следующим примером: $02 = \underbrace{0000}_{0}\underbrace{0010}_{2}$.

Наконец производится сложение с раундовыми ключами, которое выполняется просто как побитное сложение всех элементов последнего квадрата с 128 элементами раундового ключа по модулю 2. После завершения одного раунда все описанные выше операции повторяются с использованием других раундовых ключей. Раундовые ключи вырабатываются из единственного секретного ключа длиной 128, 192 или 256 бит (в зависимости от выбранного режима шифрования) при помощи специальных преобразований, включающих в себя циклические сдвиги и расширения. Точный алгоритм выработки раундовых ключей можно найти в [13]. Число раундов шифра зависит от выбранного режима его работы и изменяется в пределах от 10 до 14.

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

Особенности шифра AES

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

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

Стойкость шифра AES

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

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

Скорость шифрования AES

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

2.5. Способы построения и криптоанализ потоковых шифров на основе использования линейных рекуррентных регистров сдвига

Один из способов построения потокового шифра (моды блокового шифра с обратной связью по гамме – OFB) уже рассматривался выше. В настоящем подразделе будем изучать потоковый шифр, построенный на основе другой техники, а именно с использованием так называемых *линейных рекуррентных регистров сдвига* (ЛРР), свойства которых будут представлены далее.

Напомним, что потоковым шифром называется шифр, основанный на следующих общих преобразованиях:

шифрование

$$E_i = M_i \oplus \gamma_i(k), \quad i = 1,\ 2,\ \ldots, \tag{2.16}$$

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

дешифрование

$$M_i = E_i \oplus \gamma_i(k), \quad i = 1,\ 2,\ \ldots, \tag{2.17}$$

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

Общие свойства потокового шифра:

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

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

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

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

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

4) если на различных сеансах связи гамма повторяется, то это приводит к простому дешифрованию сообщения.

Очевидно, что наилучшим способом построения датчика гаммы было бы использование датчика чисто случайной двоичной последовательности (скажем, на основе преобразования теплового шума или эффекта радиоактивного распада). Такой датчик гаммы привел бы в идеальному шифру. Однако, как было показано ввы-

ше, это возможно только тогда, когда длина ключа пропорциональна длине сообщения. Для того чтобы построить датчик гаммы с коротким ключом, необходимо использовать двоичную *псевдослучайную последовательность* (ПСП), которая по своим свойствам будет близка к чисто случайной последовательности.

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

2.5.1. Линейный рекуррентный регистр и его основные свойства

ЛРР – это схема (или алгоритм) формирования двоичных ПСП, которая задается следующим *рекуррентным уравнением*:

$$b_j = \begin{cases} a_j, & j = 0,\ 1,\ \ldots,\ n-1; \\ \sum_{i=0}^{n-1} h_i b_{i+j-n}, & j = n,\ n+1,\ \ldots, \end{cases} \tag{2.18}$$

где $a_0,\ a_1,\ \ldots,\ a_{n-1}$ – двоичное *начальное заполнение* ЛРР; b_j – бесконечная двоичная *выходная* последовательность ЛРР; h_i – двоичные *коэффициенты*, определяющие структуру ЛРР; n – *длина* ЛРР.

(Заметим, что действия над коэффициентами и суммирование в (2.18) производятся в конечном поле $GF(2)$).

ЛРР, заданный уравнением (2.18), является, точнее говоря, двоичным ЛРР, поскольку можно очевидным образом задать и q-ичный ЛРР, если операции в (2.18) будут производиться в поле $GF(q)$, однако для построения потокового шифра вполне достаточно использовать только двоичный ЛРР (и поэтому будем в дальнейшем опускать слово «двоичный»).

Для большей наглядности рассмотрим аппаратную реализацию ЛРР, которая показана на рис. 2.16.

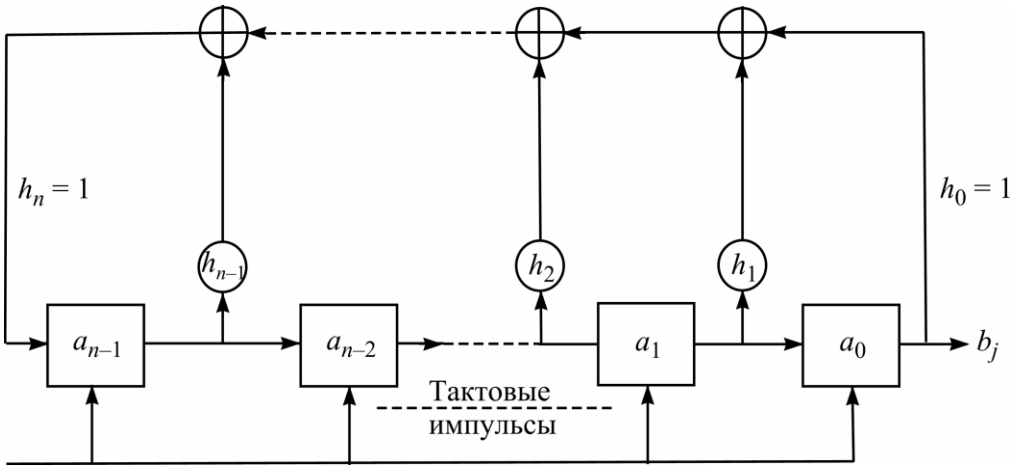


Рис. 2.16. Линейный рекуррентный регистр сдвига

Работа данной схемы достаточно очевидна: первоначально в ячейки памяти вводится двоичное начальное заполнение $a_0,\ a_1,\ \ldots,\ a_{n-1}$, затем под воздействием каждого из тактовых импульсов содержимое этих ячеек памяти сдвигается на одну

ячейку вправо; одновременно с выходов ячеек их содержимое (0 или 1) поступает на входы сумматоров по mod 2 через «коммутатор обратных связей», который пропускает к сумматорам содержимое ячеек, если соответствующий им коэффициент $h_i = 1$, и не пропускает, если $h_i = 0$; содержимое крайней правой ячейки при этом всегда поступает на выход схемы и на вход первого сумматора, а выход последнего сумматора соединен со входом первой ячейки памяти.

Таким образом, при ненулевом начальном заполнении на выходе схемы ЛРР будет появляться неограниченная по длине ненулевая двоичная последовательность, если число тактовых импульсов также не ограничено.

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

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

Для того чтобы было более удобно описывать свойства ЛРР, необходимо ввести понятие полинома обратных связей.

Определение 2.3. *Полиномом обратных связей* $h(x)$ двоичного ЛРР называется полином следующего вида:

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

где h_i – двоичный коэффициент, задающий ЛРР в (2.18), причем всегда полагаем $h_n = 1$, $h_0 = 1$.

Основные свойства ЛРР

1) Период выходной последовательности ЛРР

Докажем сначала неравенство для величины периода T любого ЛРР:

$T \le 2^n - 1$, где n – длина ЛРР.

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

Утверждение 2.1 [2]. Минимально возможный период ЛРР равен такому наименьшему целому числу n_0, для которого многочлен $x^{n_0} + 1$ делится на многочлен обратных связей ЛРР $h(x)$, причем этот период будет достигаться для начального заполнения, состоящего из коэффициентов многочлена $g(x) = (x^{n_0} + 1/h(x))$ и следующих за ними $n_0 - n - 1$ нулей. (Для других начальных заполнений ЛРР период может оказаться меньшим, чем n_0.) Если многочлен $h(x)$ является неприводимым над $GF(2)$, то при любом начальном заполнении ЛРР период выходной последовательности будет в точности равен n_0.

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

$$\begin{cases} h(x) \mid x^{2^n-1}+1; \\ h(x) \mid x^{n'}+1, \quad n' < 2^n-1. \end{cases} \tag{2.19}$$

Поэтому из утверждения 2.1 получаем, что для ЛРР, *реализованного на примитивном полиноме*, период выходной последовательности при любом начальном заполнении будет в точности равен 2^n-1. Последовательности, генерируемые такими ЛРР, называют *последовательностями максимальной длины*.

Пример 2.3. Пусть ЛРР длины 4 имеет полином обратных связей

$$h(x) = x^4 + x^3 + x + 1 = (x+1)^2(x^2+x+1).$$

Из утверждения 2.1 получаем

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

Тогда для начального заполнения 0111 выходная последовательность оказывается равной 111000 и имеет период 6.

Если теперь выбрать $h(x)$ примитивным, например $h(x) = x^4 + x + 1$, то такой ЛРР будет иметь максимальный период выходной последовательности, равный 15 для любого начального заполнения.

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

Далее будем рассматривать только ЛРР, построенные на примитивных полиномах.

2) Свойство «окна» для выходной последовательности ЛРР

Пусть $\bar{b}$ – выходная последовательность ЛРР и пусть $\bar{b}$ подпоследовательность $\bar{b}$ длины $2^n + k - 2, k \le n$.

Тогда каждая ненулевая подпоследовательность $\bar{b}$ длины k будет присутствовать в точности 2^{n-k} раз в подпоследовательности $\bar{b}$(рис. 2.17).

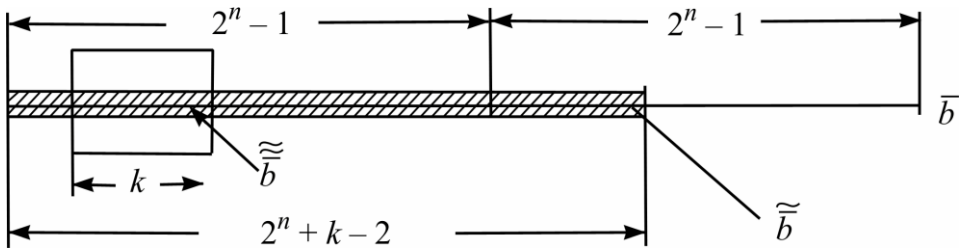


Рис. 2.17. Свойство «окна» для ЛРР

Это свойство называется свойством «окна», поскольку подпоследовательности $\bar{b}$ получаются как бы при перемещении «окна» шириной k вдоль последовательности $\bar{b}$. Если $k = n$, то все ненулевые подпоследовательности $\bar{b}$ будут появляться в «окне» точно по одному разу.

3) Свойство баланса для выходной последовательности ЛРР

Число нулей на периоде выходной последовательности ЛРР равно в точности $2^{n-1} - 1$, а число единиц 2^{n-1}.

Это свойство весьма просто доказывается. Действительно, если состояния ячеек памяти ЛРР представить десятичными числами, то легко заметить, что нечетным числам будет соответствовать появление единиц в выходной последовательности, а четным – появление нулей, но поскольку число нечетных чисел на множестве 1, 2, …, $2^n - 1$ равно в точности 2^{n-1}, а число четных чисел $2^{n-1} - 1$, то отсюда и следует доказательство данного утверждения.

Это свойство называется свойством *баланса*, поскольку при больших n оно означает, что число нулей на периоде выходной последовательности ЛРР будет практически равно числу единиц на этом периоде.

4) Свойство серий для выходной последовательности ЛРР

Различают *серии блоков* длиной k, которые задаются следующим образом:

$$\ldots 0 \underbrace{1 1 \ldots 1}_{k} 0 \ldots$$

– и *серии пробелов* длиной k:

$$\ldots 1 \underbrace{0 0 \ldots 0}_{k} 1 \ldots$$

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

5) Автокорреляционная функция выходной последовательности ЛРР

Определим автокорреляционную функцию $R(k)$ для T-периодической выходной последовательности b_i ЛРР следующим образом:

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

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

Тогда для *автокорреляционной функции* выходной последовательности ЛРР выполняется соотношение [3]

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

Из рассмотренных выше пяти свойств ЛРР следует, что ЛРР дает на выходе двоичную последовательность, которая удовлетворяет нескольким постулатам для ПСП, определенным С. Голомбом [3, 14]. Однако следующее свойство опровергает возможность использования ЛРР непосредственно в качестве датчика гаммы в потоковом шифре.

6) Предсказуемость выходной последовательности ЛРР

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

Действительно, используя рекуррентные уравнения (2.18) для нахождения выходных символов ЛРР $b_{k+n}, \ldots, b_{k+2n-1}$ через предыдущие символы, можно записать n уравнений и представить их затем в следующем матричном виде:

$$\begin{pmatrix} b_k & b_{k+1} & \ldots & b_{k+n-1} \\ b_{k+1} & b_{k+2} & \ldots & b_{k+n} \\ \ldots & \ldots & \ldots & \ldots \\ b_{k+n-1} & b_{k+n} & \ldots & b_{k+2n-2} \end{pmatrix} \cdot \begin{pmatrix} h_0 \\ h_1 \\ \vdots \\ h_{n-1} \end{pmatrix} = \begin{pmatrix} b_{k+n} \\ b_{k+n+1} \\ \vdots \\ b_{k+2n-1} \end{pmatrix}. \qquad (2.20)$$

Легко проверить, что строки матрицы, стоящей в левой части (2.20), линейно независимы, т. е. ее определитель не равен нулю. Тогда существует единственное решение этого уравнения относительно коэффициентов обратной связи $h_0 h_1 \ldots h_{n-1}$ со сложностью решения $O(n^3)$ [3].

Правда, для составления уравнений необходимо знать длину n – длину ЛРР, но это можно сделать подбором. Однако существует значительно более быстрый алгоритм решения этой задачи, известный как *алгоритм Берлекэмпа–Месси* (БМ) [3]. Данный алгоритм по не менее чем $2n$ известным последовательным выходным элементам ЛРР (и даже при неизвестной длине ЛРР) вычисляет коэффициенты обратной связи со сложностью $O(n^2)$. Это означает, что если ЛРР используется в качестве датчика гаммы, то, зная $2n$ смежных элементов сообщения и криптограммы, можно найти $2n$ элементов гаммы и вычислить ее продолжение на любую длину.

Таким образом, хотя и получен отрицательный вывод о том, что использование только ЛРР в качестве датчика шифрующей гаммы оказывается нецелесообразным, было бы весьма желательно (учитывая положительные свойства ЛРР, пп. 1–5) сохранить его в качестве датчика первичной гаммы, которая затем преобразуется при помощи нелинейных операций. Такие преобразователи называются *нелинейными узлами усложнения* (НУУ). Тогда потоковый шифр приобретает вид, показанный на рис. 2.18.

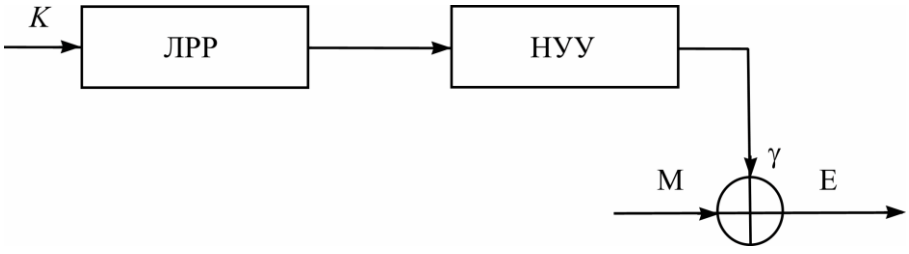


Рис. 2.18. Комбинация ЛРР и НУУ в потоковом шифре

В этом случае ключ K можно вводить как начальное состояние ЛРР или (и) как коэффициенты полинома обратных связей. (В обоих случаях длина ключа будет равна n.) Как было отмечено ранее, в качестве полиномов обратных связей необходимо всегда выбирать примитивные полиномы. При выборе ключа в виде начального заполнения этот полином является открытым, поэтому он может быть выбран заранее с использованием методов, рассмотренных ранее (или просто взят из таблиц [2, 3]). В этом случае полное число нетривиальных ключей будет, очевидно, равно $2^n - 1$, и при выборе $n \geq 128$ криптоанализ путем их полного перебора окажется невозможным. Если же ключом является полином обратных связей, то эти полиномы также должны быть примитивными, но при этом секретными. Полное число та-

ких ключей $N_2(n)$ для ЛРР длины n оказывается меньше чем $2^n - 1$ и может быть найдено из соотношения, приведенного выше, которое для двоичного ЛРР принимает следующий вид:

$$N_2(n) = \varphi(2^n - 1)/n, \quad (2.21)$$

где φ – функция Эйлера.

Если n оказывается числом Мерсенна (т. е. когда $2^n - 1$ – простое число), то (2.21) преобразуется к виду

$$N_2(n) = (2^n - 2)/n. \quad (2.22)$$

Из (2.21) и (2.22) видно, что хотя число примитивных полиномов степени n и меньше, чем 2^n, но оно также весьма велико, и при больших n число ключей будет непереборным. Поэтому для генерировании ключей в этом случае используется следующий алгоритм:

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

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

3. Если тест на примитивность не проходит, то повторяются шаги 1, 2 и так далее – вплоть до получения положительного результата.

(Поскольку число примитивных полиномов достаточно велико, число неудачных попыток не должно оказаться слишком большим.)

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

2.5.2. Основные способы криптоанализа потоковых шифров

Известно множество атак на потоковые шифры, основными из которых являются следующие:

- тотальный перебор ключей;
- использование алгоритма Берлекэмпа–Месси (БМ);
- корреляционные атаки;
- быстрые корреляционные атаки;
- аффинно-линейная аппроксимация НУУ;
- побочные атаки.

Рассмотрим каждую из этих атак более подробно.

Тотальный перебор ключей

Ключом в потоковых шифрах чаще всего является начальное заполнение ЛРР. Если длина ЛРР равна n, то общее число возможных ключей, очевидно, равно $2^n - 1$. Тогда для исключения подобной атаки это число должно быть непереборно велико. Если ключом в шифре служат отводы обратных связей ЛРР (или, что то же самое, коэффициенты полинома обратных связей), то общее число ключей будет равно числу примитивных полиномов степени n, где n – длина ЛРР. Как отмечено выше, формула (2.21), это число равно $\varphi(2^n - 1)/n$, где $\varphi(\cdot)$ – функция Эйлера. Если n – число Мерсенна, то по формуле (2.22) число таких полиномов (а следователь-

но, и число ключей) будет равно $(2^n - 2)/n$. Тогда для исключения атаки перебором ключей данные величины должны быть непереборно большими.

В потоковых шифрах можно использовать в качестве ключа одновременно начальное заполнение и отводы ЛРР. Тогда число возможных ключей очевидно возрастает.

Атака на основе использования алгоритма БМ

Пусть неизвестна структура шифра, но предполагается, что он основан на использовании одного или нескольких ЛРР и линейная эквивалентная сложность шифрующей гаммы является обозримой величиной. Тогда для выполнения атаки находится отрезок шифрующей гаммы $\gamma = M \oplus E$ достаточно большой длины N (для чего должно быть известно N бит сообщения M и соответствующих им бит криптограммы E). Далее к этой гамме применяется алгоритм БМ, что позволяет найти длину эквивалентного ЛРР, а также его отводы и начальное заполнение и, следовательно, вычислить произвольное продолжение этой гаммы. Знание продолжения гаммы позволяет, в свою очередь, дешифровать сообщение M как $M = E \oplus \gamma$ за пределами его известных N бит. Сложность выполнения алгоритма БМ составляет примерно $O(N^2)$ операций.

Для предотвращения этой атаки необходимо использовать потоковые шифры с НУУ, которые обеспечивают такую линейную эквивалентную сложность L гаммы, что нахождение $N = 2L$ элементов этой гаммы или выполнение $O(L^2)$ операций оказывается невозможным. (Примером такого шифра является генератор гаммы с управляемым тактированием, показанный на рис. 2.18, при выборе его параметров $n_1 \approx n_2 \approx n_3 \approx 128$.)

Корреляционные атаки

Предположим, что датчик шифрующей гаммы построен по схеме, показанной на рис. 2.18 Зная вид булевой функции $f(x_1, \ldots, x_m)$, обеспечивающей нелинейное преобразование выходов ЛРР, рассчитаем *теоретическую корреляцию* между двоичным выходом x_{ik} каждого из ЛРР ($i = 1, 2, \ldots, m$) и двоичным элементом гаммы y_k. (Напомним, что корреляция двух двоичных элементов x и y определяется как следующая величина: $R(x, y) = \Pr(x = y) - \Pr(x \neq y)$, где $\Pr(A)$ означает вероятность события A.) Если корреляция окажется значимо отличающейся от нуля, то это создает базу для так называемой корреляционной атаки.

Рассмотрим, для примера, генератор Джеффа, показанный еще раз на рис. 2.19, и рассчитаем одну из корреляций:

$$P(x_{1k} = y_k) = P(x_{2k} = 1) + \frac{1}{2}P(x_{2k} = 0).$$

Если

$$P(x_2 = 1) = P(x_2 = 0) = \frac{1}{2},$$

получаем

$$P(x_{1k} = y_k) = \frac{3}{4}, P(x_{1k} \neq y_k) = \frac{1}{4},$$

следовательно,

$$R(x_{1k}, y_k) = \frac{3}{4} - \frac{1}{4} = \frac{1}{2}.$$

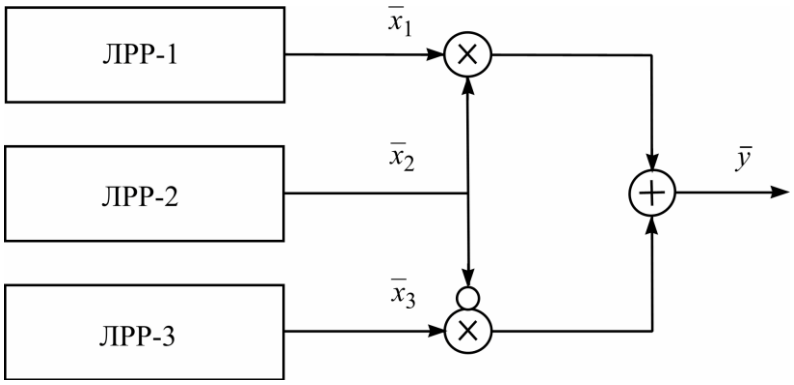


Рис. 2.19. Статическая атака на генератор Джеффа

Для выполнения корреляционного анализа поочередно перебираются ключи в каждом из ЛРР, и тот ключ, который дает максимальную корреляцию с выходом, и принимается за истинный. (Заметим, что в этом случае корреляция вычисляется как *статистическая оценка* теоретической корреляции по следующей общеизвестной формуле:

$$R(\bar{x}_i, \bar{y}) = 1/N \sum_{k=1}^{N} (-1)^{x_{ki}} (-1)^{y_k},$$

где x_{ki}, y_k – k-е элементы последовательностей $\bar{x}_i$ и $\bar{y}$ соответственно, а N – число анализируемых элементов.)

Далее таким же образом находится истинный ключ для ЛРР-3 и наконец ключ для ЛРР-2, который даст, очевидно, единичную корреляцию с исходной гаммой при правильном выборе первого и второго ключей. (Метод, который проиллюстрирован на примере генератора Джеффа, распространяется и на датчик гаммы, показанный на рис. 2.19.) Таким образом, получаем, что если при тотальном переборе необходимо перебрать $T = 2^{n_1+n_2+\ldots+n_m}$ ключей, то для корреляционной атаки число опробований будет следующим: $T' = 2^{n_1} + 2^{n_2} + \ldots + 2^{n_m}$. Очевидно, что $T' << T$, это и доказывает эффективность корреляционной атаки.

Для построения потокового шифра, защищенного от корреляционной атаки, достаточно выбрать длины входящих в него ЛРР так, чтобы число опробований T' было нереализуемым (например, при условии, что $\min n_i \geq 50$).

Если выполнение данного условия невозможно (например, из-за сложности реализации шифратора или требований высокой скорости его работы), то необходимо специальным образом выбрать булеву функцию $f(x_1, \ldots, x_m)$. Так, если эта булева функция выбрана корреляционно нечувствительной степени l, то это означает, что ненулевая корреляция будет существовать только между гаммой и объединением не менее чем l выходов ЛРР. Тогда число опробований ключей должно быть не меньше, чем $T'' = 2^{l \min \ n_i} >> T'$.

Быстрые корреляционные атаки

Это такие разновидности корреляционной атаки, которые, в отличие от «тупого» перебора, используемого в последней, применяют более эффективные методы нахождения наиболее вероятных ключей. Некоторые методы основаны на знании отводов ЛРР, образующих датчик гаммы, и состоят в нахождении корреляции с теми элементами гаммы y_j, которые с высокой вероятностью совпадают с выходами некоторых ЛРР. Другая оригинальная идея быстрой корреляционной атаки состоит в том, что

датчик гаммы аппроксимируется объединением ЛРР и *двоичного симметричного канала* с переходной вероятностью ошибки $p(e = 1) < 0{,}5$, причем ЛРР, в свою очередь, представляется как кодер *симплексного* двоичного линейного кода с длиной кодовой комбинации $2^n - 1$ и числом информационных символов n. (Такая эквивалентность ЛРР симплексному коду хорошо известна из теории кодирования [2].) Эта аппроксимация для потокового шифра показана на рис. 2.20.

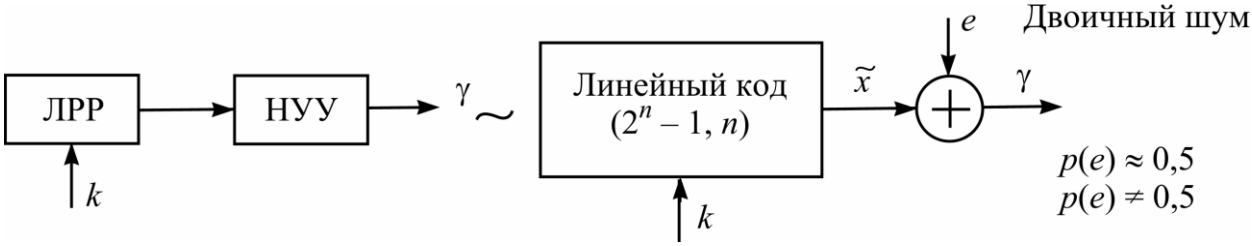


Рис. 2.20. Атака на основе эквивалентности потокового шифратора кодеру симплексного кода и канала с шумом

При данном представлении потокового шифратора задача криптоанализа сводится к исправлению ошибок e, полученных в *виртуальном канале*, и если такие ошибки будут исправлены, то оказывается возможным вычислить n информационных символов кода, которые и совпадают с ключом k, если он выбран как начальное заполнение ЛРР.

На эффективность этой атаки влияют два противоречивых факта: с одной стороны, мощный корректирующий код $(2^n - 1,\ n)$, а с другой стороны, большая вероятность ошибки виртуального канала, близкая к 0,5. Основной проблемой при практической реализации подобной атаки является нахождение эффективного метода исправления ошибок при данных условиях. Заметим, что здесь при декодировании невозможно использовать полные длины кодовых комбинаций, равные $2^n - 1$, поскольку для реальных шифраторов эти величины оказываются необозримыми.

Однако укорочение длин кодовых блоков оказывается вполне возможным, поскольку из теории кодирования следует, что требуемая длина кодового блока должна быть порядка $n/(1 - h(p(e))$, где $h(x) = -(x \log x + (1 - x)\log(1 - x))$ – энтропийная функция.

Аффинно-линейная аппроксимация НУУ

При выполнении данной атаки булева функция $f(\cdot)$ аппроксимируется как можно лучше аффинно-линейной функцией, что, конечно, приводит к значительным искажениям гаммы, но значительно уменьшает также ее ЛЭС, и поэтому становится возможной атака, основанная на алгоритме БМ, для определения ключа нового шифратора. Если шифруемое сообщение обладает большой избыточностью (например, это изображение или речь), то, даже сняв искаженную гамму (т. е. не полностью совпадающую с истинной), удается получить значительную информацию об этом сообщении.

Побочные атаки на потоковый шифр

Основная побочная атака основана на повторении шифрующей гаммы при различных сеансах связи, выполняемых на одном и том же ключе. Если такое событие происходит, то, как уже было отмечено ранее, появляется простая возможность де-

шифрования сообщений без знания ключа или использования сложных методов криптоанализа. Ввиду особой опасности этой атаки повторим еще раз ее описание. Пусть на двух сеансах связи использовался один и тот же ключ, но шифровались разные сообщения M_1 и M_2. Криптограммы, соответствующие этим сеансам связи, будут иметь вид:

$$E_1 = M_1 \oplus \gamma(k);$$
$$E_2 = M_2 \oplus \gamma(k),$$

и тогда возможна атака следующего вида:

$$E_1 \oplus E_2 = M_1 \oplus \gamma(k) \oplus M_2 \oplus \gamma(k) = M_1 \oplus M_2.$$

Поскольку, как правило, сообщения являются избыточными, а иногда и просто детерминированными, то «распутать» их сумму не представляет большого труда. (Заметим, что сумма двух сообщений по модулю какого-либо числа представляет собой пример «исторического» шифра, когда в качестве ключа выбирались известная книга и номера страниц и строк в ней, а шифрование (и дешифрование) выполнялось при помощи сложения (или вычитания) по модулю целого числа, равного объему алфавита языка, на котором было представлено шифруемое сообщение.)

Тривиальным методом исключения подобной атаки является смена ключа при каждом новом сеансе связи. Однако это требует большого расхода ключевого материала. Рассмотрим методы, позволяющие сохранять один и тот же ключ на разных сеансах связи, но изменять гамму. Если потоковый шифр использует ключ, который определяется только полиномом обратных связей ЛРР, то можно на каждом новом сеансе связи изменять начальное заполнение ЛРР. Это начальное заполнение либо выбирается детерминированным, но зависящим от номера сеанса связи и заранее согласованным передающей и принимающей сторонами, либо генерируется случайно при каждом новом сеансе связи на передающей стороне и передается в открытом виде по каналу связи на приемную сторону. Последнее обстоятельство, очевидно, не уменьшает стойкости шифра, поскольку ключом здесь является полином обратных связей ЛРР.

Возможна и более изощренная атака, которая также эксплуатирует слабость потокового шифра, состоящую в использовании суммирования по модулю 2. Для ее выполнения достаточно навязать законному пользователю хотя бы один дополнительный бит, вставленный в его предыдущее сообщение. Пусть соответственно сообщение, гамма и криптограмма первоначально имели вид: $M_1,\ M_2,\ \ldots;\ \ \gamma_1,\ \gamma_2,\ \ldots;\ \ E_1,\ E_2,\ \ldots$

Тогда после вставки известного нарушителю сообщения M' сразу после M_1 последовательности сообщения гаммы и криптограммы будут: $M_1,\ M',\ M_2,\ \ldots;\ \gamma_1,\ \gamma_2,\ \gamma_3,\ \ldots;\ E_1,\ E'_2,\ E'_3,\ \ldots$, и все сообщения сразу после вставки вскрываются нарушителем при помощи цепочки следующих тривиальных вычислений, которые даже не требуют «распутывания» сообщений:

$$\gamma_2 = E'_2 \oplus M' \rightarrow M_2 = E_2 \oplus \gamma_2 \rightarrow \gamma_3 = E'_3 \oplus M_2 \rightarrow M_3 = E_3 \oplus \gamma_3, \quad \ldots$$

Защита от такой атаки состоит в отказе от шифрования постороннего материала и изменении гаммы для новых сеансов связи.

Если ключом k шифра является начальное заполнение ЛРР, то необходимо произвести модификацию шифрования, при которой новое начальное заполнение ЛРР будет равно $IV = IV_0 \oplus k$, где k – секретный ключ, а IV_0 – двоичная последователь-

ность длины n, которая генерируется случайно и передается на приемную сторону в каждом новом сеансе связи. Такой метод также не ослабляет стойкости шифра, поскольку знание IV_0 не дает никакой информации о IV, если ключ k по-прежнему случаен.

Теперь можно сформулировать *основные требования, предъявляемые к разработке стойкого потокового шифратора*:

1) число ключей шифратора должно быть непереборно велико;

2) вырабатываемая шифратором гамма должно иметь необозримо большой период;

3) гамма должна иметь большую величину ЛЭС и хороший профиль;

4) гамма должна иметь хороший баланс 0 и 1;

5) схема шифрования должна обеспечивать устойчивость к статистическим атакам, для чего булевы функции, описывающие нелинейные преобразования выходов ЛРР, должны иметь высокий порядок корреляционной нечувствительности;

6) потоковый шифратор должен абсолютно исключать повторение гаммы, генерируемой им на одном и том же ключе.

Существует множество разработок потоковых шифров, удовлетворяющих этим требованиям. Большинство практически используемых шифров основано на использовании ЛРР с псевдослучайным тактированием. Другим направлением разработки потоковых шифров являются так называемые *ранцевые* шифры.

2.6. Аутентификация сообщений

До сих пор рассматривалась лишь одна из функций криптосистем – *конфиденциальность*, т. е. обеспечение секретности содержания передаваемых сообщений, однако криптосистемы могут выполнять и другие функции. Важнейшей из них является *аутентификация сообщений*. Аутентификация обеспечивает проверку *подлинности* (иными словами – *целостности*) сообщений, позволяя с высокой надежностью обнаружить любые их случайные или преднамеренные изменения.

Частным случаем аутентификации сообщений является *аутентификация пользователей* (или авторов) сообщений. Если в сообщении не указано явно имя пользователя, то возникает задача его косвенного и надежного определения. Эта задача называется также задачей *идентификации пользователей*.

Особым и весьма распространенным типом аутентификации является создание *аутентифицированного ключа* между двумя или более пользователями сети связи. Эта задача решается при помощи выполнения этими пользователями определенных последовательностей действий (*протокола аутентификации*), которые обеспечивают не только собственно аутентификациию, но и такие условия, например, чтобы:

- распределенный ключ был известен только законным (*легитимным*) пользователям;

- ключ пользователей действительно был известен их легитимным партнерам;

- ключ пользователей был сгенерирован заново.

Как видно из этих условий, задача по выработке аутентифицированного ключа имеет определенную специфику и должна решаться в присутствии различных преднамеренных (и часто весьма изощренных) атак, выполняемых в целях нарушения этих условий. В данном разделе будут рассмотрены лишь собственно методы аутен-

тификации сообщений, причем *без обеспечения функции их конфиденциальности* одновременно с аутентификацией. Точная постановка этой задачи формулируется ниже.

2.6.1. Общая структура аутентификации

Пусть имеется сообщение *M*, представленное в любой форме, например в виде двоичной последовательности конечной длины. *Аутентификатором* E_s к этому сообщению называется некоторая последовательность, которая является функцией от этого сообщения и от секретной последовательности k (ключа), т. е. $E_s = f(M,k)$. Таким образом, *аутентифицированное сообщение* представляет собой пару: сообщение/аутентификатор (M, E_s). Для проверки подлинности сообщения необходимо иметь тот же самый ключ *k* и знать функцию $f(\cdot)$. Тогда по известной паре $\tilde{M}, \tilde{E}_s$ законный пользователь вычисляет аутентификатор $\tilde{\tilde{E}}_s = f\left(\tilde{M}, k\right)$ и сравнивает его с принятым аутентификатором $\tilde{E}_s$. Если $\tilde{\tilde{E}}_s = \tilde{E}_s$, то сообщение признается подлинным, в противном случае – ложным (т. е. не подлинным).

Различают два вида атак на системы аутентификации:

атака имперсонизации, когда злоумышленник пытается подменить сообщение *M* на другое сообщение M', не зная аутентификатора E_s и ключа *k*;

атака подстановки, когда злоумышленник, зная пару (M, E_s), но не зная ключа *k*, по заданному им ложному сообщению M' пытается построить такой ложный аутентификатор E_s', что законный пользователь не сможет обнаружить эту подмену и примет пару (M', E'_s) как подлинную.

2.6.2. Классификация систем аутентификации и характеристики их эффективности

Различают два основных типа систем аутентификации:

- *безусловно стойкие;*
- *вычислительно стойкие.*

Эффективность безусловно стойких систем аутентификации определяется следующими параметрами: вероятностью успешной атаки имперсонизации p_i, вероятностью успешной атаки подстановки p_s, вероятностью необнаруженного навязывания $p_D = \max[p_i, p_s]$, длиной *b* аутентификатора E_s и объемом (длиной в битах *N*) требуемого для аутентификации ключа *k*.

Эффективность вычислительно стойких систем аутентификации определяется той же совокупностью параметров и, кроме того, минимально возможной сложностью вычислений (числом операций и объемом памяти), при которой данные вероятностные характеристики еще выполняются.

Таким образом, безусловно стойкие системы являются таковыми независимо от вычислительного ресурса злоумышленника, тогда как для вычислительно стойких систем их надежность зависит от этого вычислительного ресурса. Хотя на первый взгляд кажется, что всегда надо отдавать предпочтение безусловно стойким системам, это на самом деле не так. Как будет показано ниже, безусловно стойкие систе-

мы имеют существенный недостаток (большой расход ключа), который резко ограничивает область их применения. (Заметим, что тут имеется полная аналогия с совершенными и вычислительно стойкими шифрами, которые рассматривались ранее.)

2.6.3. Безусловно стойкие системы аутентификации

В [14] приводятся следующие нижние границы для вероятностей ошибок, характеризующих безусловно стойкие системы аутентификации:

$$p_i \geq 2^{-I(E_s;k)}; \tag{2.23}$$

$$p_s \geq 2^{-H(k/E_s)}; \tag{2.24}$$

$$p_D \geq 1/\sqrt{|k|}, \tag{2.25}$$

где $I(X;Y), H(Y/X)$ – количество информации и условная энтропия (по Шеннону), вычисленные для соответствующих случайных переменных; $|k|$ – общее количество ключей, используемых для аутентификации. Очевидно, что если ключ представляет собой двоичную цепочку длины *N*, то граница (2.25) принимает вид

$$p_D \geq 2^{-N/2}. \tag{2.26}$$

Если для некоторой системы аутентификации выполняется равенство в (2.26), то такая система называется *совершенной*. В [14] приводится следующее необходимое условие для получения совершенной системы аутентификации:

$$|M| \leq \sqrt{|k|} + 1. \tag{2.27}$$

Неравенство (2.25) на первый взгляд кажется неверным, поскольку случайность ключа как бы не используется полностью для обеспечения случайности подмены. Однако кажущееся противоречие объясняется тем обстоятельством, что знание злоумышленником аутентификатора, соответствующего известному сообщению, уменьшает неопределенность выбора ключа, при котором эта подмена не будет обнаружена. Например, для системы аутентификации, показанной на рис. 2.21, при получении сообщения *M* и аутентификатора мог быть выбран не любой из 8 ключей, а лишь один из двух – k_7 или k_8. Тогда злоумышленник, который хочет подменить сообщение *M* на M' может с равной вероятностью выбрать аутентификатор $E_s^{'}$ или $E_s^{''}$.

Для построения систем аутентификации с гарантированными величинами p_i, p_s потребуется изложить далее некоторый дополнительный математический аппарат.

Определение 2.4. *Хеш-функцией* $y = h(x)$ называется преобразование, отображающее множество всех двоичных последовательностей *X* длины *n* во множество двоичных последовательностей *Y* длины *b*, где $b < n$. Символически это преобразование можно записать так: $GF(2)^n \to GF(2)^b$.

Хеш-функции бывают *ключевыми* и *бесключевыми* (т. е. зависящими или не зависящими от ключа). Если хеш-функция является ключевой, то можно говорить о классе хеш-функций, где каждая функция из класса соответствует выбору определенного ключа.

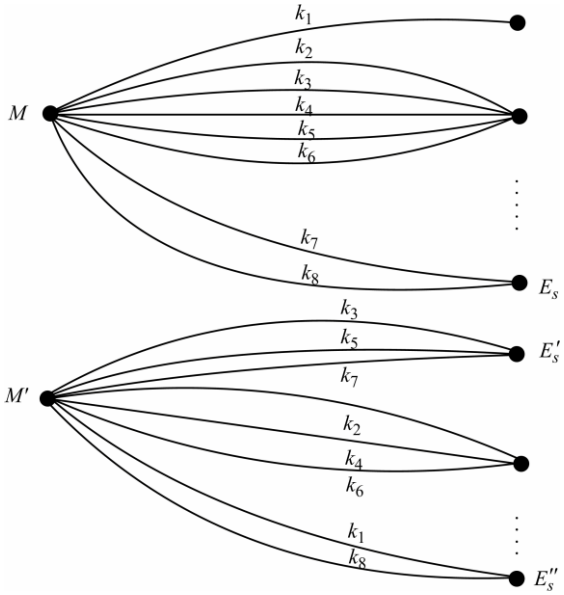


Рис. 2.21. Безусловно стойкая система аутентификации

Определение 2.5. Классом *строго универсальных* хеш-функций называется такое множество *H* отображений , что:

а) для любых $x \in X, y \in Y$

$$\#\{h \in H : y = h(x)\} = \frac{|H|}{|Y|},$$

где $|H|$ – общее количество хеш-функций *h*; $|Y|$ – общее количество аутентификаторов *Y*; $\#\{\cdot\}$ – количество хеш-функций, удовлетворяющих условию, представленному в фигурных скобках;

б) для любых $x_1, x_2 \in X$, $x_1 \neq x_2$ и $y_1, y_2 \in Y$

$$\#\{h \in H : h(x_1) = y_1, h(x_2) = y_2\} = \frac{|H|}{|Y|^2}.$$

Пример строго универсального класса хеш-функций. Пусть $x \in GF(2)^n$, $y = h(x) = (x \times h_0 + h_1)_b$, где «×», «+» означают соответственно умножение и сложение в конечном поле $GF(2^n)$; h_1, h_0 – двоичный ключ общей длины $2n$; $(\cdot)_b$ означает «усечение», т. е. выбор *b* левых (или правых) бит последовательности, стоящей в скобках. Легко проверить, что такой класс хеш-функций является строго универсальным.

Рассмотрим метод построения безусловно стойкой системы аутентификации на основе использования хеш-функций, выбранных из строго универсального класса, приведенного выше. В такой системе длина сообщения должна быть равна *n*, длина ключа $2n$, а длина аутентификатора $b \leq n$.

Утверждение 2.2. Если для аутентификации используются функции, выбираемые равновероятно из строго универсального класса, то вероятность успешного выполнения оптимальной атаки подстановки будет равна $p_s = \frac{1}{2^b}$, где b – длина аутентификатора.

Доказательство. Пусть имеется некоторое сообщение M и законный пользователь выбрал некоторый ключ k_0, который определяет единственную хеш-функцию из строго универсального класса. По ключу k_0 этот пользователь вырабатывает аутентификатор $E_s = h(M, k_0)$. Обозначим через H' множество всех ключей из H, которые переводят заданное M в такой аутентификатор E_s (рис. 2.22).

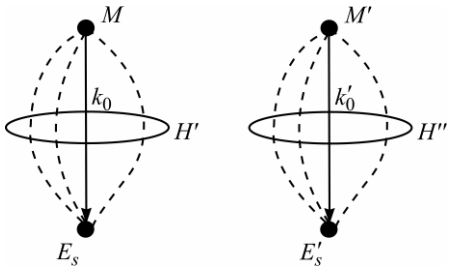


Рис. 2.22. Атака подстановки на систему аутентификации, использующую хеш-функции из универсального класса

Пусть злоумышленник хочет сформировать ложное сообщение M', которое законный пользователь аутентифицировал бы как правильное. Тогда он должен создать ложный аутентификатор $E'_s = h(M', k_0)$. Однако он не знает ключа k_0, которым воспользовался бы для проверки законный пользователь. Вся информация, которую имеет злоумышленник при известных ему M и E_s (даже при его неограниченном вычислительном ресурсе), – это множество ключей H'. Таким образом, *оптимальная стратегия* злоумышленника состоит в том, чтобы случайно (наугад) выбрать ключ k'_0 из множества H'. Тем не менее ему не обязательно в точности угадать истинный ключ k_0, а достаточно лишь угадать один из ключей, принадлежащих H', которые дают отображение: $M' \to E'_s$. Обозначим множество таких ключей через H'' (рис. 2.22). Согласно свойству а) строго универсального класса, количество ключей из равно в точности $|H'| = \frac{|H|}{|E_s|}$, а по свойству б) количество ключей из H'' будет в точности равно $|H''| = \frac{|H|}{|E_s|^2}$. Тогда вероятность того, что злоумышленник, зная множество H', угадает один из элементов множества H'', т. е. что навязывание ложного сообщения M' окажется успешным, может быть записана так:

$$p_s = \frac{|H''|}{|H'|} = \frac{1}{|E_s|} = \frac{1}{2^b}, \qquad (2.28)$$

что и требовалось доказать.

Сравним полученный результат с нижней границей для вероятности необнаруженной подстановки (2.25):

$$p_s \geq \frac{1}{\sqrt{|k|}} = \frac{1}{\sqrt{2^{2n}}} = \frac{1}{2^n}.$$

Так как в схеме $p_s = \frac{1}{2^b}$, эта граница достигается при $b = n$ и для этого частного случая схема аутентификации на основе использования для аутентификации универсальных хеш-функций оказывается совершенной.

Приведенная выше схема аутентификации имеет, однако, два существенных недостатка:

длина ключа достаточно велика (она всегда в 2 раза превышает длину сообщения!);

схема совершенна лишь при длине аутентификатора, равной длине сообщения.

Эти особенности неприемлемы при многих практических применениях. Кроме того, общим недостатком всех безусловно стойких схем аутентификации является то, что они могут быть использованы лишь *однократно*. Это означает, что если необходимо аутентифицировать на одном и том же ключе несколько сеансов при передаче различных сообщений, то это приведет к тому, что вероятность $p_s \to 1$ при увеличении числа сеансов. Для исключения последнего недостатка существует модификация, предназначенная для обеспечения безопасной работы многократных систем аутентификации. Пусть уже имеется система однократной аутентификации, основанная на использовании строго универсальных хеш-функций, которая определяется следующим уравнением выработки аутентификатора $E_s = h(k, M_1)$, где k – секретный ключ, M_1 – аутентифицируемое сообщение. Тогда аутентификатор для L-кратной системы предлагается формировать следующим образом: $E_s = (E_{s_1}, E_{s_2}, ..., E_{s_L})$, где

$$E_{s_i} = h\left(k, M_i\right) \oplus k_i \qquad (2.29)$$

при $i = 1, 2, 3, \ldots, L$; k_i – дополнительный b-битовый ключ для i-го сеанса аутентификации.

В [15] доказывается, что если при использовании такой системы злоумышленник наблюдает L сообщений $M_1, M_2, ..., M_L$ и L соответствующих им аутентификаторов $E_{s_1}, ..., E_L$ и знает, что последние сформированы по правилу (2.29) (но, конечно, не знает секретных ключей $k, k_1, ..., k_L$), то вероятность необнаруженной подстановки любого ложного сообщения $M' \neq M_i, i = 1, 2, ..., L$, не может быть больше, чем $P_s = \frac{1}{2^b}$, где b – как и ранее, число бит в аутентификаторе. Таким образом, в данной системе многократной аутентификации обеспечивается та же надежность, что и ранее была в однократной системе, но длина требуемого для этого ключа возрастает до $2n + Lb$ бит, что может оказаться, однако, значительно меньше, чем значение $2nL$, которое потребовалось бы при смене ключа k на каждом сеансе.

Для того чтобы избавиться от отмеченных ранее недостатков приведенной выше схемы однократной аутентификации (большая длина ключа и большая длина аутентификатора), рассмотрим другую схему аутентификации. Пусть сообщение M представлено в виде

$$M = (M_1, M_2, \ldots, M_t),$$

где $M_i \in GF\left(2^n\right)$.

Тогда сформируем аутентификатор следующим образом:

$$E_s = \sum_{i=1}^{t} M_i k_1^i + k_2,$$

где $k_1, k_2 \in GF\left(2^n\right)$ – ключи, а все действия выполняются в поле $GF(2^n)$.

Для такой схемы получаем число всех возможных сообщений $|M| = 2^{nt}$, число всех возможных аутентификаторов $|E_s| = 2^n$, полное число ключей $|k| = 2^{2n}$. Легко рассчитать *скорость аутентификации* (как отношение числа бит сообщения к сумме числа бит сообщения и аутентификатора): $R = \dfrac{t}{t+1}$. (Видно, что в этом случае в качестве аутентификатора просто выбирается один элемент конечного поля $GF(2^n)$.) Скорость расхода ключа в такой системе (т. е. отношение длины ключа к длине сообщения) будет, очевидно, следующая: $R_k = \dfrac{2n}{tn} = \dfrac{2}{t}$. Можно доказать, что для такой схемы вероятность необнаруженной подстановки будет иметь следующий вид [16]:

$$P_s = \frac{t}{2^n}. \qquad (2.30)$$

Таким образом, для рассмотренной выше системы аутентификации при использовании длинных сообщений скорость аутентификации оказывается близкой к единице, скорость расхода ключа – близкой к нулю (что приближает ее по эффективности к системе без аутентификации), тогда как вероятность появления необнаруживаемого навязывания может быть сделана достаточно малой, и она лишь в t раз больше, чем в совершенных системах аутентификации.

Можно добиться и более свободного варьирования параметрами системы аутентификации, если перейти к выбору хеш-функций из так называемого *почти строго универсального* класса или использовать технику *интерактивной аутентификации*, но рассмотрение этих методов выходит за рамки основного курса. Тем не менее какие бы методы безусловно стойкой аутентификации ни использовались, все они имеют уже отмечавшийся ранее существенный недостаток. Если мы хотим отдельно аутентифицировать сообщения, передававшиеся на разных сеансах связи (или, скажем, содержащиеся в разных файлах), то это неизбежно приведет к *росту длины требуемого секретного ключа*, пропорциональному числу сеансов (файлов). Для того чтобы избавиться от этого недостатка и получить возможность многократно аутентифицировать различные сообщения при помощи одного и того же доста-

точно короткого ключа, необходимо переходить к использованию вычислительно стойких систем аутентификации.

2.6.4. Вычислительно стойкие системы аутентификации

Система аутентификации называется *вычислительно стойкой*, если возможность необнаружения подделки сообщения определяется вычислительной мощностью нарушителя.

Основные определения и классификация

Различают два класса вычислительно стойких систем аутентификации:

- с использованием *симметричных* методов шифрования;
- с использованием *несимметричных* методов шифрования.

В первой части книги «Основы криптографии» рассматриваем только системы аутентификации с использованием симметричных методов шифрования. Системы аутентификации с использованием несимметричных методов шифрования будут изучаться во второй части этой книги («Основы криптографии с открытым ключом»).

Вычислительно стойкие системы аутентификации, как и безусловно стойкие, используют в основном технику хеш-функций, сконструированных, однако, по другому принципу. Эти хеш-функции делятся также на два основных класса:

- *ключевые хеш-функции* (зависящие от секретного ключа, разделяемого между законными пользователями);
- *бесключевые хеш-функции*, вычисляемые по полностью открытому алгоритму.

В последнем случае, для того чтобы обеспечить преимущество законным пользователям перед незаконными, необходимо после выполнения бесключевого хеширования выполнить дополнительную процедуру с использованием ключей. При этом роль хеш-функций состоит в существенном уменьшении длины аутентифицируемого сообщения, что позволяет упростить процедуру собственно аутентификации, которая обычно выполняется с использованием несимметричной техники шифрования, называемой *цифровой подписью*. Поскольку вопросы несимметричного шифрования будут изучаться во второй части книги, там же будут рассмотрены и бесключевые хеш-функции, а здесь остановимся только на ключевых хеш-функциях.

Заметим, что в иностранной технической литературе принято называть ключевые хеш-функции, предназначенные для криптографического применения, *кодом аутентификации сообщений* (кратко – *MAC-кодом*), тогда как бесключевые хеш-функции называют там *кодом, обнаруживающим модификации* (кратко – *MDC-кодом*).

Основные свойства и способы построения ключевых хеш-функций

Не умаляя общности, будем предполагать далее, что входом для MAC-кода являются двоичные последовательности x длины m, а выходом – двоичные последовательности y длины l, и обозначать это преобразование через $h(k,\ x)$, где k – секретный ключ аутентификации. Тогда аутентифицированное сообщение M будет представлять собой пару $(x,\ y)$, где $x = M,\ y = h(k, x)$. Для проверки подлинности аутентифицированного сообщения $(\tilde{x}, \tilde{y})$ законный пользователь вычисляет $\tilde{y}h(k,\ \tilde{x})$ и

сравнивает его с $\tilde{y}$. При совпадении этих последовательностей сообщение полагается подлинным, а при несовпадении – искаженным.

Определение 2.6. *Ключевой криптографической хеш-функцией* (MAC) называется функция $h(k,\ x)$, обладающая следующими свойствами:

1) хеш-функция легко вычисляется для известных $x,\ k$;

2) длина выходной последовательности l может быть выбрана, вообще говоря, произвольной, но в типичных случаях $l << m$, где m – длина входной последовательности;

3) если задано произвольное число аутентифицированных пар $(x_i, h(k, x_i)),\ i = 1,\ 2,\ \ldots$, то вычислительно невозможно найти ни одной новой аутентифицируемой пары $(x,h(k,x))$ для любого нового входа $x \neq x_i$ при неизвестном ключе k (включая и случай, когда $h(k,x) = h(k,x_i)$ для некоторого i). Третье свойство означает, что MAC является *вычислительно стойким* методом аутентификации, поскольку злоумышленник, имея в своем распоряжении многократно аутентифицированные сообщения, не может без знания ключа создать такое новое сообщение и аутентификатор к нему, чтобы оно было воспринято как подлинное законным пользователем, обладающим секретным ключом k.

Заметим, что третье свойство предполагает вычислительную невозможность нахождения ключа k по известным парам $x_i, h(k,x_i)$, но выполнение только этого условия не влечет немедленно вычислительную стойкость системы аутентификации, поскольку нахождение секретного ключа не всегда необходимо для успешной подделки сообщения.

Построение MAC на основе CBC-моды блоковых шифров

Наиболее распространенным способом построения MAC является использование CBC-мод различных блоковых шифров. Пусть имеется некоторый блоковый шифр с длиной блока n и с алгоритмом шифрования $E = f_k(M)$. Предположим, что сообщение M имеет произвольную длину m (не обязательно кратную n). Тогда первоначально производится процедура *добавления* t дополнительных бит (обычно известной последовательности), что обеспечивает кратность $m + t$ длине блока шифра n. После этого формирование аутентификатора y длины $l = n$ (выхода хеш-функции) выполняется по следующему алгоритму:

$$y \quad = f_k(M_i \oplus y_{i-1}), i = 1,\ 2,\ \ldots,\ t, y_0 = 0, t = (m+t)/n, \qquad (2.31)$$

где M_i – i-й подблок сообщения длины n, т. е.

$$M = \left(M_1,\ M_2,\ \ldots,\ M_i,\ \ldots,\ M_{(m+t)/n}\right).$$

Очевидно, что длина MAC оказывается равной длине блока выбранного шифра n, сложность выполнения аутентификации определяется сложностью шифрования, а вычислительная стойкость аутентификации – стойкостью этого шифра.

Если полный перебор всех ключей оказывается возможным, то знание одной или нескольких пар аутентифицированных сообщений позволит осуществить необнаруженное навязывание любого сообщения. В случае, когда (как в шифре DES) число ключей оказывается переборным, обычно используется *усиленный* CBC-MAC, в котором (подобно тройному DES с двойным ключом) аутентификатор y' формируется следующим образом: $y' = f_k(g_{k'}(y))$, где $g_{k'}(\cdot)$ – функция дешифрования вы-

бранного блокового шифра для второго ключа $k' \neq k$, y – аутентификатор, построенный по (2.34). (Очевидно, что такая модификация требует удвоения длины ключа аутентификации.)

Если длина ключа в выбранном блоковом шифре не допускает его полного перебора, а длина аутентификатора меньше длины ключа, то атака имперсонизации может состоять в попытке случайного угадывания правильного аутентификатора для ложного сообщения, при которой, очевидно, вероятость такого угадывания будет равна 2^{-l}, где l – длина аутентификатора. Атака подстановки на такую систему аутентификации затруднена из-за вычислительной сложности нахождения даже единственно возможного ключа для заданного сообщения и правильного аутентификатора. Здесь уместно еще раз подчеркнуть принципиальную разницу между вычислительно стойкой и безусловно стойкой системами аутентификации, поскольку для последней не представляет труда вычислить хотя бы один ключ, который при известном сообщении M, дает известный аутентификатор E_s. Действительно, если хеширование в строго универсальном классе было задано соотношением $y = (xh_0 + h_1)_b$, где $x = M, y = E_s$, то, дополняя двоичную последовательность y длины b произвольной двоичной последовательностью $\tilde{y}$ длины $n - b$ так, что длина их конкатенации (объединения) $\tilde{y}(y, \tilde{y})$ оказывается равной n, можно для произвольного ключа $\tilde{h}_1$ найти соответствующий ему ключ $\tilde{h}_0 = (\tilde{y}_1)x^{-1}$ для любого $x \neq 0$, причем сложность этих вычислений в конечном поле $GF(2^n)$ оказывается полиномиальной по n.

Следовательно, можно достаточно просто вычислить хотя бы один ключ $(\tilde{h}_0,\ \tilde{h}_1)$, который формирует заданный аутентификатор E_s для заданного сообщения M. Однако, как уже отмечалось ранее, проблема состоит в том, какой ключ в действительности был использован среди огромного множества допустимых ключей.

Заметим, что существуют модификации MAC, где длина аутентификатора отличается от длины блокового шифра [3].

Построение MAC на основе MDC

Известны различные методы посторения MAC на основе MDC [3]. Один из рассмотренных в [2] методов состоит в том, что сначала при помощи MDC находится промежуточный аутентификатор, не зависящий от ключа, а затем уже над ним выполняется некоторое преобразование (подобное шифрованию), зависящее от секретного ключа. Чаще всего, в качестве такого преобразования используется несимметричное шифрование.

Другими вариантами построения MAC на основе MDC является использование ключа как части сообщения, поступающего на вход MDC.

Построение MAC
с использованием специально разработанных алгоритмов

В этом случае сообщение обычно разбивается на блоки определенной длины, и далее эти блоки подвергаются определенным преобразованиям, использующим, например выполнение операций сложения по модулю определенного числа, и управляемым секретным ключом [3].

Пример практически используемой вычислительно стойкой системы аутентификации (режим выработки имитовставки для ГОСТ 28147)

В этом режиме используется модификация блокового шифра CBC, рассмотренная выше. Сообщение *M* разбивается на подблоки длиной 64 бита (поскольку именно такую длину имеет шифр, используемый в ГОСТ 28147. Далее последний подблок дополняется (если в этом есть необходимость) до длины 64 бита, и затем выполняется операция (2.34), где в качестве преобразования $f_k(\cdot)$ используется шифрование по ГОСТ 28147 при выборе ключа длиной 256 бит. Результатом выполнения полного цикла преобразования будет, очевидно, двоичный блок длиной 64 бита, который может быть сокращен до 32 бит при помощи выбора 32 символов, взятых в начале полученного ранее 64-битового блока. Последовательность из 32 бит и называется *имитовставкой*, которая присоединяется к полному сообщению. Для верификации принятого сообщения по известному ключу и принятому сообщению вычисляется имитовставка по правилу (2.31), которая сравнивается с принятой имитовставкой, и при их совпадении сообщение полагается подлинным, а в противном случае – отвергается.

Как отмечалось ранее, стойкость такой аутентификации определяется стойкостью используемого шифра, и поскольку шифр ГОСТ 28147 считается стойким и, конечно, не допускает перебора всех его 2^{256} ключей в обозримое время, то и основанную на этом шифре аутентификацию можно полагать стойкой. Для дальнейшего ее улучшения можно увеличить длину имитовставки до 64 бит.

3. Криптографические системы с открытым ключом

3.1. Принципы построения криптографических систем с открытым ключом (ОК)

Основная идея состоит в создании таких криптосистем, когда алгоритм шифрования и дешифрования, а также ключ шифрования являются полностью открытыми, но несмотря на это найти ключ дешифрования или сообщение не представляется возможным в обозримое время. Такое преобразование можно назвать *односторонним с подсказкой*, поскольку прямое преобразование (шифрование) выполняется просто, тогда как обратное к нему преобразование (дешифрование) без дополнительной подсказки (ключа дешифрования) выполняется весьма сложно.

Определение 3.1. *Асимметричной* криптосистемой (*криптосистемой с ОК*) называется такая криптосистема, в которой ключ дешифрования не равен ключу шифрования, причем невозможно найти ключ дешифрования по известному ключу шифрования.

Рассмотрим математическую модель шифрования/дешифрования, которая уже была приведена во второй части пособия, но с использованием других (более подходящих для нашего случая) обозначений: $C = f_e(M)$, где e – ключ шифрования; $M = g_d(C)$, где d – ключ дешифрования; M – сообщение; C – криптограмма; f, g – функции шифрования и дешифрования соответственно.

Для криптосистем с открытым ключом (КС ОК) также предполагается выполнение принципа *Керхгоффа* (т. е. посторонним должно быть известно все, кроме ключа дешифрования d).

Сформулируем более точно основные требования, которые предъявляются к криптосистемам с ОК:

1) нахождение пары ключей e и d должно быть вычислительно простым;

2) при известном ключе шифрования e нахождение $C = f_e(M)$ должно быть вычислительно простым;

3) при известном ключе дешифрования d восстановление сообщения $M = g_d(C)$ должно быть вычислительно простым;

4) при известном ключе шифрования e нахождение d должно быть вычислительно сложным (необозримо большим);

5) при известном ключе шифрования e, но неизвестном ключе дешифрования d нахождение сообщения M по известной криптограмме C должно быть вычислительно сложным.

Если условие 4 выполняется, то очевидно, что ключ шифрования можно сделать *открытым (общедоступным*). Отсюда происходит и название данного типа криптосистем.

Для того чтобы показать, что КС ОК имеет практический смысл, нужно ответить на два вопроса:

а) Можно ли выполнить пять вышеперечисленных требований?

б) Какие преимущества имеет КС ОК в сравнении с традиционными (симметричными) криптосистемами?

1-е преимущество КС ОК

Основным преимуществом КС ОК является упрощение процедуры распределения ключей.

Пусть в сети имеется N пользователей и каждый хочет конфиденциально связаться с каждым. Тогда в такой сети нужно иметь $C_N^2 = \frac{N(N-1)}{2}$ ключей, причем ключ для каждой пары должен распределяться секретным образом (рис. 3.1, а), и только после этого открыто передается зашифрованное сообщение.

В КС ОК ключ шифрования e передается открыто (или хранится в каком-либо центре распределения ключей (рис. 3.1, б), а хранить в секрете необходимо только свой ключ дешифрования d. Ключи других пользователей можно просто при необходимости запросить (рис. 3.1, б).

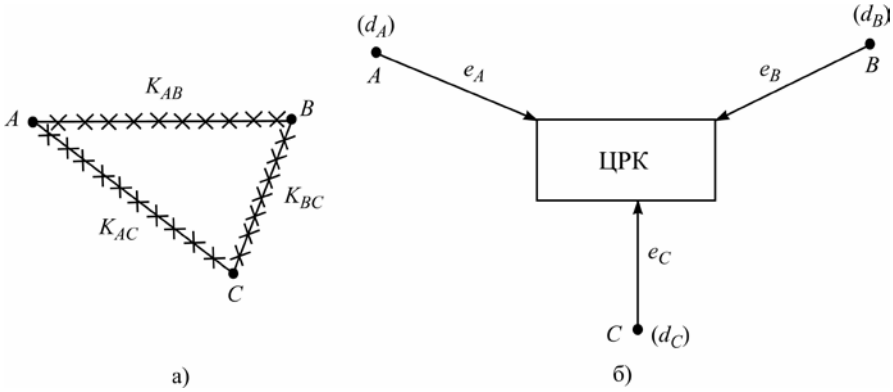


Рис. 3.1. Распределение ключей в различных КС:
а) симметричная КС; б) КС ОК

Однако задача распределения ключей полностью не решается с использованием КС ОК, поскольку здесь возникает проблема *аутентификации* (обеспечения подлинности) открытых ключей. Если эта проблема не решена, то злоумышленник E может выдать себя за легального пользователя A, посылая e_E к A и выдавая себя, скажем, за B.

2-е преимущество КС ОК

Вторым преимуществом КС ОК перед обычными КС (криптосистемами) является возможность выполнения их пользователями других криптографических функций

(цифровая подпись, некоторые протоколы и т. п.), в которых пара взаимодействующих пользователей не обязательно является дружественной.

3-е преимущество КС ОК

Третьим преимуществом КС ОК перед обычными КС является свойство «прозрачности» обеспечения стойкости этих криптосистем. Такие системы называются *доказуемо секретными*. Это означает, что стойкость криптосистемы будет настолько сложна, насколько сложно решение *строго определенной математической задачи*. В этом плане КС ОК позволяют доказать их стойкость проще, чем в традиционных симметричных КС.

Для того чтобы положительно ответить на первый вопрос, необходимо привести хотя бы один пример, удовлетворяющий всем пяти требованиям.

Предварительно рассмотрим один «механический» иллюстрирующий пример, смысл которого ясен из рис. 3.2.

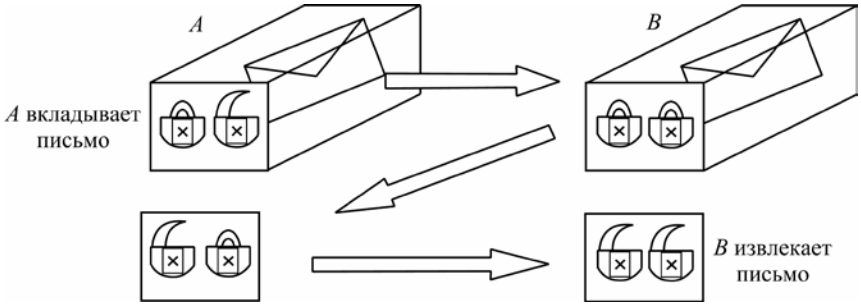


Рис. 3.2. Отсылка письма по почте в сейфе при наличии у отправителя и получателя разных ключей к сейфу

Данная идея может быть формализована в виде некоторого алгоритма, использующего обычную симметричную КС, но удовлетворяющую условию *перестановочности* операций шифрования с разными ключами, т. е. когда

$$f_{k_A}\left(f_{k_B}(M)\right) = f_{k_B}\left(f_{k_A}(M)\right),$$

где $f_x(y)$ – функция шифрования «*y*» при ключе «*x*».

Тогда шифрование без предварительного распределения ключей выполняется следующим образом:

A		B
(k_A)		(k_B)
1.	$\xrightarrow{C_A = f_{k_B}(M)}$	
2.	$\xleftarrow{C_B = f_{k_B}\left(f_{k_A}(M)\right)}$	
3.	$\xrightarrow{C'_A = f^{-1}_{k_A}(C_B)}$	$f^{-1}_{k_B}(C'_A) = M$

где $f_x^{-1}(y)$ – это функция дешифрования *y* на ключе *x*.

Однако здесь также остается открытой проблема аутентификации.

В данном разделе пособия предполагается представить криптосистемы, удовлетворяющие пяти вышеперечисленным требованиям. Характерной особенностью КС ОК является то обстоятельство, что алгоритмы шифрования и дешифрования в них обычно выполняются как действия по модулю некоторого целого числа или как операции в конечных полях. Этим КС ОК отличаются от традиционных криптосистем, для которых более типичными являются обычные булевы операции над двоичными переменными.

Для того чтобы понять работу КС ОК и доказать их стойкость, необходимо подготовить некоторый математический базис. Этот базис состоит из основ теории чисел и теории квадратичных вычетов.

3.2. Построение практических криптосистем с открытым ключом

3.2.1. Криптосистема RSA (Райвеста–Шамира–Адлемана)

Метод формирования пар открытых/закрытых ключей для КС RSA

Каждый пользователь КС RSA, допустим A, выполняет следующие операции для формирования пары ключей:

1) генерирует пару простых чисел p и q;

2) вычисляет $n = p \cdot q$ и функцию Эйлера $\varphi(n) = (p-1) \cdot (q-1)$;

3) генерирует e, где $1 \leq e \leq \varphi$, такое что $\gcd(e, \varphi) = 1$;

4) находит число $d = e^{-1} \bmod \varphi$, т. е. решение уравнения $e \cdot d = 1 \bmod \varphi$;

5) выбирает числа e, n как свой открытый ключ, а d – как свой секретный ключ.

Как было показано ранее, описанные выше операции могут быть выполнены достаточно быстро даже для чисел n, имеющих 100 и более десятичных разрядов.

Действительно, операции выполняются следующим образом:

1) случайным генерированием чисел и тестированием Миллера–Рабина;

2) обычным умножением, что требует $O(\log p^2)$ битовых операций;

3) использованием алгоритма Евклида;

4) при помощи расширенного алгоритма Евклида для нахождения обратного элемента.

Вся процедура генерирования пар ключей выполняется один раз на длительное время действия этих ключей.

Шифрование в КС RSA

Предположим, что пользователь B хочет передать пользователю A сообщение M в зашифрованном виде. Для этого он выполняет следующие шаги:

- получает подлинный открытый ключ пользователя A (e_A, n_A);

- преобразует сообщение M в последовательность целых чисел M_1, M_2, …, M_i, …, не превосходящих $(n_A - 1)$;
- для каждого числа M_i, соответствующего части сообщения, формирует криптограмму c_i по правилу $c_i = M_i^{e_A} \bmod n_A$ и отправляет результат пользователю A.

Видно, что процедура шифрования может быть выполнена достаточно быстро при использовании алгоритма быстрого возведения в степень по модулю.

Дешифрование в КС RSA

Пользователь A для дешифрования предназначенной ему криптограммы c_i выполняет следующие операции с использованием своего секретного ключа d_A:

а) дешифрует $M_i = {c_i}^{d_A} \bmod n_A$, $i = 1,\ 2,\ \ldots$;

б) преобразует число M_i в содержательный вид (форму представления информации).

Видно, что операция дешифрования выполняется достаточно быстро. Докажем, что представленная выше процедура дешифрования действительно позволяет получить истинное сообщение $M = M_i$, которое и было ранее зашифровано.

Доказательство. Поскольку по условию $e \cdot d = 1 \bmod \varphi$, то существует такое число «*k*», что $e \cdot d = 1 + k \cdot \varphi$. Предположим сначала, что $\gcd(M, p) = 1$, тогда по теореме Ферма имеем

$$M^{p-1} = 1 \bmod p. \tag{3.1}$$

Возведем обе части (3.1) в степень $k(q-1)$, а затем умножим обе части на M:

$$M^{1+k(p-1)(q-1)} = M \bmod p. \tag{3.2}$$

Рассмотрим показатель степени в левой части равенства (3.2):

$$1 + k(p-1)\,(q-1) = 1 + k \cdot \varphi = e \cdot d.$$

Подставляя это выражение в (3.2), получим $M^{1+k(p-1)(q-1)} = M^{e \cdot d}$. Из равенства (3.2) получаем, что $M^{e \cdot d} = M \bmod p$, но, с другой стороны, $M^e = c$, следовательно, $c^d = M \bmod p$, что и требовалось доказать.

Если теперь отказаться от заданного ранее условия $\gcd(M, p) = 1$, то тогда обязательно должно выполняться условие $\gcd(M, p) = p$ и соотношение (3.2) будет тривиально выполняться, так как обе его части равны 0 по модулю p.

В итоге получаем, что $c^d = M \bmod p$ во всех случаях. Следуя аналогичной технике, можно доказать, что $c^d = M \bmod q$, а поскольку p и q различные простые числа, то отсюда следует, что

$$c^d = M \bmod (p \cdot q) \Rightarrow c^d = M \bmod n.$$

Последнее равенство доказывает, что используемая в RSA процедура дешифрования действительно восстанавливает исходное сообщение.

Стойкость КС RSA

Итак, для КС RSA имеем следующие соотношения, связывающие ключи, сообщение и криптограмму:

$$\begin{aligned} &c = M^e \bmod n; \\ &M = c^d \bmod n; \\ &n = p \cdot q; \\ &e \cdot d = 1 \bmod \varphi; \\ &\varphi = (p-1)\cdot(q-1). \end{aligned} \qquad (3.3)$$

Из уравнений (3.3) видно, что система RSA может быть вскрыта, если удастся найти p и q, т. е. факторизовать n.

Исходя из этого факта p и q должны выбираться такой большой разрядности, чтобы факторизация числа n потребовала необозримо большого времени, даже с использованием всех доступных и современных средств вычислительной техники. В настоящее время задача факторизации чисел не имеет полиномиального решения. Разработаны лишь некоторые алгоритмы, упрощающие факторизацию, но их выполнение для факторизуемых чисел большой разрядности все равно требует необозримо большого времени. Действительно, сложность решения задачи факторизации для наилучшего известного сейчас алгоритма факторизации равна $O\left(e^{\sqrt{\ln(n)\cdot\ln\ln(n)}}\right)$. Так, например, если $\ln n = 200$, то число операций будет приблизительно равно 10^{28}.

При увеличении числа разрядов n до 1000 и более время факторизации становится совершенно необозримым.

Если теперь предположить, что алгоритм факторизации неизвестен, но удалось как-то в полиномиальное время найти секретный ключ d, то можно доказать, что число n тогда можно факторизовать, т. е. найти такие p и q за полиномиальное время, что $n = p \cdot q$. Если каким-то другим способом удалось в полиномиальное время найти параметр φ, то ясно, что КС RSA может быть вскрыта.

Однако, как было показано ранее, знание φ позволяет просто факторизовать n, и, следовательно, задача нахождения φ вычислительно эквивалентна задаче факторизации. Тем не менее для систем RSA нельзя строго утверждать, что стойкость этой КС эквивалентна задаче факторизации [21]. Подобное свойство имеет место для иных КС, например для КС Рабина, которая будет рассматриваться далее.

Другой естественной атакой на КС RSA является дискретное логарифмирование. Эта атака (при известном сообщении) выполняется следующим образом: $d = \log_c M \bmod n$. Однако задача дискретного логарифмирования по модулю многоразрядных чисел также относится к трудным в математике, и оказывается, что она имеет почти такую же сложность, как и задача факторизации [21].

Побочные атаки на КС RSA

Помимо двух основных атак на КС RSA – *факторизации* и *логарифмирования* – существует ряд побочных атак, основанных на предположении об использовании в этой КС таких параметров или режимов, которые значительно упрощают ее криптоанализ.

Рассмотрим далее некоторые из этих атак, связанных с неправильным выбором параметров КС RSA при ее разработке или эксплуатации.

Первая побочная атака.
Выбор малой величины открытого ключа e

Смысл такого выбора e для КС RSA состоит в том, что это позволяет ускорить процедуру шифрования. Рассмотрим в качестве примера величину $e = 3$. Предположим, что с таким малым e производится шифрование в КС RSA и одно и то же открытое сообщение посылается хотя бы трем разным пользователям, имеющим разные n. Покажем, что в этом случае можно восстановить сообщение M без знания ключа. Последовательность криптограмм будет выглядеть следующим образом:

$$\begin{aligned} C_1 &= M^3 \bmod n_1; \\ C_2 &= M^3 \bmod n_2; \\ C_3 &= M^3 \bmod n_3. \end{aligned} \qquad (3.4)$$

Весьма вероятно, что будет выполнено условие $\gcd(n_i, n_j) = 1$ при $i \neq j$. Вспомним китайскую теорему об остатках, которая гласит, что система уравнений (3.4) имеет общее решение: $x = M^3 \bmod (n_1 \cdot n_2 \cdot n_3)$, которое может быть найдено в полиномиальное время.

Вероятно также, что сообщение $M < n_1 \cdot n_2 \cdot n_3$, и тогда дешифрование выполняется тривиальным образом как извлечение кубичного корня из числа: $M = \sqrt[3]{x}$. Для защиты от этой атаки необходимо либо не использовать малые e, либо не отправлять одни и те же сообщения разным пользователям. Если в этом есть объективная необходимость, то сообщение нужно немного изменить, добавляя, например, в конце его небольшие различные случайные числа. Такой метод называется методом «*подсаливания*» сообщения.

Вторая побочная атака.
Атака при малом объеме возможных сообщений

Предположим, что число сообщений ограничено значениями $M_1, M_2, \ldots, M_r$, где r обозримо. (Это могут быть, например, различные команды – вперед, назад, влево, вправо и т. п.). Тогда сообщение может быть легко расшифровано.

Действительно, пусть криптограмма C перехвачена. Тогда необходимо попытаться зашифровать все команды известным открытым ключом и определить ту криптограмму, которая совпадает с принятой C:

$$\left.\begin{array}{l} C_1 = M_1^e \bmod n; \\ C_2 = M_2^e \bmod n; \\ \vdots \\ C_r = M_r^e \bmod n \end{array}\right\} \stackrel{?}{=} C.$$

Способ борьбы с такой атакой – это «*подсаливание*» сообщений (т. е. присоединение к ним небольших цепочек бит, полученных с использованием чисто случайного датчика).

Третья побочная атака.
Малая экспонента дешифрования (d)

При выборе малого d существенно упрощается алгоритм дешифрования («малая» в данном случае означает, что $d \approx \sqrt{n}$). Полное описание этой атаки довольно сложно (атака Винера – Wiener's attack [21]). Основная идея состоит в том, что для малых d выводится равенство:

$$\frac{k}{d \cdot g} = \frac{l}{n} + \frac{k}{d \cdot g} \cdot \left(\frac{1}{p} + \frac{1}{q} - \frac{1}{n}\right) - \frac{1}{d \cdot n},$$

где d – секретный ключ; e – ключ шифрования; n – модуль; p, q – простые числа, где $n = p \cdot q$; k, g – простые, малые, целые числа (меньше 100).

Тогда можно доказать, что $\frac{k}{d \cdot g}$ оказывается *усечением конечной непрерывной дроби*, представляющей рациональную часть $\frac{e}{n}$. Поскольку конечная непрерывная дробь вычисляется достаточно легко, можно найти секретный ключ d.

Четвертая побочная атака.
Атака с использованием мультипликативного свойства шифра RSA

Из описания КС RSA следует, что для любых сообщений M_1, M_2

$$(M_1 \cdot M_2)^e = M_1^e \cdot M_2^e = C_1 \cdot C_2 \bmod n,$$

где $C_1 = M_1^e \bmod n$, $C_2 = M_2^e \bmod n$. Это свойство может использовать злоумышленник. Предположим, что злоумышленник E хочет дешифровать сообщение C, предназначенное для пользователя A, и предположим, что A согласен дешифровать любую другую криптограмму для E, кроме C. Тогда E может дешифровать C, действуя следующим образом:

1) E выбирает $x \in Z_n$ и вычисляет $\tilde{C} = C \cdot x^{e_A} \bmod n$. Затем E просит A дешифровать $\tilde{C}$;

2) A выполняет эту просьбу – дешифрует: $\widetilde{M} = \tilde{C}^d \bmod n$, затем передает результат злоумышленнику E. Поскольку

$$\widetilde{M} = \tilde{C}^{d_A} = C^{d_A} \underbrace{\left(x^{e_A}\right)^{d_A}}_{x} \bmod n = Mx \bmod n,$$

то можно выполнить следующий шаг;

3) E, зная $\widetilde{M}$, определяет $M = \widetilde{M} \cdot x^{-1} \bmod n$, т. е. дешифрует криптограмму C.

Чтобы защититься от такой атаки, нельзя дешифровать чужие сообщения. Если все же от этого нельзя отказаться, то после дешифрования надо проверить, что получилось – случайный или осмысленный текст. Если получился случайный текст, то не передавать по запросу дешифрованное сообщение *E*.

Пятая побочная атака.
Атака на систему RSA, использующую общие модули для нескольких пользователей

Это типичная ситуация при генерировании пар ключей для пользователей некоторым общим для них «центром распределения ключей» (рис. 3.3).

Рис. 3.3. Атака на КС RSA, использующую общие модули

В этом случае любой пользователь i, имеющий ключи (e_i, d_i), способен определить любую другую пару ключей. Действительно, знание своего секретного ключа d_i позволяет ему факторизовать n, т. е. определить q и p). Зная e_j (как открытый ключ другого, j-го пользователя), а также используя тот факт, что $e_j \cdot d_j = 1 \bmod (p-1)(q-1)$, i-й пользователь может вычислить секретный ключ d_j любого другого пользователя.

Шестая побочная атака.
Циклическая атака

Предположим, что известна лишь одна криптограмма *C*, полученная в криптосистеме RSA. Тогда злоумышленник может легко найти ее преобразования:

$$C_1 = C^{e^1} \bmod n;\ C_2 = C^{e^2} \bmod n;\ C_3 = C^{e^3} \bmod n, \ldots, C_r = C^{e^r} \bmod n.$$

Эти вычисления он продолжает до тех пор, пока результат не совпадет с исходной криптограммой *C*. (Это событие должно произойти рано или поздно на каком-то шаге k, так как шифрование – это по существу перестановка чисел $\{0, 1, 2, \ldots, n-1$.) Тогда он может найти сообщение как $M = Cek-1 \bmod n$, поскольку $Me = Cek = C$. Однако в [21] доказывается, что такой метод дает и факторизацию числа n, и поэтому при больших n этот подход не лучше прямого метода факторизации модуля КС RSA.

Седьмая побочная атака.
Отсутствие шифрования

Этот случай возможен, если в результате шифрования получаем открытое сообщение, т. е. $M^e \bmod n = M$. Такое условие должно выполниться хотя бы для одного из сообщений, например, для сообщений $M = 0,\ 1,\ \ldots,\ n-1$. На самом деле таких сообщений, которые *вообще!* не шифруются [21], существует в точности $[1+\gcd(e-1,p-11+\gcd e-1,q-1$. Их число всегда не менее 9. Однако при случайном выборе q и p доля таких сообщений будет ничтожно мала и они почти никогда не встретятся на практике.

Физические атаки на КС RSA

Атака, основанная на нахождении времени выполнения определенных операций

Такие атаки предполагают, что шифрование/дешифрование выполняются в аппаратной форме. Пусть существует некоторый невскрываемый чип дешифрования (рис. 3.4).

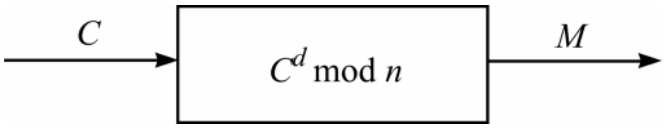


Рис. 3.4. Дешифрование при помощи невскрываемого чипа

Напомним, что быстрое возведение в степень состоит в последовательном возведении криптограммы в квадрат и умножениях на криптограмму по модулю n.

Пример 3.1. Рассмотрим дешифрование при помощи возведения в степень:

$$C^{171} = \left(\left(\left(\left(\left(\left((\mathrm{C})^2\right)^2 C\right)^2\right)^2 C\right)^2\right)^2 C\right)^2 C,$$

что соответствует выполнению следующих операций возведения в квадрат и умножения:

кв.→кв.→умнож.→кв.→кв.→умнож.→кв.→кв.→умнож.→кв.→умнож.

Типично то, что операция умножения требует больше времени, чем возведение в квадрат. Если злоумышленник может определить время выполнения последовательности операций при дешифровании в легальном чипе, например, контролируя потребление энергии или излучения в окружающее пространство, то он может восстановить и последовательность действий, а следовательно, и секретный ключ d. (Защитой от такой атаки может быть зашумление окружающего пространства или источников питания специальными шумовыми генераторами или нормализация времени выполнения всех операций.)

Атака внешним воздействием

Эта атака выполняется при помощи радиационного или электромагнитного излучения.

В некоторых случаях для сокращения времени производят вычисления по составляющим модуля с последующим использованием китайской теоремы об остатках, т. е. $M_1 = C^d \bmod p$, $M_2 = C^d \bmod q$, а затем находят $M = (M_1 \cdot a + M_2 \cdot b \bmod n$ при выполнении условий: a=1 mod p и a=0 mod q; b=0 mod p и b=1 mod q.

Предположим, что в момент вычисления M_1 внутри чипа все происходит правильно, а при вычислении M_2 возникает ошибка (скажем, за счет созданного злоумышленником электромагнитного импульса). Тогда $M_1 = \mathrm{C}^d \bmod p$, $M_2 \neq \mathrm{C}^d \bmod q$, и результат вычисления сообщения будет содержать ошибки, т. е. $\widetilde{M} = \left(a \cdot M_1 + b \cdot \widetilde{M}_2\right) \bmod n$. Далее злоумышленник, зная M и $\widetilde{M}$, находит:

$$(M - \widetilde{M}) \bmod p = a \cdot M_1 - a \cdot M_2 = 0 \bmod p;$$

$$(M - \widetilde{M}) \bmod q = b \cdot M_2 - b \cdot \widetilde{M}_2 = b \cdot (M_2 - \widetilde{M}_2) \bmod q \neq 0 \bmod q.$$

Отсюда следует, что нахождение $\gcd\left(M - \widetilde{M}, n\right) = p$ дает нетривиальную факторизацию n, т. е. его сомножитель p, поскольку $M - \widetilde{M}$ делится на p, но не делится на q, т. е. $\gcd(M - \widetilde{M}, n) \neq n$.

Выбор параметров для КС RSA

Как было отмечено ранее, правильный выбор параметров и режимов работы может обеспечить стойкость КС RSA для любых существующих вычислительных средств криптоанализа. При этом необходимо обратить внимание прежде всего на выбор величины модуля n и его множителей:

1) уместно выбрать битовую длину $l(n)$ модуля n более 768 (а для обеспечения высокой стойкости $l(n) \geq 1024$);

2) для того чтобы избежать атак с использованием некоторых специальных алгоритмов факторизации, числа p и q должны быть примерно одинаковой величины, т. е. порядка $\sqrt{n}$ (512 бит для высокой стойкости), причем их разность $(p - q)$ не должна быть малой, поскольку тогда эту разность можно будет найти перебором и затем факторизовать n. Иногда рекомендуется использовать так называемые *строго простые числа* p и q, которые удовлетворяют следующим условиям:

- $p - 1$ и $q - 1$ имеют большой простой делитель r;
- $p + 1$ и $q + 1$ имеют большой простой делитель x;
- $r - 1$ имеет большой простой делитель y.

Однако последние исследования показывают [21], что эти условия не являются необходимыми для обеспечения стойкости даже относительно всех атак при выборе $l(n) > 1024$;

3) выбор малых экспонент шифрования e (число $e = 3$ используется на практике, так как это требует одного возведения в квадрат и одного умножения), вообще говоря, допустим, но для защиты от соответствующей атаки необходимо тогда «подсаливать» сообщения.

Используют также экспоненту шифрования $e = 2^{16} + 1 = 65537$, что требует 16 возведений в квадрат и одного умножения. Данный метод, даже без «подсаливания» сообщений, имеет преимущество перед выбором экспоненты $e = 3$, поскольку используемая для малых экспонент атака будет эффективной, если только *одно и то же сообщение* шифруется и посылается одновременно 65537 пользователям (!), что, конечно, маловероятно;

4) при построении КС для группы пользователей, использующих шифры RSA, в случае когда именно центры распределения ключей снабжают их ключевыми данными, необходимо избегать распределения общих модулей.

3.2.2. Криптографическая система Рабина

Генерирование ключей

Эта процедура выполняется при помощи следующих шагов:

1) необходимо генерировать два простых числа p и q примерно одинаковой разрядности;

2) вычислить $n = p \cdot q$;

3) выбрать в качестве открытого ключа – n, а в качестве закрытого – его множители p, q.

Шифрование

Если пользователь B хочет зашифровать сообщение M для пользователя A, то он выполняет следующие шаги:

1) получает открытый ключ пользователя A, т. е. n;

2) представляет сообщение M в виде блоков такой длины, что сообщение из каждого блока может быть представлено целым числом $M_i \in (0,\ 1,\ 2,\ \dots,\ n-1)$.

3) вычисляет криптограмму $C = M_i^2 \bmod n$;

4) отправляет C пользователю A.

Дешифрование

Если пользователь A хочет расшифровать криптограмму C, то он выполняет следующие шаги:

1) зная p и q, реализует полиномиально сложный алгоритм для нахождения $\sqrt[2]{C}$, т. е. пользователь A находит четыре корня: M_1, M_2, M_3, M_4;

2) используя избыточность, содержащуюся в сообщении, определяет, какое из этих четырех сообщений является истинным. Если в сообщении отсутствует естественная избыточность, то ее необходимо ввести до шифрования (например, повторив несколько последних бит сообщения).

Замечание. Ранее был рассмотрен полиномиально сложный алгоритм для нахождения $\sqrt{a} \bmod p$, где p – простое число. Данный метод легко распространяется на случай, когда модуль – это произведение двух простых чисел. Такой модифицированный алгоритм выполняется следующими шагами:

1) вычисляют $\sqrt{a} \bmod p$, т. е. находят два корня $+r, -r$ по $\bmod p$;

2) аналогично вычисляют корни $+s, -s$ по $\bmod q$;

3) используя рассмотренный выше алгоритм, находят числа c и d, такие, что $c \cdot p + d \cdot q = 1$;

4) рассчитывают $x = (r \cdot d \cdot q + s \cdot c \cdot p) \bmod n$, $y = (r \cdot d \cdot q - s \cdot c \cdot p) \bmod n$;

5) в качестве решения $\sqrt{a} \bmod n$ выбирают числа $\pm x(\bmod n)$, $\pm y(\bmod n)$.

Видно, что схема Рабина обеспечивает простое генерирование ключей, а также весьма простой алгоритм шифрования, но имеет сложный алгоритм дешифрования. Данный метод можно рекомендовать в тех случаях, где шифрование должно быть простым, а при дешифровании этого не требуется.

Стойкость КС Рабина

Ясно, что если злоумышленник сможет факторизовать n, то он становится «легальным» пользователем, однако задача факторизации является сложной и не имеет пока полиномиальных решений. Таким образом, выбором, скажем, $l(n) = 768$ (или для большей стойкости $l(n) = 1024$) обеспечивается невозможность факторизации. Кроме того, для системы Рабина можно доказать и обратное утверждение.

Теорема [21]. Пусть $n = p \cdot q$, где p и q – простые числа, и пусть существует алгоритм R, который для каждого целого числа C, которое заведомо является квадратом некоторого числа x по $\bmod n$ (т. е. $x^2 = C \bmod n$), находит решение этого уравнения x при помощи $F(n)$ битовых операций. Тогда существует вероятностный алгоритм, который факторизует n со средним числом битовых операций $2\big(F(n) + \mathcal{O}\lg 2n$.

Доказательство. Выберем случайное целое число m, $0 < m < n$, и найдем $C = m^2 \bmod n$. Решим уравнение $x^2 = C \bmod n$, используя алгоритм R решения при помощи $F(n)$ операций. Обозначим через k найденные решения. Имеется четыре возможности (примем вероятность каждой из них $1/4$):

а) $k = +m \bmod p$ и $k = +m \bmod q$;

б) $k = +m \bmod p$ и $k = -m \bmod q$;

в) $k = -m \bmod p$ и $k = +m \bmod q$;

г) $k = -m \bmod p$ и $k = -m \bmod q$.

Для решения задачи факторизации подходят случаи б) и в). Тогда факторизацию можно выполнить, вычисляя для этих случаев:

б) $\gcd(k - m, n) = p$;

в) $\gcd(k - m, n) = q$.

Итак, при каждом случайном выборе m приходим к возможности факторизации n с вероятностью $1/2$. Так как известно, что сложность нахождения gcd требует $O(\lg^2 n)$ операций, то, учитывая сложность $F(n)$ выполнения алгоритма R, получаем $2\big(O(\lg^2 n) + F(n)\big)$ операций, необходимых для факторизации.

Таким образом, стойкость КС Рабина строго эквивалентна задаче факторизации и поэтому ее можно назвать *доказуемо секретной криптосистемой.* Заметим, что КС Рабина, так же как и КС RSA, имеет побочные атаки (при малом количестве сообщений, при использовании общих модулей и т. д.) [21].

3.2.3. Метод распределения ключей Диффи–Хеллмана

Недостатком всех КС с открытым ключом является относительная сложность шифрования и дешифрования по сравнению с симметричными КС, что приводит к большим затратам времени при выполнении таких процедур. Это особенно существенно при шифровании больших объемов информации (например, содержащейся на жестких дисках компьютеров или в больших базах данных). Преимуществом же КС с открытым ключом является простой метод распределения ключей. Для объединения положительных свойств симметричных и несимметричных КС используют так называемые *гибридные КС*, в которых распределение ключей осуществляется при помощи несимметричного алгоритма, а собственно шифрование – при помощи симметричного алгоритма. Так как обновление ключей требуется сравнительно редко, то гибридная система обеспечивает практически такую же скорость шифрования/дешифрования, как и симметричная КС.

Для распределения ключей можно использовать любую КС ОК, например RSA или Рабина. Тогда гибридная КС будет иметь вид, показанный на рис. 3.5.

На таком принципе построена, например, система PGP, используемая для шифрования электронной почты. Однако для решения задачи только по распределению ключей к симметричным шифрам можно применить другой асимметричный алгоритм, называемый распределением ключей *Диффи–Хеллмана*. Перед выполнением

этого алгоритма пользователи A и B согласуют открытые параметры $\alpha \in Z_p$ и p, где p – простое число. Далее выполняется протокол, показанный на рис. 3.6.

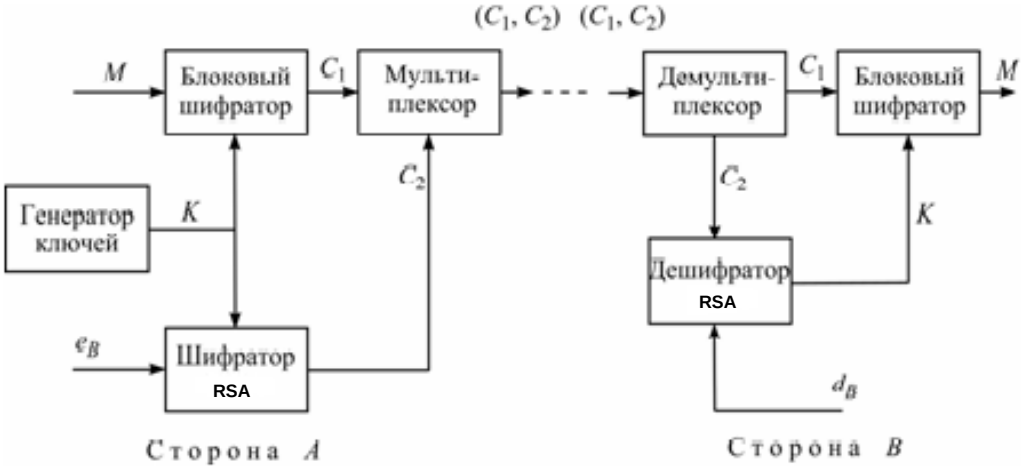


Рис. 3.5. Гибридная криптосистема

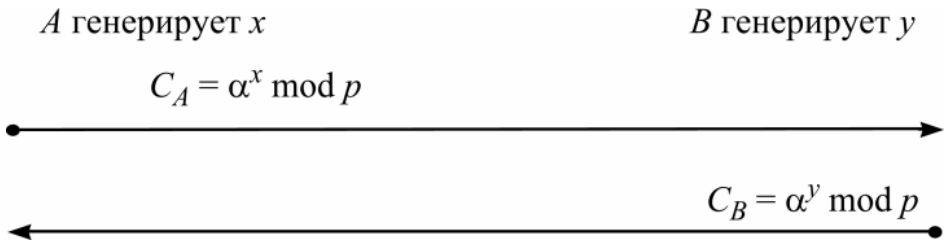


Рис. 3.6. Распределение ключей по методу Диффи–Хеллмана

После получения соответственно C_A и C_B пользователи A и B вычисляют общий ключ K следующим образом:

$$\begin{cases} K_A = C_B^x \bmod p = (\alpha^y)^x \bmod p = \alpha^{xy} \bmod p; \\ K_B = C_A^x \bmod p = (\alpha^x)^y \bmod p = \alpha^{xy} \bmod p, \end{cases} \quad K = K_A = K_B.$$

Стойкость криптографической системы Диффи–Хеллмана

Если злоумышленник умеет в обозримое время вычислять дискретный логарифм, то он может найти и секретный ключ A или B, поскольку $x = \log_\alpha C_A \bmod p$. Однако поскольку задача дискретного логарифмирования является трудной, данный способ при больших величинах p нереализуем. Вместе с тем стойкость КС Диффи–Хеллмана не эквивалентна задаче факторизации, а соответствует так называемой *проблеме Диффи–Хеллмана,* которая формулируется следующим образом: *зная* p, α, $\alpha^x \bmod p$, $\alpha^y \bmod p$, *нужно найти* $\alpha^{xy} \bmod p$. (Впрочем, последняя задача по сложности несущественно уступает задаче дискретного логарифмирования.)

Важно отметить, что метод распределения ключей Диффи–Хеллмана может быть полностью скомпрометирован активными злоумышленниками, выдающими себя за пользователей A или B. Если этот факт подмены (или имитации) величин

C_A, C_B не будет обнаружен, то вырабатывается совместный ключ не между A и B, а между злоумышленником и одним из пользователей. Таким образом, для КС Диффи–Хеллмана, так же как и для всех КС ОК, необходимо обеспечивать подлинность открытых данных (т. е. обеспечить решение задач аутентификации).

3.2.4. Криптографическая система Эль-Гамаля

Это некоторая модификация КС RSA, стойкость которой не связана непосредственно с проблемой факторизации. Она широко используется в стандартах цифровой подписи и допускает естественное обобщение на случай КС, построенных на основе использования эллиптических кривых, что будет рассмотрено далее.

Генерирование ключей

Пользователь *A* проделывает следующие операции для генерирования ключей:

1) генерирует простое число p и примитивный элемент $\alpha \in GF(p)$;

2) выбирает случайное число a такое, что $1 \le a \le p-2$, и вычисляет число α^a;

3) в качестве открытого ключа выбирает набор: $(p, \alpha, \alpha^a \bmod p)$, а в качестве закрытого ключа – число a.

Шифрование

Пользователь B выполняет следующие шаги для шифрования сообщения M, предназначенного пользователю A:

1) получает открытый ключ A;

2) представляет сообщение M в виде цепочки чисел M_i, каждое из которых не превосходит $p-1$;

3) выбирает случайное число k такое, что $1 \le k \le p-2$;

4) вычисляет $\gamma = \alpha^k \bmod p$, $\delta = M_i \cdot (\alpha^a)^k \bmod p$;

5) посылает криптограмму $C = (\gamma, \delta)$ пользователю A.

Дешифрование

Пользователь A выполняет следующие шаги для дешифрования сообщения, полученного от пользователя B:

1) используя свой закрытый ключ, вычисляет $\gamma^{-a} \bmod p$;

2) восстанавливает сообщение $M_i = \gamma^{-a}\delta \bmod p$.

Действительно, $\gamma^{-a}\delta = \alpha^{-ak} M_i \alpha^{ak} = M_i \bmod p$.

Особенностью схемы Эль-Гамаля является то, что она относится к так называемым схемам *рандомизационного шифрования*, поскольку при шифровании в ней используется дополнительная случайность в виде числа k.

(Считается, что рандомизационное шифрование более стойко по отношению к некоторым методам криптоанализа, например к таким как статистические атаки [21].)

Преимущество КС Эль-Гамаля состоит также и в том, что тогда все пользователи в сети могут выбирать одинаковые α и p, что невозможно для КС RSA. Кроме того, как будет показано далее, эта схема может быть естественным образом распространена на случай эллиптических кривых.

Существенным недостатком схемы является то, что длина криптограммы в ней в 2 раза больше длины сообщения.

Стойкость криптографической системы Эль-Гамаля

Проблема восстановления сообщения M по заданным p, α, α^a, δ и γ при неизвестном a эквивалентна решению задачи Диффи–Хеллмана.

Ясно также, что если будет решена проблема нахождения дискретного логарифма, то криптосистема Эль-Гамаля будет вскрыта. При выборе p с разрядностью 768 бит (для повышенной стойкости – до 1024 бит), стойкость КС Эль-Гамаля будет такой же, как и у КС RSA при выборе в последней тех же параметров для модуля.

Важно отметить, что для шифрования различных сообщений M_i, M_j необходимо использовать различные значения чисел k, поскольку в противном случае $\frac{\delta_1}{\delta_2} = \frac{M_i}{M_j}$, и тогда сообщение M_j может быть легко найдено, если известно сообщение M_i.

3.2.5. Построение криптографических систем на основе использования эллиптических кривых

Элементы теории эллиптических кривых над конечными полями

Определение 3.2. Пусть $GF(q)$, $q = p^n$, – некоторое конечное поле, причем $p \neq 2$. Тогда *эллиптической кривой* E над полем $GF(q)$ называется множество пар элементов (x, y), $x, y \in GF(q)$, которые удовлетворяют уравнению

$$y^2 = x^3 + ax^2 + bx + c, \tag{3.5}$$

где $a,\ b,\ c \in GF(q)$.

Если $p \neq 2, 3$, то уравнение (3.5) приводится к виду

$$y^2 = x^3 + bx^2 + c. \tag{3.6}$$

Если $p = 2$, то уравнение (3.5) будет иметь следующий вид:

$$y^2 + xy = x^3 + ax^2 + c. \tag{3.7}$$

Пример 3.2. Пусть нужно проверить, что точка $x = 1$, $y = 4$ лежит на эллиптической кривой $y^2 = x^3 + 2x + 3$ над полем $GF(5)$. Тогда находим:

$$y^2 \bmod 5 = 1;\ x^3 \bmod 5 = 1;\ 2x = 2;\ 1 + 2 + 3 \bmod 5 = 1 = 1 \bmod 5.$$

Видно, что уравнение (3.6) удовлетворяется, и, следовательно, точка лежит на кривой. Эллиптическая кривая имеет не более чем $2q + 1$ точек (одна точка в бесконечности и для каждого x не более двух значений y, удовлетворяющих уравнению (3.5)). Однако реально их значительно меньше.

Число точек на эллиптической кривой над полем $GF(q)$ можно оценить при помощи *теоремы Хассе* [19].

Теорема. Пусть N – число точек эллиптической кривой над полем $GF(q)$. Тогда имеем оценку

$$|N-(q-1)| \leq 2\sqrt{q}. \tag{3.8}$$

Важным свойством эллиптических кривых является то, что между точками этих эллиптических кривых можно задать операции *сложения* и нахождения *противоположной точки*.

Множество точек на эллиптической кривой образуют так называемую *группу* относительно операции специфического сложения, заданной на эллиптической кривой. Далее нам потребуется определение понятия группы.

Определение 3.3. Конечной группой G называется конечное множество элементов $g \in G$, на котором задано сложение между элементами, противоположные элементы $-g$ и нейтральный элемент 0, которые удовлетворяют следующим аксиомам.

Если $g, g' \in G$, то

$$g + g' \in G;\ g + (g' + g") = (g + g') + g";$$
$$g + 0 = g;\ g + (-g) = 0.$$

Если $g + g' = g' + g$, то группа называется *Абелевой.*

Легко видеть, что в любом конечном поле операции относительного сложения и умножения образуют конечные Абелевы группы.

Определение операции нахождения противоположного элемента и суммы элементов на эллиптической кривой. Пусть эллиптическая кривая задана уравнением $y^2 = x^3 + bx + c$. Для того чтобы лучше понять, как выполняется сложение точек и нахождение противоположной точки, рассмотрим сначала эллиптические кривые, заданные над полем вещественных чисел, которые тогда можно изобразить геометрически, как это показано на рис. 3.7.

Для любой точки P эллиптической кривой с координатами $(x,\ y)$ обратная точка $-P$ будет иметь (по определению) координаты $(x,\ -y)$ (рис. 3.7).

Для двух точек P и Q на эллиптической кривой, которые имеют соответственно координаты $(x_1,\ y_1)$ и $(x_2,\ y_2)$, причем $x_1 \neq x_2$, их сумма определяется геометрически, как точка, обратная той, которая получается на пересечении прямой линии, проходящей через точки P, Q, и эллиптической кривой еще в одной точке R (рис. 3.7, а).

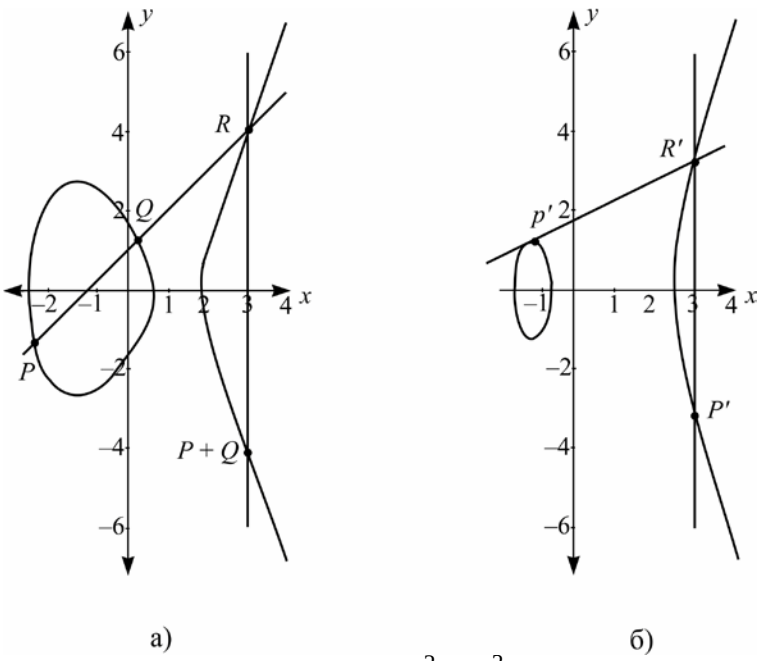


Рис. 3.7. Эллиптическая кривая $y^2 = x^3 – 5x + 3$ на плоскости:
а) сложение различных точек;
б) сложение одинаковых точек

Если $x_1 = x_2$, а это означает, что $P = -Q$, то, по определению, $P + Q = O$, где O – точка в бесконечности. Если же $P = Q = P'$, то $P' + P' = -R'$, где R' – точка, которая получается как пересечение касательной к эллиптической кривой в точке P' и другой ветви эллиптической кривой (рис. 3.7, б). Если касательная в точке P' является «одиночной», то $R' \neq P'$, а если это «двойная» касательная, т. е. когда P' – это точка перегиба кривой, то $R' = P'$.

Все эти геометрические построения могут быть заданы в аналитической форме, которая позволяет найти координаты точки, противоположной данной или равной сумме двух точек, заданных на эллиптической кривой.

Рассмотрим сначала случай $p \neq 2$, т. е. когда эллиптическая кривая задается уравнением (3.5). Пусть также $P \neq Q$, где $P(x_1, y_1)$ и $Q(x_2, y_2)$ – две разные точки на эллиптической кривой. Тогда

$$y - y_1 = \lambda(x - x_1);\ \lambda = \frac{y_2 - y_1}{x_2 - x_1};$$

$$\left(\lambda(x - x_1) + y_1\right)^2 = x^3 + ax^2 + bx + c$$

и для координат точки, равной сумме точек P и Q, получаем уравнения:

$$x_3 = \lambda^2 - a - x_1 - x_2;\ y_3 = \lambda(x_3 - x_1) + y_1. \qquad (3.9)$$

Если $p = 2$, т. е. эллиптическая кривая задается уравнением (3.7), то для координат точки, равной сумме точек, получаем уравнения:

$$x_3 = a - \lambda^2 - \lambda - x_1 - x_2;\ y_3 = \lambda(x_3 - x_1) + y_1. \qquad (3.10)$$

Аналогичным образом можно вывести аналитическое уравнение для касательной (используя понятие формальной производной) и затем получить выражения для

суммы точек, когда $P = Q$ [19]. Тогда в случае $p \neq 2$ получаем для координат точки-суммы то же выражение (3.8), но с другим значением:

$$\lambda = (3x_1^2 + 2ax_1 + b)/2y_1. \quad (3.11)$$

Для случая $p = 2$ получаем другие выражения для координат точки-суммы:

$$x_3 = x_1^2 + c/x_1^2;\ \ y_3 = x_1^2 + (x_1 + y_1/x_1)x_3. \quad (3.12)$$

Очевидно, что все операции, заданные в уравнениях для нахождения координат точек суммы могут выполняться и над конечным полем $GF(p^n)$, что и требуется для построения КС ОК, тогда как эллиптические кривые над полем вещественных чисел и их геометрическая интерпретация были полезны лишь для наглядности.

«Возведение в k-ю степень» точки P на эллиптической кривой понимается как k-кратное сложение этой точки с самой собой на этой кривой: «P^k» $= P + P + \ldots + P$.

Для быстрого вычисления степени точек на эллиптической кривой при больших значениях показателей можно использовать технику быстрого возведения в степень, рассмотренную ранее, которая состояла в двоичном разложении числа k, последующем сложении и возведении в квадрат.

Пример 3.3. Пусть $k = 171 = 10101011$, $P \in E$ над $GF(q)$. Тогда

$$2P = P + P;\ 4P = 2P + 2P;\ 128P = 64P + 64P;\ 8P = 4P + 4P;$$
$$171 \cdot P = 2\Big(2\Big(2\big(2\big(2(2(2P) + P)\big) + P\big)\Big) + P\Big) + P.$$

Ранее было дано определение Абелевой группы относительно операций сложения. Из рассмотренных свойств точек, принадлежащих эллиптической кривой над полем $GF(q)$, следует, что множество таких точек образует Абелеву группу относительно сложения, причем порядок этой группы (т. е. число ее элементов) не совпадает с порядком поля q. Для многих криптосистем с открытым ключом операции шифрования и дешифрования фактически выполняются на Абелевых группах, и этот факт открывает возможности для построения КС ОК на основе использования эллиптических кривых.

Задание КС над эллиптическими кривыми

Теория эллиптических кривых может быть полезной для решения различных задач, связанных с построением КС ОК, а именно для задач:

- тестирования простых чисел;
- факторизации больших чисел;
- вычисления дискретных логарифмов;
- построения новых криптосистем с ОК;
- построения новых систем с цифровыми подписями.

Далее рассмотрим только две последние задачи. К эллиптическим кривым E, над которыми строятся КС ОК, обычно предъявляются следующие требования:

- кривые рассматриваются либо над простыми полями $GF(p)$, либо над полями $GF(q)$ характеристики $p = 2$, когда $q = 2^n$;

- исходя из соображения стойкости целесообразно использовать так называемые *несуперсингулярные кривые*, которые задаются следующими уравнениями:

$$y^2 = x^3 + ax + b \text{ для } p > 2;\ y^2 + xy = x^3 + ax^2 + b \text{ для } p = 2,$$

где $a,\ b \in GF(q)$;

- кривая должна иметь точку высокого порядка (2^{160}) относительно операции сложения на этой кривой (отметим, что порядком точки называется ее порядок как элемента Абелевой группы);

- в качестве коэффициентов $a,\ b$ можно выбирать (0, 1), но рекомендуется генерировать их случайным образом.

Обобщение КС Эль-Гамаля на случай эллиптических кривых

Для того чтобы создать КС Эль-Гамаля на эллиптической кривой, необходимо выбрать достаточно большое поле и задать уравнение этой кривой. Число точек на этой кривой должно быть непереборно велико. Алгоритм случайной генерации кривой с требуемыми свойствами подробно описан в [22].

Генерирование ключей

1. Генерируется точка $\alpha \in E$ большого порядка (см. третье требование).

2. Выбирается случайное целое число a такое, что $1 \le a \le q - 2$, и вычисляется $a \cdot \alpha$, что означает сложение точки $\alpha \in E$ с самой собой a раз по правилам сложения, задаваемым уравнениями, аналогичными (3.2)–(3.5).

3. Данные E, $\alpha \in E$, $a \cdot \alpha$ представляют собой открытый ключ, тогда как a – закрытый ключ.

Шифрование

Пользователь B шифрует сообщение M для пользователя A следующим образом:

1) получает открытый ключ пользователя A, т. е. $(\alpha, a \cdot \alpha)$;

2) представляет сообщение M как точку (или как точки) на эллиптической кривой E, для чего данные размещает в координате x, а координату y выбирает так, чтобы точка (x, y) лежала на заданной кривой E;

3) выбирает большое целое случайное число k;

4) вычисляет $\gamma = k \cdot \alpha$, $\delta = M + k \cdot a \cdot \alpha$;

5) посылает пользователю A криптограмму $C = (\gamma, \delta)$.

Дешифрование

Пользователь A дешифрует криптограмму следующим образом:

1) используя свой секретный ключ a, вычисляет $-\gamma \cdot a = b$;

2) восстанавливает сообщение $M = b + \delta$.

Стойкость КС Эль-Гамаля, реализованной на эллиптических кривых

Определение 3.4. Пусть E – эллиптическая кривая, P – точка на этой кривой и n – целое неотрицательное число. Тогда *проблемой дискретного логарифмирования над заданной эллиптической кривой E* называется решение относительно n уравнения

$$nP = Q, \tag{3.13}$$

где P и Q известны.

Ясно, что если криптоаналитик в состоянии решить задачу логарифмирования над эллиптической кривой, то КС, описанная выше, будет взломана. Однако решение этой задачи имеет сложность порядка n^{β}, $\beta > 0$, что неизмеримо сложнее, чем вычисление nP при известных n, P. Более того, для построения стойкой КС Эль-Гамаля над эллиптическими кривыми достаточно выбрать битовую длину цепочки, описывающей n порядка 150–180 [22], тогда как для «классической» КС Эль-Гамаля необходимо выбирать представление модуля p порядка 768 бит. Именно то обстоятельство, что «логарифмирование» над эллиптическими кривыми значительно сложнее логарифмирования над конечными полями, и создает перспективы использования таких кривых в КС ОК.

Построение системы распределения ключей Диффи–Хеллмана над эллиптическими кривыми

Предварительно пользователи согласуют между собой эллиптическую кривую E над полем $GF(q)$ и точку P на этой кривой, так же как это делалось при построении КС Эль-Гамаля. Далее пользователь A случайно генерирует целое положительное число n_A, $1 \le n_A \le q - 2$, и посылает по каналу связи к B «точку» эллиптической кривой $P_a = n_A P$. В свою очередь, пользователь B случайно генерирует целое положительное число n_B, $1 \le n_B \le q - 2$, и посылает по каналу связи к A «точку» эллиптической кривой $P_b = n_B P$. Наконец A вычисляет ключ $K_A = n_A P_b$, а B – ключ $K_B = n_B P_a$, которые, очевидно, будут совпадать и используются как общий ключ между A и B.

Стойкость такого метода распределения ключей будет определяться сложностью решения задачи дискретного «логарифмирования» над эллиптическими кривыми, и при соответствующем выборе поля $GF(q)$, самой эллиптической кривой E и точки P на ней она может быть сделана достаточно высокой.

3.2.6. Криптографическая система с открытым ключом Мак-Элиса

В отличие от предыдущих КС ОК данная КС использует сложность решения задачи по *исправлению ошибок линейными кодами* для обеспечения ее стойкости.

Краткие сведения о линейных кодах **[23]**

Линейный двоичный $(n,\ k)$ код V (ЛК) – это множество, состоящее из 2^k двоичных последовательностей $\bar{y}$ длиной n каждая, для которых справедливо условие:

$$\text{если } \left.\begin{matrix}\bar{y}_1 \in V,\\ \bar{y}_2 \in V,\end{matrix}\right\} \text{ то } \bar{y}_1 \oplus \bar{y}_2 \in V,$$

где $\oplus$ означает поэлементное суммирование по $\mathrm{mod}\ 2$.

Каждый ЛК может быть однозначно задан своей *порождающей* $k \times n$ *матрицей* G, состоящей из двоичных элементов 0 и 1 (рис. 3.8).

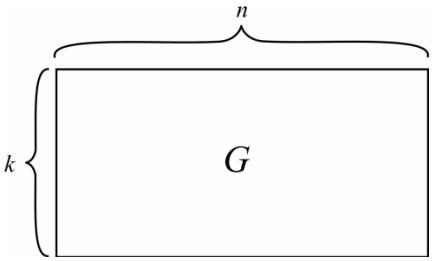


Рис. 3.8. Порождающая матрица линейного кода

Если G – порождающая матрица кода V, а $\bar{x}$ – двоичная информационная последовательность длины k, то кодирование (т. е. преобразование ее в кодовую последовательность $\bar{y}$) производится по правилу

$$\bar{y} = \bar{x} \cdot G, \tag{3.14}$$

где операции умножения вектора на матрицу выполняются в поле $GF(2)$.

Линейный код может быть представлен в систематической форме, когда каждое его слово состоит из k *информационных* символов $\bar{x}_k$ и следующих за ними $n-k$ *проверочных* символов $\bar{c}_{n-k}$, которые формируются по информационным при помощи линейных операций над полем $GF(2)$. Эти линейные соотношения являются нетривиальной частью так называемой *проверочной матрицы* кода, которая однозначно определяется по его порождающей матрице [23].

Если в кодовом слове $\bar{y}$ возникает ошибка $\bar{e}$ (при его передаче по каналу связи или при хранении), т. е. $\bar{y}\langle\ \rangle$ (где $\bar{e}$ имеет единицы в позициях с ошибками и нули в остальных позициях, а $\oplus$ означает поэлементное суммирование по $\mathrm{mod}\ 2$), то, используя свойства кода, некоторые ошибки можно исправить (восстановить информационную последовательность $\bar{x}$ по искаженному кодовому слову $\bar{y}$). Исправляющие свойства кода задаются его так называемым *минимальным кодовым расстоянием* $d_{\min}$, которое определяется как минимальное число позиций, в которых отличаются любые несовпадающие между собой кодовые комбинации.

Если код имеет минимальное кодовое расстояние $d_{\min}$, то он гарантированно (т. е. полностью) исправляет все ошибки кратности не более $t = \left[\frac{d_{\min}}{2}\right]$, где $[x]$ озна-

чает целую часть числа x, а кратность ошибки – число искаженных позиций, т. е. число единиц в образце ошибки $\bar{e}$.

В теории кодирования [23] разработаны методы построения ЛК (т. е. фактически их порождающих матриц), которые обеспечивают максимально возможные величины $d_{\min}$ при заданных параметрах $(n,\ k)$.

Как видно из соотношения (3.14), процедура кодирования для ЛК оказывается достаточно простой. Более того, для некоторых классов ЛК (например, для так называемых *циклических* кодов) она может быть еще более упрощена [23].

Однако процедура декодирования (т. е. исправления ошибок) остается весьма сложной для произвольных линейных кодов. Действительно, если известно, что данный код имеет заданное $d_{\min}$ и произошла ошибка кратности $t \le [d_{\min}/2]$, то для декодирования достаточно вычислить *расстояния Хэмминга* (т. е. число различных позиций) между принятым искаженным словом $\bar{y}$ и всеми возможными кодовыми словами $\bar{y} \in V$ и принять решение о передаче того кодового слова, для которого это расстояние минимально. Однако число кодовых слов в коде с параметрами $(n,\ k)$ равно 2^k и при больших значениях k это число оказывается непереборно большим, т. е. такая процедура является практически нереализуемой. Более того, в теории вычислительной сложности [23] доказывается утверждение, что если бы для любых ЛК был найден полиномиально сложный алгоритм исправления ошибок по *минимуму расстояния Хэмминга*, то его можно было бы использовать и для решения множества других трудных задач, которые до сих пор не имеют полиномиально сложных решений. Поэтому задача нахождения простого алгоритма декодирования для произвольных ЛК является весьма сложной.

Для того чтобы упростить процедуру исправления ошибок в задачах связи, используют *подоптимальные алгоритмы* декодирования для подклассов ЛК (например, для так называемых *кодов Гоппы*, для которых сложность декодирования задается соотношением $O(n^2)$ [3]).

Особенность ЛК, выражающаяся в простоте кодирования и сложности декодирования, где последняя может быть преодолена при переходе к некоторым подклассам ЛК и используется для построения КС Мак-Элис.

Описание КС Мак-Элис

Если пользователь A хочет сгенерировать свою пару открытый/закрытый ключ, то эта процедура реализуется следующими шагами:

1) генерируется случайная порождающая матрица G_A для специального подкласса двоичных $(n,\ k)$-кодов (скажем, для кода Гоппы), которые гарантированно исправляют t_A ошибок и имеет полиномиально сложный алгоритм декодирования;

2) генерируется случайная двоичная *несингулярная* (т. е. имеющая ненулевой определитель) матрица S_A размером $k \times k$;

3) генерируется случайная *перестановочная* $n \times n$ матрица P_A (перестановочной называется такая матрица, произведение которой на любой вектор дает лишь перестановку его позиций. Такая матрица содержит в каждой строке и в каждом столбце по одной единице, а остальные ее элементы – нули.);

4) вычисляется матрица $\widehat{G}_A = S_A \cdot G_A \cdot P_A$ (ее размерность будет $k \times n$);

5) публикуется открытый ключ $\widehat{G}_A$, t_A, и сохраняется в тайне секретный ключ (S_A, G_A, P_A).

Шифрование КС Мак-Элис

Если пользователь B хочет зашифровать сообщение M для пользователя A, то он должен выполнить следующие шаги:

1) получить от A открытый ключ ($\widehat{G}_A$, t_A);

2) преобразовать сообщение M в последовательность двоичных блоков M_i длины k. Далее для каждого из полученных блоков проделать следующие шаги 3–5;

3) сгенерировать случайный двоичный вектор Z_i длины n и *веса* (т. е. числа единиц в нем) не более t_A;

4) вычислить двоичный вектор $C_i = M_i \widehat{G}_A \oplus Z_i$;

5) послать вектор C_i к пользователю A как криптограмму для сообщения M_i.

Дешифрование КС Мак-Элис

Для того чтобы восстановить сообщение M_i по криптограмме C_i, пользователь A должен выполнить следующие шаги:

1) вычислить $\widehat{C}_i = C_i \cdot P_A^{-1}$, где P_A^{-1} – это матрица, обратная P_A;

2) используя известный алгоритм декодирования для кода с порождающей матрицей G_A, исправить не более t_A ошибок в $\widehat{C}_i$, что даст некоторый двоичный вектор $\widehat{M}_i$длины k;

3) восстановить $M_i = \widehat{M}_i \cdot S_A^{-1}$.

Для доказательства того, что описанная выше процедура действительно восстанавливает зашифрованное сообщение M_i, преобразуем сначала представление для $\widehat{C}_i$:

$$\widehat{C}_i = C_i \cdot P_A^{-1} = (M_i \widehat{G}_A \oplus Z_i)\ P_A^{-1} = (M_i S_A G_A P_A \oplus Z_i) \cdot P_A^{-1} = \\ = (M_i S_A)\ G_A \oplus Z_i \cdot P_A^{-1}. \quad (3.15)$$

Из выражения (3.15) видно, что двоичный вектор $\widehat{C}_i$ представляет собой закодированное ЛК (с порождающей матрицей G_A) сообщение ($M_i S_A$) с добавкой двоичного шума $Z_i P_A^{-1}$ веса не более t_A, так как P_A^{-1} будет также перестановочной матрицей, умножение на которую сохраняет вес слова Z_i, который был определен ранее не более чем t_A.

Поскольку код с порождающей матрицей G_A может гарантированно исправить не менее t_A ошибок, это означает, что по $\widehat{C}_i$ пользователь A может абсолютно точно восстановить $(M_i S_A)$, причем сложность декодирования будет при этом полиномиальной. Наконец, исходное сообщение M_i восстанавливается после умножения последнего результата на S_A^{-1}, т. е.

$$M_i S_A \cdot S_A^{-1} = M_i.$$

Заметим, что в отличие от метода RSA и подобно тому, как это было в шифре Эль-Гамаля, метод Мак-Элис является рандомизационным, поскольку случайный вектор помехи Z_i не входит в состав ключей этой КС. Если на шаге 1 генерирования ключей используется семейство кодов Гоппы, то можно иметь в виду [3], что для любого неприводимого над полем $GF(2^m)$ многочлена $g(x)$ степени t_A существует двоичный код Гоппы длины $n = 2^m$ с числом информационных символов $k \geq n - mt_A$, где t_A – число ошибок, исправляемых этим кодом, причем этот код имеет полиномиально сложный алгоритм декодирования.

Стойкость КС Мак-Элис

Рассмотрим две основные атаки на КС Мак-Элис:

1) зная C_i, $\widehat{G}_A$, t_A, можно попытаться исправить ошибки в C_i, но поскольку, как легко убедиться, порождающая матрица $\widehat{G}_A$ является совершенно случайной, для определяемого ей кода не известно неперебoрных методов исправления ошибок, а переборные практически нереализуемы при соответствующем выборе параметров исходного кода;

2) для восстановления M_i можно попытаться случайно выбрать k столбцов в матрице $\widehat{G}_A$. Если теперь обозначим через G_k, C_{ik}, Z_k ограничение $\widehat{G}_A, C_i, Z_i$ этими столбцами, то будет выполняться уравнение $(\mathrm{C}_k + Z_k) = M_i \cdot \widehat{G}_k$. Если случайно окажется, что $Z_k = 0$ и $\widehat{G}_k$ несингулярна, то M_i может быть получено как решение уравнения $C_k = M \cdot \widehat{G}_k$. Однако вероятность того, что $Z_k = 0$, оказывается равной C_{n-t}^k / C_n^k, и при соответствующем выборе параметров она будет весьма малой величиной.

Для обеспечения высокой стойкости КС Мак-Элис рекомендуются следующие ее параметры [3]: $n = 1024$ бит, $t = 38$, $k > 644$.

Представленный ранее материал позволяет сформулировать следующие *основные свойства КС Мак-Элис:*

1) достаточно предсказуемая стойкость;

2) простота шифрования и дешифрования;

3) увеличение длины криптограммы по сравнению с длиной сообщений в n/k раз;

4) большая битовая длина как открытого, так и закрытого ключей.

Последнее свойство особенно существенно ограничивает практическое применение КС Мак-Элиса.

3. 3. Цифровые подписи с использованием криптографических систем с открытым ключом

Аутентификация сообщений с использованием симметричных не позволяет решить проблему *возможного спора* между создателем сообщения и его получателем. Действительно, получатель, обладая тем же секретным ключом, что и автор сообщения, может по своему желанию сформировать новое сообщение, добавить к нему аутентификатор и утверждать, что он получил именно это сообщение. Для разрешения проблемы подобного спора при использовании симметричных КС необходимо присутствие третьего участника (*доверенного лица*), которому должна посылаться копия сообщения. Поскольку такой сценарий в большинстве случаев является неудобным, необходимо использовать несимметричные КС, выполняя аутентификацию (в том числе и обеспечение подлинности сообщений) на основе так называемых *цифровых подписей* (ЦП).

Будем здесь рассматривать аутентификацию сообщений без обеспечения их конфиденциальности, поскольку такая задача может быть тривиально решена при помощи аутентификации уже зашифрованных сообщений.

ЦП – это некоторые дополнительные данные, присоединяемые к основному сообщению, которые формируются зависящими как от сообщения, так и от секретного ключа автора сообщения. Для проверки подлинности сообщения (называемой иначе процедурой *верификации*) используется открытый ключ автора сообщения, который может быть доступен любому пользователю.

3.3.1. Основные требования, предъявляемые к цифровым подписям

К цифровой подписи естественно предъявить следующие требования:

1) уникальность (т. е. возможность использования ее только одним определенным пользователем);

2) невозможность отказа от выполненной ЦП;

3) допустимость верификации ЦП любым пользователем;

4) простота выполнения процедур создания и верификации ЦП;

5) возможность выполнения ЦП для больших объемов информации (файлов, дисков и т. д.).

Как видно из перечисленных выше требований, ЦП применительно к данным и документам, представленным в электронной форме, является почти полным эквивалентом обычной *бумажной подписи*. Однако между ними имеется одно важное и принципиальное различие: *бумажная подпись не зависит от содержания сообщения, тогда как ЦП зависит.* Именно эта особенность ЦП и позволяет обес-

печить невозможность ее подделки при помощи простого копирования в электронной форме и переноса в другое сообщение!

Стойкость ЦП понимается как ее способность противостоять необнаруженной преднамеренной подделке или случайному искажению, причем для ЦП, основанных на использовании КС ОК, под стойкостью понимается *вычислительная стойкость*, т. е. невозможность появления необнаруженной подделки при ограниченном вычислительном ресурсе злоумышленника.

Существует множество алгоритмов формирования ЦП, основанных на различных КС ОК, к рассмотрению некоторых из них мы сейчас и переходим.

3.3.2. Цифровые подписи на основе различных криптографических систем с открытым ключом

ЦП на основе КС RSA

Пусть имеется некоторое сообщение $\overline{M}$ и некоторым пользователем A сгенерирована пара открытый/закрытый ключ для системы RSA, т. е. числа e_A, n_A; d_A. Тогда сообщение $\overline{M}$ разбивается на блоки, каждый из которых может быть представлен целым числом, не превосходящим n_A. Для каждого из таких сообщений-цифр M формируется ЦП S по следующему правилу: $S = M^{d_A} \bmod n_A$. Далее ЦП присоединяется к сообщению, образуя так называемое *подписанное сообщение*, т. е. пару M, S. Для верификации ЦП пользователь должен получить открытый ключ A, а также «подписанное» (но возможно фальсифицированное) сообщение $\widetilde{M}$, $\tilde{S}$ и вычислить $\widetilde{M}\tilde{S}^{e_A} \bmod e_A$. Далее он сравнивает $\widetilde{M}$ с $\widetilde{M}$ и при их совпадении полагает, что сообщение $\widetilde{M}$ действительно подписано A, в противном случае отвергает его, как подделку или искажение. Правомочность такой верификации для неискаженных подписанных сообщений основывается непосредственно на алгоритме дешифрования КС RSA.

Стойкость ЦП данного типа очевидно эквивалентна стойкости КС RSA, а последняя, в свою очередь, основывается на сложности решения проблемы факторизации или логарифмирования. Имеются также и побочные атаки на ЦП, похожие на побочные атаки на КС RSA [3].

Интересно отметить, что фактически проверка ЦП здесь сводится к дешифрованию сообщения, и поэтому для сообщений, содержащих естественную или искусственно введенную избыточность, можно формировать и передавать при необходимости только ЦП S, обеспечивая этим и выполнение функции его конфиденциальности.

ЦП на основе КС Эль-Гамаля

Поскольку такая ЦП является более сложной и практически важной, рассмотрим ее подробнее, включая и алгоритм генерирования пар ключей.

Генерирование ключей:

1) генерируется большое простое число p и примитивный элемент α над конечным полем $GF(p)$;

2) генерируется число a: $1 \le a \le p-2$;

3) вычисляется $y = \alpha^a \bmod p$;

4) выбирается открытый ключ верификации ЦП: (p, a, y) и секретный ключ создания ЦП: a.

Формирование ЦП

Если пользователь A хочет подписать сообщение M, представленное в виде числа, принадлежащего Z_p, то он выполняет следующие операции:

1) генерирует секретное число k: $1 \le k \le p-2$, $\gcd(k, p-1) = 1$;

2) вычисляет $r = \alpha^k \bmod p$;

3) вычисляет $k^{-1} \bmod (p-1)$;

4) вычисляет $t = k^{-1}(M - ar) \bmod (p-1)$;

5) Формирует ЦП S к сообщению M как пару чисел $S = (r, t)$.

Проверка (верификация) ЦП

Для того чтобы проверить подпись S, созданную A под сообщением M, любой пользователь выполняет следующие шаги:

1) получает открытый ключ A: (p, α, y);

2) проверяет, что $1 \le r \le p-1$, и если это не выполняется, то отвергает ЦП;

3) рассчитывает $\nu_1 = y^r r^t \bmod p$;

4) рассчитывает $\nu_2 = a^M \bmod p$;

5) принимает ЦП как правильную при условии, что $\nu_1 = \nu_2$.

Покажем, что изложенный выше метод верификации ЦП действительно даст правильный ответ для корректно созданной ЦП.

Если ЦП была сгенерирована A, то $t = k^{-1}(M - ar) \bmod (p-1)$. Умножая обе части этого равенства на k, получаем $tk = (M - ar) \bmod (p-1)$. После переноса M в левую часть, имеем $M = (ar + kt) \bmod (p-1)$, что эквивалентно равенству $M = (p-1) \cdot l + ar + kt$, где l – целое число.

Найдем теперь v_2, полученное на шаге 4 алгоритма верификации, используя теорему Ферма для дальнейших преобразований

$$\nu_2 = a^M \bmod p = a^{(p-1)\cdot l + ar + kt} \bmod p = \alpha^{ar+kt} \bmod p =$$
$$= (\alpha^a)^k \cdot (\alpha^k)^t = y^r \cdot r^t \bmod p = \nu_1.$$

что и доказывает правильность верификации.

Стойкость ЦП на основе КС Эль-Гамаля

Злоумышленник может попытаться подделать подпись к своему сообщению M следующим образом: сгенерировать случайное число k, вычислить $r = \alpha^k \bmod p$, а затем попытаться найти

$$t = k^{-1}(M - ar) \bmod (p-1).$$

Однако для выполнения последней операции ему необходимо знать a, что возможно лишь после решения задачи *дискретного логарифмирования* (см. шаг 3 генерирования ключей). При соответствующем выборе параметров ЦП эта задача оказывается вычислительно нереализуемой.

Выполняя ЦП к различным сообщениям, необходимо генерировать различные случайные числа k, поскольку в противном случае секретный ключ ЦП легко вычисляется. Действительно, если это не так, то получаем следующие равенства:

$$t_1 = k^{-1}(M_1 - ar) \bmod (p- \quad t_2 = k^{-1}(M_2 - ar) \bmod (p-1),$$

из которых получаем, что $(t_1 - t_2)k = M_1 - M_2 \bmod (p-1)$.

Если $t_1 - t_2 \neq 0 \bmod (p-1)$, что весьма вероятно, то

$$k = (t_1 - t_2)^{-1}(M_1 - M_2) \bmod (p-1).$$

После нахождения k вычисление секретного ключа a оказывается весьма простым по известному числу t (см. шаг 4 генерирования ЦП). Существование такой атаки требует хранения в секрете случайных чисел k или их уничтожения сразу же после генерирования ЦП, поскольку если злоумышленник будет иметь к ним доступ, то он легко найдет секретный ключ и сможет выполнять подписи под любыми сообщениями от лица законного пользователя.

Заметим, что не представляет труда подделать подпись под бессмысленным сообщением $M' = tu \bmod (p-1)$, где u – произвольное число. Действительно, для такого сообщения число $r = \alpha^u y^v \bmod p$, где v – любое число, для которого $\gcd(v, p-1) = 1$, и число $t = -rv^{-1} \bmod (p-1)$ дадут, как легко проверить, правильную ЦП.

Наконец полезно отметить, что если не выполнен шаг 2 алгоритма верификации ЦП, то злоумышленник может правильно подписать любое сообщение по своему выбору при условии, что в его распоряжении имеется какое-либо другое сообщение с правильной ЦП (см. доказательство в [3]).

Таким образом, при выборе модуля p, который в двоичном представлении имеет длину порядка 768 бит, обеспечивается хорошая стойкость ЦП, а для обеспечения долговременной стойкости целесообразно увеличить ее до 1024 бит.

Заметим, что возможны модификации алгоритма ЦП Эль-Гамаля, которые обеспечивают помимо выполнения ЦП также и выполнение функции конфиденциальности сообщений [24].

Обобщение ЦП на случай эллиптических кривых

Такое обобщение легко достигается при использовании ЦП на основе КС Эль-Гамаля. Это обобщение делается совершенно аналогично тому, как производится шифрование над эллиптическими кривыми. Сообщение представляется точками на эллиптических кривых, и все операции возведения в степень (например, $a^a \bmod p$) заменяются на операции умножения числа a на точку α эллиптической кривой E достаточно высокого порядка, т. е. на операцию $(a \cdot \alpha) = \alpha + \alpha + \ldots + \alpha$.

Преимущество выполнения ЦП на эллиптических кривых по сравнению с модульными операциями, также как и в случае шифрования, состоит в том, что может быть выбрана значительно меньшая длина открытого ключа для обеспечения такой же стойкости ЦП при атаке на нее дискретным логарифмированием.

Другие системы ЦП

DSA (федеральный стандарт США). Является обобщением ЦП Эль-Гамаля с тем основным отличием, что для генерирования, формирования и проверки ЦП используются модульные операции по двум различным большим простыми числам p и q [3].

Стандарт ЦП ГОСТ Р 3410–94 (российский стандарт) . Имеется также модификация этого ГОСТа для случая эллиптических кривых [25].

Все рассмотренные ранее варианты выполнения ЦП имеют три основных недостатка:

1) значительное увеличение объема (длины) подписанного сообщения по сравнению с объемом (длиной) самого сообщения, поскольку длина ЦП оказывается больше или равной длине сообщения;

2) вычисление ЦП к каждому последовательному сообщению M_i, $i = 1, 2, \ldots, L$, приводит к недопустимо большому расходу времени при выполнении ЦП для значительных величин L;

3) как было отмечено при описании ЦП на основе КС Эль-Гамаля, для сообщений, не содержащих избыточности (например, данных или криптограмм), возможна подделка ЦП.

Для устранения этих недостатков используются так называемые *бесключевые хеш-функции*. В этом случае подписи подлежит не само сообщение, а хеш-функция сообщения, т. е. обычно короткая цепочка длины порядка 128–512 бит. Ясно, что при этом первый и второй недостатки, приведенные выше, устраняются, поскольку длина ЦП оказывается равной длине хеш-функции. Что же касается третьего недостатка, то он также устраняется при использовании однонаправленных хеш-функций, определение и способы реализации которых будут описаны далее.

3.3.3. Бесключевые хеш-функции

Как отмечалось выше, подписываются обычно не сами сообщения, а результат их преобразования хеш-функциями (ХФ), который будем кратко называть далее просто *хешем.* В первой части данной книги уже рассматривались *ключевые хеш-функции*. Однако при выполнении ЦП не имеет смысла их использовать, поскольку проверка ЦП не должна, как правило, требовать знания секретного ключа. Кроме того, ключевые ХФ строятся обычно на основе симметричных систем шифрования, тогда как ЦП основана на несимметричных КС. Таким образом, вид ХФ, т. е. алгоритм вычисления хеша $h = h(x)$, должен быть полностью известным. (Далее, не умаляя общности, будем полагать, что как аргументы ХФ x, так и значения хешей h являются двоичными цепочками длин n и m соответственно.)

Вместе с тем выполнение подписей под ХФ (а не непосредственно под самими сообщениями) приводит к появлению новой атаки на ЦП. Действительно, если удастся подменить истинное сообщение на другое, но такое, которое имеет тот же самый хеш, то проверка ЦП под этим фальсифицированным сообщением покажет ее правильность и, следовательно, такая подмена не будет обнаружена.

Поэтому ХФ, используемые для выполнения ЦП (называемые *криптографическими хеш-функциями*), должны удовлетворять особым требованиям.

Основные требования, предъявляемые к криптографическим ХФ

Однонаправленность. При известном хеше h вычислительно неосуществимо (т. е. требует нереализуемо большого числа операций) нахождение хотя бы одного значения x, для которого $h(x) = h$, т. е. $h(x)$ оказывается *однонаправленной функцией (ОНФ*).

Слабая коллизионная стойкость. Для заданных $x, h(x) = h$ вычислительно неосуществимо найти такое другое значение x', которое удовлетворяет уравнению $h(x') = h$.

Сильная коллизионная стойкость. Вычислительно неосуществимо найти такую пару аргуметнов $x,\ x'$, для которых выполняется соотношение $h(x) = h(x')$.

Заметим, что в качестве криптографической ХФ нельзя, вообще говоря, использовать функции из класса универсальных. Действительно, если выбрать ХФ из такого класса, как $h = [x \times g]_b$, где $x, g, \in GF(2^n)$, $\times$ – умножение в этом поле и $[\cdot]_b$ означает укорочение цепочки длиной n до цепочки длиной b, то однонаправленность не будет выполнена, поскольку по известным $h,\ g,\ b$ легко находится $x = g^{-1} \times (h, \tilde{h})$, где g^{-1} означает обратный элемент к $g \in GF(2^n)$, а $(h,\ \tilde{h})$ – это объединение цепочки хеша h с произвольной двоичной цепочкой $\tilde{h}$ длины $n - b$.

Хорошая криптографическая ХФ обладает тем свойством, что при любом случайном выборе аргумента x вероятность преобразования его ХФ в фиксированный

хеш $h = h(x)$ будет близка к величине 2^{-m}, где m – длина двоичной цепочки хеша. Тогда при случайном выборе L различных аргументов ХФ x_1, x_2, …, x_L вероятность того, что хотя бы для одного их них хеш совпадет с заранее заданным значением h, будет равна $1 - (1 - 2^{-m})^L \cong L2^{-m}$. Таким образом, число попыток L, необходимых для обращения ХФ, будет с вероятностью P равно $P2^m$.

В частном случае вероятности успеха $P = 1/2$ число таких попыток оказывается равным 2^{m-1}. Отсюда, казалось бы, следует, что даже при выборе длины хеша, равной 64, такая атака оказывается нереальной, не говоря уже о том, что даже при успешном подборе аргумента, дающего заданное значение хеша, это еще не означает, что полученный аргумент будет соответствовать сообщению, которое хочет фальсифицировать злоумышленник.

Оказывается, что нарушение свойства *строгой коллизионной устойчивости* является более простой задачей, и это требует значительного увеличения длины хеша для обеспечения высокой стойкости ЦП.

Атака в целях нарушения этого свойства основана на так называемом *парадоксе дней рождения*, суть которого состоит в следующем. Пусть имеется урна, содержащая N шаров, пронумерованных от 1 до N. Их этой урны случайно извлекается M шаров с замещением (т. е. с возвратом извлеченного шара обратно в урну). Номера извлеченных шаров регистрируются в списке. Тогда вероятность $P(N,\ M)$ того, что в этом списке хотя бы один номер шара встретится не менее двух раз, может быть оценена асимптотически (т. е. при больших величинах N) по следующей формуле:

$$P(N, M) = 1 - \exp(-M^2/2N). \qquad (3.16)$$

Эта задача называется парадоксом дней рождения [26] потому, что если выбрать число N равным числу дней в невисокосном году, т. е. 365, а число людей M в случайно собранной группе равным 23, то вероятность того, что по крайней мере два человека из этой группы родились в один и тот же день года при точном расчете, а не по асимптотической формуле (3.16), оказывается удивительно большой – 0,507.

Атака на ХФ, основанная на парадоксе дней рождения, состоит в следующем. Пусть имеется истинное (легальное) сообщение x и сообщение x', которое хочет создать злоумышленник таким образом, чтобы оно было подтверждено подписью легального лица. Тогда злоумышленник генерирует список Z, состоящий из $2^{m/2}$ небольших модификаций сообщения x, благоприятных для легального владельца ЦП. Далее он генерирует поочередно сообщения x', любое из которых подходит для фальсификации. Как только появляется такое $\tilde{x}'$, что $h(\tilde{x}') = h(\tilde{x})$ для любого $\tilde{x} \in Z$, процедура генерирования x' заканчивается. Затем злоумышленник просит владельца ЦП подписать сообщение $\tilde{x}$ и подменяет его сообщением $\tilde{x}'$. Теперь он может быть

уверен, что скопированная им подпись к сообщению $\tilde{x}$ будет верна и для сообщения $\tilde{x}'$. Среднее число попыток генерирования сообщения x' будет $2^{m/2}$ с вероятностью порядка 0,4, как это следует из формулы (3.16), если в ней положить $N = 2^m, M = 2^{m/2}$.

Способы построения стойких криптографических бесключевых ХФ

Как было отмечено в предыдущем пункте, ХФ должны удовлетворять трем основным условиям, что достигается выбором алгоритма формирования хешей и соответствующим выбором параметров этого алгоритма. Известно три основных способа построения ХФ на основе использования:

блоковых шифров;

специально разработанных алгоритмов хеширования;

КС ОК.

Возможны и комбинации данных методов.

Далее будем рассматривать только первый подход, поскольку третий представляет пока лишь теоретический интерес, а второй, хотя и широко используется на практике (стандарты MD-4, MD-5, SHA-1 [3], ГОСТ Р 34.10–94 [25]), имеет весьма громоздкое описание, которое выходит за рамки данного курса. Заметим только, что при реализации последнего алгоритма используются обычно различные методы сжатия данных.

Использование первого подхода представляется вполне естественным, поскольку стойкие блочные шифры не позволяют найти вход (сообщение) по выходу (криптограмме). Однако нам нужно построить бесключевую ХФ! Если попытаться построить ее как итерацию блокового алгоритма шифрования, т. е. когда $h = h_t;\ h_i = f_{h_{i-1}}(M_i),\ i = 1, 2, \ldots, t;\ \ h_0 = IV$, где M_i $(i = 1, 2, \ldots, t)$ – последовательные блоки сообщения, $f_x(y)$ – алгоритм блочного шифрования сообщения y на ключе x, IV – известное начальное значение, то легко видеть, что такая ХФ не будет ОНФ. В связи с этим при хешировании на основе блоковых шифров, как увидим далее, обычно используют следующую модификацию предыдущего метода:

$$h_i = f_{h_{i-1}}(M_i) \oplus M_i, \tag{3.17}$$

где $\oplus$ – это операция побитного сложения по $\mathrm{mod}\ 2$.

Если блоковый шифр является вычислительно стойким, то ХФ, построенная по алгоритму (3.17), будет ОНФ.

Другая модификация блокового алгоритма шифрования, называемая обычно схемой Рабина, формирует хеши по следующему правилу:

$$h_i = f_{M_i}(h_{i-1}),\ h_0 = IV,\ i = 1, 2, \ldots, t. \tag{3.18}$$

Легко видеть, что даже при известном IV алгоритм (3.18) удовлетворяет свойству ОНФ, однако специальная атака, называемая *встречей посредине*, позволяет

найти «коллизию» даже для наперед заданного сообщения [24]. Впрочем, число необходимых операций для выполнения такой атаки составляет около $2^{m/2}$, где m – длина хеша, и поэтому при выборе $m \geq 128$ она оказывается практически нереализуемой.

При реализации ХФ на основе блоковых шифров важно различать методы с длиной хеша, равной длине шифруемого блока, и длинами хешей, кратными длине блока шифрования. Последнее важно, когда используются алгоритмы шифрования с относительно короткой длиной блока, например как у шифра DES, где эта длина равна 64 бита. Действительно, как отмечалось ранее, атака, основанная на парадоксе дней рождения, делает такую ХФ нестойкой, но при удвоении длины хеша атака становится вычислительно нереализуемой. Рассмотрим далее несколько примеров алгоритмов хеширования для ХФ с одиночной и двойной длиной хешей. Другие примеры подобного рода ХФ можно найти в [3].

Примеры ХФ с одиночной длиной блоков-выходов (хешей) [3]

Схема Матиаса–Мейера–Оceaca (Matyas–Meyer–Oseas):

$$h = h_t,\ h_0 = IV, h_i = f_{g(h_{i-1})}(M_i) \oplus M_i,\ 1 \leq i \leq t,$$

$g(\cdot)$ – функция, которая преобразует n-битовый блок в ключевой блок x длины k, подходящей для алгоритма шифрования $f_x(y)$;

$M = (M_1,\ M_2,\ \ldots,\ M_i,\ \ldots,\ M_t)$, M_i – n-битовые блоки.

Схема Дэвиса–Межера (Davies–Meger):

$$h = h_t,\ h_0 = IV,\ h_i = f_{M_i}(h_{i-1}) \oplus h_{i-1},\ 1 \leq i \leq t,$$

где IV – известный начальный блок длины n; $M = (M_1,\ M_2,\ \ldots,\ M_i,\ \ldots,\ M_t)$, M_i – k-битовые блоки (k – длина ключа для выбранного алгоритма шифрования).

Схема Миагучи–Пренила (Miyaguchi–Preneel):

$$h = h_t,\ h_0 = IV,\ h_i = f_{g(h_{i-1})}(M_i) \oplus h_{i-1},\ 1 \leq i \leq t,$$

где $M = (M_1,\ M_2,\ \ldots,\ M_i,\ \ldots,\ M_t)$, M_i – n-битовые блоки.

Пример ХФ MDC-2 с двойной длиной выхода (хеша) (Алгоритм на основе шифра DES) [3]

Пусть сообщение $M = (M_1,\ M_2,\ \ldots,\ M_i,\ \ldots,\ M_t)$, M_i – 64-битовые блоки, а знак $||$ обозначает конкатенацию, C_i^L, C_i^R – левая и правая 32-битовые половины блока C_i; IV и $\tilde{IV}$ – 64-битовые несекретные постоянные, которые выбраны (в 16-ричном представлении) следующим образом:

$$IV = 0 \times 5252525252525252;\ \tilde{IV} = 0 \times 2525252525252525.$$

Далее определим g and $\tilde{g}$ как две функции, которые преобразуют 64-битовые блоки u в 56-битовые блоки, подходящие для ключей DES, следующим образом:

$$g(u) = u_1\ 10\ u_4u_5u_6u_7u_9u_{10}\ \ldots\ u_{63};$$
$$\tilde{g}(u) = u_1\ 01\ u_4u_5u_6u_7u_9u_{10}\ \ldots\ u_{63};$$
where $u = (u_1, u_2, \ldots, u_{64})$.

Тогда 128-битовые хеши h формируются следующим образом:

$$h_0 = IV,\ K_i = g(h_{i-1}),\ C_i = E_{K_i}(M_i) \oplus M_i,\ h_i = C_i^L || \tilde{C}_i^R;$$
$$h_0 = \tilde{IV},\ \tilde{K}_i = \tilde{g}\left(\tilde{h}_{i-1}\right),\ \tilde{C}_i = E_{\tilde{K}_i}(M_i) \oplus M_i,\ \tilde{h}_i = \tilde{C}_i^L || C_i^R,\ i = 1, 2, \ldots, t,$$
$$h = h(M) = h_t || \tilde{h}_t.$$

3.3.4. Выводы о возможности построения стойких цифровых подписей

Материал, изложенный в предыдущих разделах, позволяет сделать следующие важные выводы о возможности аутентификации сообщений на основе ЦП.

1. ЦП является эффективным средством обеспечения подлинности и целостности всех видов сообщений, которые могут быть представлены в цифровой форме.

2. Стойкость ЦП (т. е. ее устойчивость к необнаруженной подделке сообщений) может быть обеспечена при помощи выбора необходимого несимметричного алгоритма шифрования и ХФ, а также соответствующих параметров этих алгоритмов, причем в практически используемых системах под стойкостью понимается вычислительная стойкость, определяемая числом необходимых операций или (и) объемом памяти, которые необходимы для подделки ЦП.

3. ЦП обладает всеми основными свойствами обычной бумажной подписи, и поэтому она может рассматриваться наравне с последней и при судебном разбирательстве. Однако, в отличие от бумажной подписи, передача секретного ключа ЦП другим лицам делает их фактически уполномоченными выполнять эту подпись.

4. ЦП требует определенного объема памяти для ее хранения. Однако при больших объемах сообщения это относительное увеличение объема оказывается незначительным.

5. Время создания и верификации ЦП (даже при обеспечении долговременной стойкости) оказывается сравнительно небольшим.

6. Длины ключей для создания и верификации ЦП имеют порядок 768–1024 бит. В некоторых приложениях, где к длинам ключей предъявляются жесткие требования, используются специальные алгоритмы ЦП, например на основе эллиптических кривых [22].

7. Подделка открытых ключей злоумышленниками может привести к ложной верификации ЦП. Для защиты от такой угрозы можно использовать так называемые *центры сертификации ключей*, где их подлинность гарантируется доверием к этим центрам со стороны легальных пользователей.

8. Существуют различные разновидности ЦП, которые будут рассматриваться далее в разделе, посвященном криптографическим протоколам.

3.4. Криптографические протоколы

В настоящее время многие коммерческие сделки и деловые операции выполняются через *Интернет*, причем для их защиты от действий злоумышленников оказывается вполне достаточным использование таких изученных ранее криптографических функций, как *конфиденциальность* и *аутентификация*, реализуемых при помощи криптосистем шифрования/дешифрования и цифровой подписи.

Однако существует множество других взаимодействий между двумя или более пользователями сети Интернет (называемых обычно протоколами), для которых обеспечение их безопасности не может быть реализовано только перечисленными выше средствами. Примерами подобных протоколов являются: электронные платежи, лотереи и аукционы, тайное голосование, анонимная покупка данных, выполнение совместных вычислений с сохранением индивидуальных данных в секрете и т. д.

Для обеспечения безопасности выполнения этих или иных действий, выполняемых дистанционно (через Интернет или в какой-либо другой локальной или корпоративной сети) в условиях присутствия в этих сетях недобросовестных пользователей (в том числе и среди законных участников этих процедур) используются специально для них разработанные так называемые *криптографические протоколы* (КП).

Проведем краткий обзор важнейших КП, а также рассмотрим достаточно подробно такие КП, как разделение секретных данных, идентификация на основе нулевых разглашений, поручительство информации и тайное голосование.

3.4.1. Обзор основных криптографических протоколов [27]

№ п/п	Название протокола	Задачи, решаемые после выполнения данного протокола
1	Разделение секретов	Метод, при котором организатор протокола вычисляет частные секреты («тени») k_i, $1 \le i \le n$, от исходного секрета k и секретным образом распределяет k_i пользователям так, чтобы выполнялось следующее условие: любые m или более пользователей, которые могут объединить свои тени k_i, легко восстанавливают k, но любая группа, состоящая из $m-1$ или меньшего числа пользователей, не может этого сделать. Знание $m-1$ или меньшего числа теней вообще не дает никакой информации об исходном секрете
2	Скрытный канал (фактически это стеганография, но используемая в криптосистемах)	При помощи обмена совершенно неподозрительными сведениями два участника хотят передать некоторую дополнительную информацию в присутствии наблюдателя, который не должен догадаться о самом факте ее присутствия в основном сообщении

3	Различные версии ЦП	
3.1	Неоспоримая ЦП	Подобно обычной ЦП, неоспоримая ЦП зависит от подписанного документа и секретного ключа автора ЦП. Однако в противоположность обычной ЦП неоспоримая ЦП не может быть верифицирована без участия ее автора
3.2	Останавливаемая ЦП (обеспечивает стойкость, не зависящую от вычислительной мощности нарушителя)	Если подписавший хочет отменить свою ЦП, подозревая, скажем, подделку, то надо убедить в этом суд. Основная идея останавливаемой ЦП состоит в том, что для каждого открытого ключа имеется множество секретных ключей подписи и невозможно узнать, какой именно из них был использован при ЦП данного документа в действительности
3.3	Групповая ЦП	Только члены определенной группы уполномочены подписывать некоторые сообщения. Каждый член этой группы может проверить правильность данной подписи, но для «рядовых» членов группы невозможно определить авторство подписи и только специально уполномоченный «супервизор» сможет это сделать
3.4	«Слепая» ЦП	Необходимо подписать сообщение, не зная его содержания, так чтобы каждый впоследствии смог убедиться в правильности этой подписи. Если небезопасно подписывать сообщения «вслепую», то должна существовать возможность получать определенные сведения о сообщениях, сохраняя при этом основные свойства слепой подписи
3.5	Одновр-ное подписание контрактов	Два участника хотят подписать контракт, но ни один из них не должен сделать это раньше другого
3.6	Заказная ЦП	Один участник хочет послать сообщение другому, но при условии, что тот сможет его прочитать только после того, как пошлет отправителю квитанцию о получении этого сообщения
4	Поручительство информации	Один из участников хочет передать бит (в более общем случае цепочку бит) на хранение другому участнику, но так чтобы последний смог его прочитать лишь позднее (по доп-ному распоряжению и при помощи посылки дополнительной информации от 1-го участника). Однако, с другой стороны, участник, сохраняющий бит, должен исключить возм-сть его изменения при поступлении дополнительных сведений со стороны 1-го участника
5	Доказательства с нулевым разглашением секретов	Доказывающий участник должен убедить проверяющего участника, что он обладает определенной информацией (например, секретным ключом или доказательством гипотезы Гольдбаха), но не выдать ему при этом абсолютно никаких сведений о самой этой информации,

		кроме, конечно, самого факта, что он эту информацию имеет. (Частным случаем этого протокола является протокол Фейге–Фиата–Шамира об идентификации пользователей, который подробно рассмотрен далее.)
6	Обманчивая передача	Некоторый пользователь обладает набором секретов (скажем, компрометирующих документов). Другой пользователь хочет купить один из них, но при условии, что продавец компроматов не узнает, какой именно документ у него покупают
7	Тайное голосование	Это компьютерный аналог обычного тайного голосования, выполняемого дистанционно при помощи протокола, который должен удовлетворять следующим требованиям: - в голосовании принимают участие только авторизованные избиратели; - никто не может проголосовать более одного раза; - никто (включая и избирательную комиссию) не может узнать результат голосования каждого из участников; - никто (включая и избирательную комисию) не может изменить результаты голосования без обнаружения этого факта; - все авторизованные избиратели могут убедиться, что их голос правильно учтен в итоге голосования; - общеизвестным становится список всех проголосовавших. (Возможны и другие наборы требований.)
8	Совместные секретные вычисления	Группа участников хочет выполнить вычисление определенной функции от своих секретных данных, но так чтобы эти индивидуальные секретные данные не стали известны другим участникам
9	Цифровая наличность (ЦН)	Использование кредитных карт не решает проблему анонимности платежей, поскольку позволяет отследить, на чей счет переводятся деньги. Таким свойством обладает обычная бумажная наличность. Ее цифровой аналог должен обладать следующими свойствами: а) ЦН может передаваться по компьютерным сетям; б) ЦН не может быть скопирована и использована повторно; в) никто не может отследить связи между покупателем и продавцом; г) магазин не нуждается в наличии связи с банком покупателя (*off-line* покупки); д) ЦН может быть передана (по желанию ее хозяина) другим пользователям; е) ЦН может быть разменена на более мелкие части, дающие в сумме ту же величину ЦН

10	Ограничение расстояния	Необходимо криптографически удостоверить расстояние между участниками протокола, выполняемого по компьютерным сетям. Существование данного протокола решает так называемую проблему «*мафиозной атаки»* (известной также под названием «*гроссмейстерской проблемы»* [27])
11	Анонимное широковещание	Группа пользователей некоторой сети передает по ней широковещательные сообщения, но предназначенные определенным пользователям, только которые и могут дешифровать эти сообщения. Условие этого протокола заключается в том, что никто, кроме отправителя и легального получателя сообщений, не может определить этих пользователей, в том числе и при помощи анализа телетрафика
12	Отслеживание предателей	Платные ТВ-программы могут быть зашифрованы при помощи ключей, которые передаются легальным подписчикам этих программ.
		Однако существует опасность, что некоторые из таких нечестных подписчиков (предателей) могут продать эти ключи другим пользователям, не оплатившим эти программы. Протокол обеспечивает возможность поиска таких предателей в случае, когда ТВ-декодер пользователей может быть открыт для необходимого контроля. Заметим, что такая же задача (но с использованием другой техники) решается частным видом стеганографии (fingerprinting)
13	Честные криптосистемы	Необходимо обеспечить конфиденциальность сообщений, однако по постановлению суда эти сообщения должны просто расшифровываться

3.4.2. Описание процедур выполнения некоторых криптографических протоколов

КП «разделение секретов»

В этом случае в качестве секретных данных могут рассматриваться, например, команды по применению ядерного оружия, либо ключи к зашифрованным файлам, содержащим важную информацию по рецептуре продуктов различного рода «ноу хау» промышленных процессов и т. п. Разделение здесь необходимо для исключения возможности несанкционированного использования этих данных одной персоной. Разделение данных предполагает, что для выделения основного секрета необходимо объединение частных секретов (*теней*) нескольких лиц, причем в количестве не менее некоторой заданной величины. Очевидно, что тайный сговор достаточно большой группы лиц, владеющих частными секретами для получения основного

секрета, значительно менее вероятен, чем появление одного такого предателя. Это и обусловливает необходимость создания протокола разделения секретов. Возможно использование простейшей схемы разделения секретов – «*Все или никто»*, в которой только объединение частных секретов *всех* их обладателей позволит им вычислить основной секрет. Такая схема просто реализуется при помощи чисто случайного генерирования двоичных цепочек секретов k_i, $i = 1, 2, \ldots, n-1$, всех пользователей, кроме одного, тогда как последний получает частный секрет k_n следующего вида:

$$k_n = \bigoplus \sum_{i=1}^{n-1} k_i \bigoplus k,$$

где k – основной секрет, а $\bigoplus$ означает побитовое суммирование по mod 2. Однако в этом случае отсутствие (или потеря) даже одного из частных секретов приведет к невозможности восстановления основного секрета. Поэтому и возникает необходимость создания пороговой схемы, где для восстановления основного секрета достаточно объединить только m из n частных секретов.

Определение 3.5. *Пороговой (n, m)-схемой* называется такой алгоритм формирования частных секретов k_i, $i = 1, 2, \ldots, n$ (теней), по основному секрету k, что при объединении m или более таких теней существует алгоритм, позволяющий восстановить в точности основной секрет, тогда как объединение менее чем m теней не дает абсолютно никакой информации об основном секрете k (рис. 3.9).

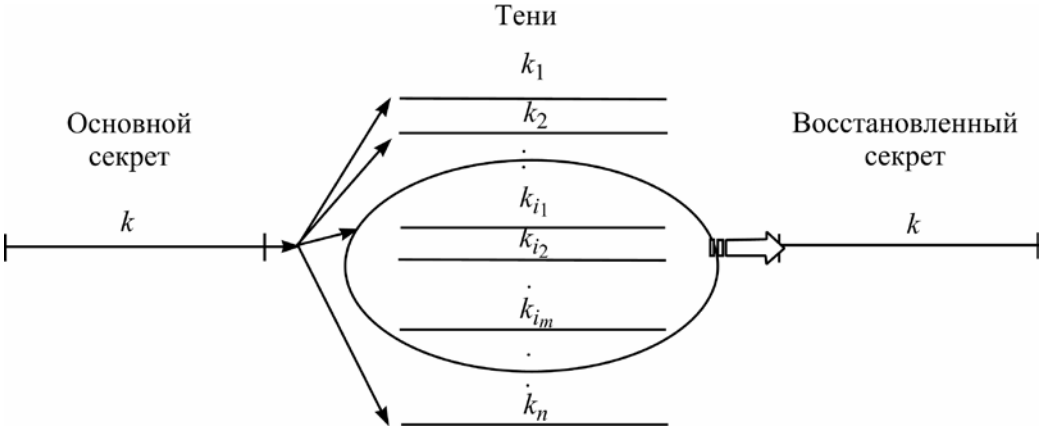


Рис. 3.9. Пороговая (n, m)-схема

Существует множество различных способов построения пороговых схем. Рассмотрим далее описание одной из них, основанной на *интерполяции полиномов над конечными полями*.

Пусть основной секрет $k \in GF(p)$, где $GF(p)$ – конечное простое поле, что всегда возможно для любого k при выборе необходимой величины p.

Выберем коэффициент a_0 полинома степени $m-1$ при его постоянном члене равным основному секрету k, а остальные его коэффициенты $a_1, \ldots, a_{m-1} \in GF(p)$ выберем чисто случайными. Тогда полином $(m-1)$-й степени однозначно определяется всеми этими коэффициентами как

$$h(x) = a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \cdots + a_1x + a_0.$$

Частные ключи (тени) вычисляются по формуле $k_i = h(x_i),\ i = 1,\ 2,\ \ldots,\ n,$ $x_i \in GF(p)$, где, в частности, можно взять $x_i = i, n < p$.

Затем k_i передаются каждому из n пользователей по секретным каналам.

Если m (или более) пользователей $i_1,\ i_2,\ \ldots,\ i_m$ объединят свои индивидуальные ключи $k_{i_1},\ k_{i_2},\ \ldots,\ k_{i_m}$, то они смогут восстановить полином $h(x)$, используя интерполяционную формулу Лагранжа [27]:

$$h(x) = \sum_{S=1}^{m} k_{i_S} \prod_{\substack{j=1 \\ S \neq j}}^{m} \frac{\left(x - x_{i_j}\right)}{\left(i_{i_S} - x_{i_j}\right)} \bmod P.$$

Тогда основной ключ легко может быть найден как

$$a_0 = h(0) = \sum_{S=1}^{m} k_S\, C_S \bmod P, \text{ где } C_S = \prod_{\substack{S'=1 \\ S \neq S'}}^{m} \frac{x_{j_{S'}}}{x_{j_S} - x_{j_{S'}}}.$$

Если же число теней оказывается менее m, то они не будут содержать никакой информации о коэффициенте a_0, поскольку для любого значения $a_0 \in GF(p)$всегда найдется полином $(m-1)$-й степени $\tilde{h}(x)$, удовлетворяющий уравнениям $\tilde{h}(x_i) = k_i,$ $i = 1,\ 2,\ \ldots,\ m-1$.

Пример 3.4. Пусть требуется создать пороговую схему (5, 3) предназначенную для пяти пользователей, в которой для восстановления основного секрета требуется три или более теней.

Тени получаются при помощи вычисления многочлена в *пяти* различных точках. В частном случае первой тенью может быть значение многочлена при $x = 1$, второй тенью – значение многочлена при $x = 2$ и т. д.

Пусть k равно 13. Чтобы создать пороговую (5, 3) - схему, в которой любые три из пяти пользователей смогут восстановить k, сначала выберем простое число $p = 17$ (оно больше количества теней и больше основного секрета в данном примере). Предположим, что числа 2 и 10 оказались случайно выбранными коэффициентами полинома, который в этом случае будет равен

$$h(x) = 2x^2 + 10x + 13.$$

Пятью тенями оказываются тогда следующие числа:

$$k_1 = h(1) = (2 + 10 + 13) \bmod 17 = 8;$$
$$k_2 = h(2) = (8 + 20 + 13) \bmod 17 = 7;$$
$$k_3 = h(3) = (18 + 30 + 13) \bmod 17 = 10;$$
$$k_4 = h(4) = (32 + 40 + 13) \bmod 17 = 0;$$
$$k_5 = h(5) = (50 + 50 + 13) \bmod 17 = 11.$$

Эти тени распределяются среди пяти участников протокола. Допустим, 1-й, 3-й и 5-й из них, собрав свои тени, хотят восстановить основной секрет. Используя интерполяционную формулу Лагранжа, они находят

$$h(x) = \left[\frac{8\cdot(x-3)\cdot(x-5)}{(1-3)\cdot(1-5)} + \frac{10\cdot(x-1)\cdot(x-5)}{(3-1)\cdot(3-5)} + \frac{11\cdot(x-1)\cdot(x-3)}{(5-1)\cdot(5-3)}\right] \bmod 17.$$

Произведя все вычисления по модулю 17, получаем $h(x) = 2x^2 + 10x + 13$. Свободный член в полученном полиноме 13 и есть восстановленный основной секрет.

Основное преимущество пороговой схемы на основе интерполяционных полиномов Лагранжа состоит в том, что при появлении новых пользователей не надо менять основной секрет, и если он является ключом, на котором зашифрованы некоторые данные, то их не надо перешифровывать, а достаточно лишь вычислить для основного ключа дополнительные индивидуальные секреты.

Имеется множество различных обобщений [27] для пороговых схем разделения секретов, например:

- взвешенные секреты распределения порогов по группам пользователей;
- схемы с обнаружением фальсификации отдельными пользователями своих частных данных.

Еще одним интересным обобщением схемы с разделением секретов является так называемая $(n,\ m,\ m_0,)$ *рамп-схема*, в которой объединение m и более пользователей однозначно восстанавливает секрет, при объединении теней s пользователей в интервале $m_0 \leq s < m$ основной секрет восстанавливается лишь частично, а при числе объединенных пользователей, меньшем, чем m_0, восстановить его оказывается невозможно.

Доказательства с нулевым разглашением

При помощи протоколов из данного семейства один из пользователей компьютерной сети может доказать другому пользователю, что у него имеется некоторая информация, не раскрывая самой информации.

Например, один из участников доказал гипотезу Гольдбаха о том, что каждое четное целое число больше двух можно представить в виде суммы двух простых чисел, но опасаясь плагиата, он не хочет выдать проверяющему технические детали доказательства, тем не менее хочет убедить его, что доказательство выполнено им математически корректно.

Доказательства с нулевым разглашением обычно принимают форму интерактивных протоколов. *Проверяющий* (V) задает *доказывающему* (P) ряд вопросов. Если P знает секрет, то он ответит на все вопросы V правильно. Если же секрет ему не известен, то у него есть лишь некоторая вероятность (обычно 0,5) ответить правильно. Тогда после значительного количества вопросов V сможет достоверно убедиться, знает ли P секрет. Однако ни один из заданных V вопросов и ответов P на них не должен дать V ни малейших сведений об информации, которой обладает P.

Предположим, что P известна некоторая информация, которая является решением трудной проблемы. Базовый протокол нулевого разглашения состоит обычно из нескольких раундов следующего типа:

1. P использует свою информацию и случайное число для преобразования основной трудной проблемы в другую, изоморфную ей проблему. Затем он использует свою информацию и известное ему случайное число для решения новой трудной проблемы.

2. P передает V решение новой проблемы, используя протокол 4 *поручительства информации*.

3. P объясняет V сущность новой трудной проблемы, однако V не может использовать это знание для получения какой-либо информации о первоначальной проблеме или путях ее решения.

4. V просит P одно из двух:

а) доказать ему, что новая и старая проблемы изоморфны,

б) открыть решение, полученное V на этапе шага 2, и доказать, что это действительно решение новой проблемы.

5. P исполняет просьбу V.

6. P и V повторяют *n* раз шаги 1–5.

Математическое доказательство того факта, что протоколы подобного типа действительно обеспечивают отсутствие утечки к V какой-либо информации о решении основной проблемы и, кроме того, дают надежную гарантию для V, что P имеет такое решение, весьма сложно. Сами проблемы и случайное изоморфное преобразование должны выбираться осторожно, чтобы V не получил никакой информации о решении основной проблемы даже после многих повторений шагов протокола. Заметим, что далеко не все трудные проблемы можно использовать для доказательств с нулевым разглашением, но многие из них допускают это. Рассмотрим в качестве иллюстрирующего примера (который сам по себе не имеет важного практического применения) использование в качестве такой трудной задачи нахождение так называемого *гамильтонового цикла* на заданном графе.

Напомним сначала, что гамильтоновым циклом (ГЦ) называется замкнутая непрерывная линия на графе, которая проходит по ребрам графа через все его вершины только один раз, за исключением исходной вершины. Не каждый граф содержит ГЦ, и до сих пор не известно общих полиномиально сложных методов установления этого факта, а тем более нахождения самого этого цикла, даже если он существует.

Если два графа идентичны во всем, кроме наименования точек, то они называются *топологически изоморфными*. Для графов очень больших размеров доказательство их изоморфности может потребовать много компьютерного времени. Это одна из так называемых *NP-трудных* проблем в теории сложности решений [21].

Предположим, что некоторый граф G известен как P, так и V, пусть также P удалось каким-то образом найти на нем ГЦ и он хочет доказать этот факт V, не выдавая ГЦ. Тогда рассмотренный выше протокол доказательства с нулевым разглашением принимает для данной задачи следующий вид:

1. P случайным образом переставляет нумерацию вершин графа G, получая другой граф H, который будет, очевидно, изоморфен G. Так как P знает этот изоморфизм и ему известен ГЦ на графе G, он легко находит такой же ГЦ на графе H. С другой стороны, если бы P не знал ГЦ на графе G, то он не смог бы в обозримое время найти его и на графе H. Более того, если бы P знал ГЦ на каком-то другом графе $\tilde{G}$, то он не смог бы доказать в обозримое время, что $\tilde{G}$ и G топологически изоморфны (даже если бы это было верно), поскольку доказательство этого факта для произвольных графов является, как уже отмечалось ранее, трудной задачей.

2. P посылает V копию графа H.

3. V просит P выполнить одно из двух:

   а) доказать, что H и G изоморфны,

   б) показать ГЦ на H.

4. P исполняет его просьбу, делает одно из двух:

   а) доказывает, что H и G изоморфны, не показывая ГЦ на G или H,

   б) показывает ГЦ только на H, не доказывая изоморфизма G и H.

5. P и V повторяют n раз шаги протокола 1–4.

Если P знает ГЦ на графе G, то он сможет правильно выполнить задания V на каждой из n итераций. Если P этого не знает, то он не сможет выполнить требования V на всех шагах. Лучшее, что сможет сделать нечестный P, это построить такой граф $\widetilde{H}$, который будет или изоморфным G, или имеющим известный ему ГЦ. Очевидно, что у P оказывается только 50% шансов угадать, какое доказательство потребует от него V на шаге 3. Таким образом, задавая достаточно большое число итераций n, можно надежно обеспечить разоблачение нечестного P.

С другой стороны, этот протокол не дает V никакой информации, помогающей ему из ответов P установить ГЦ графа G, поскольку P для каждого нового раунда протокола генерирует новый граф H.

Видно, что протокол, рассмотренный выше, требует обмена информацией между P и V, поэтому такого типа протоколы называются *интерактивными.* Достаточно неожиданным является тот факт, что протоколы с нулевым разглашением не обязательно должны быть интерактивными. Это означает, что P публикует все необходимые данные заранее и каждый пользователь имеет возможность проверить, что P обладает некоторыми секретными знаниями, не получив из этих знаний никакой информации и не вступая даже в диалог с P! Описание *неинтерактивного* протокола для доказательства ГЦ заданного графа можно найти в [28].

Идентификация пользователей при помощи протокола с нулевым разглашением

Широкое распространение интеллектуальных карт для разнообразных коммерческих, гражданских и военных применений потребовало обеспечения безопасности идентификации таких карт и их владельцев. Во многих приложениях главная проблема заключается в том, чтобы при предъявлении интеллектуальной карты оперативно обнаружить обман и отказать нарушителю в допуске, ответе или обслуживании.

Схемы идентификации на основе паролей слабо соответствуют требованиям указанных приложений. Один из существенных недостатков такой идентификации заключается в том, что после того, как доказывающий передаст проверяющему пользователю свой пароль, проверяющий может, используя данный пароль, выдать себя впоследствии за проверяемого пользователя.

Протоколы идентификации на основе симметричных алгоритмов шифрования также имеют свои недостатки. Для работы таких протоколов идентификации необходимо, чтобы проверяющий и доказывающий с самого начала имели один и тот же секретный ключ. Следовательно, встает вопрос о распределении и доставке секретных ключей. При исполнении таких протоколов пользователь доказывает знание секретного ключа, производя с его помощью расшифрование запросов. Проверяющий пользователь имеет принципиальную возможность так сформировать запросы, чтобы передаваемые ответы могли быть обработаны в целях извлечения из них дополнительной информации о секретном ключе с последующим возможным раскрытием этого ключа [28].

Широко известны способы идентификации на основе использования цифровой подписи. Преимущество идентификации на основе использования доказательств с нулевыми знаниями над остальными способами идентификации (в частности, и на основе ЦП) заключается в том, что в ходе ее выполнения никакой информации о секретном ключе не «утекает» к проверяющему и ко всем посторонним наблюдателям, в то время как, например, в случае идентификации на основе ЦП в ней присутствует информация о секретном ключе подписи, которую, хотя и вычислительно сложно, но принципиально можно найти. Кроме того, алгоритмы выполнения и проверки ЦП содержат в себе сложные операции модульного возведения в степень больших чисел, требующие при их программной реализации больших ресурсов процессорного времени, тогда как в алгоритме идентификации с нулевым разглашением (НР) применяются гораздо более простые модульные математические операции (возведение в квадрат и умножение), что позволяет значительно снизить требования к вычислительным ресурсам верификации.

Концепцию HP может использовать проверяющий для доказательства того, что некоторый пользователь обладает определенным секретным ключом, однозначно его идентифицирующим, и при этом избежать утечки какой-либо информации об этом ключе. Рассмотрим далее пример построения подобного протокола [28].

Пусть секретным ключом пользователя будет цепочка чисел $\bar{C} = (C_1, C_2, \ldots, C_k)$, где $1 \leq C_j \leq n$, а $n = p \cdot q$ – модуль, p и q большие простые числа, причем координаты $\bar{C}$ удовлетворяют уравнениям

$$d_j \cdot C_j^2 = (\pm 1) \bmod n,\ i = 1,\ 2,\ 3,\ \ldots,\ k, \tag{3.19}$$

где $d_1, d_2, \ldots, d_k$ – открытый идентификатор пользователя.

Некоторый центр предварительно формирует число $n = p \cdot q$, где p и q – большие простые числа, причем $p = 3 \bmod 4$ и $q = 3 \bmod 4$; далее необходимость в обращении к центру для выполнения протокола идентификации отпадает.

Проверяющий (V) знает число n и то, что секретный идентификатор определенного пользователя $\bar{C}$ удовлетворяет уравнению (3.19), но сам этот идентификатор для него неизвестен. При выполнении протокола идентификации проверяющий V по предъявленным ему данным должен убедиться, что их автор имеет в своем распоряжении $\bar{C}$, но не может получить больше об этом векторе никакой *дополнительной информации*. Однако вычислительная мощность V должна быть ограничена, поскольку в противном случае V сможет найти $\bar{C}$ еще до выполнения протокола и, следовательно, выдавать себя в дальнейшем за пользователя P.

Протокол HP состоит из четырех шагов:

1. Проверяемый пользователь (P) генерирует случайное число r и находит число

$$x = r^2 \bmod n. \tag{3.20}$$

Далее он посылает это число V.

2. V генерирует случайное подмножество $S \subseteq (1,\ 2,\ \ldots,\ k)$ и отправляет его P.

3. P вычисляет

$$T_c = \prod_{j \in S} C_j \bmod n \tag{3.21}$$

и посылает V число

$$y = r \cdot T_c \bmod n. \tag{3.22}$$

4. V вычисляет

$$X = y^2 \cdot T_d \bmod n, \tag{3.23}$$

где

$$T_d = \prod_{j \in S} d_j \bmod n, \tag{3.24}$$

и проверяет равенство

$$x = \pm X \bmod n. \tag{3.25}$$

Если равенство (3.25) выполняется, то производится следующая итерация (т. е. повторение протокола с новыми случайными данными). Если равенство (3.25) не выполняется, то проверяемый пользователь P не идентифицируется V.

Если справедливо равенство (3.25) (т. е. P является действительно обладателем секретного вектора $\bar{C}$), то уравнение (3.25) всегда выполняется. Действительно,

$$y^2 \cdot T_d = r^2 \cdot T_c^2 \cdot T_d = \pm r^2 = \pm x \bmod n. \qquad (3.26)$$

Умножение случайного числа r на T_c на шаге 3 действительно необходимо, поскольку иначе, выбирая на каждой итерации подмножество $S = \{j\}$, т. е. на первой итерации – $S = \{1\}$, на второй – $S = \{2\}$, и так далее до $S = \{k\}$, V сможет вычислить весь секретный идентификатор $C_1, C_2, \ldots, C_k$.

Стойкость данной системы идентификации базируется на невозможности извлечения квадратных корней по $\bmod n$ при неизвестной факторизации n. Никакой другой информации о векторе $\bar{C}$ проверяющий V в процессе выполнения протокола не получает. (Предполагается, что всегда выполняется условие $\gcd(C_j, n) = 1$, поскольку в противном случае модуль n может быть легко факторизован.) С другой стороны, существует только один способ для P обмануть V, т. е. обеспечить себе идентификацию при отсутствии у него чисел C_j. Этот способ заключается в угадывании подмножества S и формировании $x = \pm r^2 \cdot T_d \bmod n$ и $y = r$. Тогда равенство $x = y^2 \cdot T_d \bmod n$ будет всегда тривиально выполняться. Вероятность угадывания множества S со стороны P равна 2^{-k}, и тогда вероятность обмана при выполнении t итераций будет равна $2^{-k \cdot t}$. Соответственно чем больше число таких итераций, тем меньше вероятность обмана со стороны P при выполнении такого протокола.

Заметим также, что дополнительные условия на простые множители n нужны для того, чтобы V мог быть уверен, что числа C_j, соответствующие числам d_j, существуют.

Пример выполнения процедуры идентификации. Пусть центр предоставил пользователю модуль $n = 19 \cdot 31 = 589$. Предположим, что идентифицируемый пользователь сгенерировал секретный идентификатор $\bar{C} = (90, 544, 460, 263, 567)$. Открытый идентификатор $d(p)$ вычисляется как решение уравнения (5.1). Это уравнение решается путем нахождения обратных элементов к C_j^2 по модулю n, что дает, как легко проверить, вектор $\bar{d} = (472, 121, 253, 283, 359)$.

Выполним одну итерацию идентификации:

1. P генерирует случайное число $r = 859$, вычисляет $x = 859^2 \bmod 589 = 453$ и посылает его V.

2. V генерирует множество $S = (1, 5, 4)$ и отправляет его P.

3. P вычисляет $T_c = \prod_{j \in S} C_j = 90 \cdot 567 \cdot 263 = 13420890$ и посылает V число $y = 859 \cdot T_c \bmod 589 = 390$.

4. V находит $T_d = \prod_{j \in S} d_j = 472 \cdot 359 \cdot 283 = 47953784$, затем вычисляет $X = 390^2 \cdot 47953784 \bmod 589 = 453$ и проверяет равенство (5.7): $x = X$. Так как последнее равенство выполнилось, *V* делает вывод об успешной идентификации P.

Поручительство информации

Напомним сущность этого КП. Пользователь сети *A* отдает на хранение (*поручительство)* пользователю *B* некоторую битовую цепочку данных *M* (в частном случае – цепочку длины 1, т. е. один бит). *B* не может прочитать эту цепочку до определенного момента времени, задаваемого *A*, однако и *A* не может изменить позже содержание цепочки *M*, хранящейся у *B*. Данная задача решается следующим простым протоколом:

1. *B* генерирует случайную битовую цепочку *R* и посылает ее по сети к *A*.

2. *A* объединяет R со своим сообщением M и посылает B криптограмму $E = f_K(R, M)$, где $f_K(\cdot)$ – алгоритм шифрования некоторым стойким блоковым шифром на секретном ключе *K*.

3. *B* хранит *E* до поступления команды от *A.*

4. В определенное время *A* посылает *B* свой секретный ключ *K*, при помощи которого *B* дешифрует *E*.

5. B проверяет присутствие своей цепочки R в дешифрованном сообщении. Очевидно, что все задачи протокола решаются после выполнения данных шагов.

Можно использовать для решения той же задачи другой протокол, который не требует выполнения процедур шифрования/дешифрования и активности от пользователя *B*:

1. *A* генерирует случайные числа R_1, R_2.

2. *A* объединяет их со своим сообщением *M*, формируя цепочку $E = (R_1,\ R_2,\ M)$.

3. *A* выполняет хеширование *E* при помощи однонаправленной бесключевой хеш-функции $h(\cdot)$, т. е. вычисляет $h = h(E)$.

4. *A* посылает *B* цепочку $h,\ R_1$.

5. В определенное время *A* посылает *B* цепочку $E = (R_1, R_2, M)$.

6. *B* убеждается в правильности сохраняемого им сообщения *M*, выполняя хеширование цепочки *E*.

Безопасность данного протокола для *B* определяется тем обстоятельством, что *A* не может найти другую такую цепочку $(R_1,\ R'_2,\ M')$, для которой $h(R_1,\ R'_2,\ M') = h(R_1,\ R_2,\ M)$. (Если бы *A* не посылала *B* цепочку R_1, то он имел бы возможность изменить обе величины R_1, R_2, а затем подделать сообщение *M*.)

Тайное голосование (ТГ)

Напомним основные требования, предъявляемые обычно к процедуре тайного голосования:

1. Участвовать в выборах могут только те легитимные пользователи, которые были предварительно включены в списки, составленные *избирательной комиссией* (ИК).

2. Результат голосования каждого легитимного участника должен сохраняться в тайне от всех участников процедуры голосования (в том числе от других легитимных пользователей, ИК и всех посторонних), за исключением, конечно, самого подавшего голос.

3. Результат голосования каждого участника должен быть правильно учтен ИК и учтен лишь единожды.

4. При отсутствии легитимного избирателя в списке ИК или при неправильном учете результатов голосования в ИК, эти ошибки должны быть исправлены без нарушения тайны голосования.

5. ИК не может сфальсифицировать результаты тех легитимных избирателей, которые не захотели принять участие в голосовании.

Заметим, что для выполнения требований 3–5 необходимо участие самих избирателей, поскольку никто другой больше, чем они, не заинтересован узнать, приняли ли они участие в голосовании и правильно ли учтен их голос.

Рассмотрим сначала *принципиальные трудности*, возникающие при реализации протокола тайного голосования.

Ясно, что наиболее сложной для выполнения частью протокола при использовании только телекоммуникационных каналов является сохранение тайны результата голосования как для всех посторонних пользователей сети (включая и легитимных избирателей), так и для ИК. Защита от перехвата результатов тайного голосования со стороны всех пользователей, кроме ИК, может быть решена достаточно просто – путем передачи этих данных в ИК в зашифрованном виде. Однако, это не решит проблему относительно ИК, поскольку она должна знать результат голосования каждого легитимного участника, но она не должна быть в состоянии идентифицировать эти данные, связав их с именами пользователей. Для решения этой проблемы имеется лишь единственная возможность – использование так называемого *анонимного канала связи*. Это означает организацию такого телекоммуникационного канала, который в точности передает некоторые данные, но не позволяет определить адреса отправителей этих данных. Решение этого вопроса возможно при использовании:

- *ретранслятора*, обеспечивающего анонимность пересылки.
- *протокола анонимного вещания.*

Эти два способа имеют существенные недостатки. Первый требует участия в протоколе доверительных «третьих лиц», которые не должны раскрыть результаты голосования кому бы то ни было [29]. Второй способ (описанный в [27]) требует специальной организации вещательного канала и выполнения весьма сложного и требующего большего времени протокола. Новый подход к организации анонимного канала предложен в [30], но он к настоящему времени недостаточно проработан. Тем не менее будем в дальнейшем полагать, что анонимный канал существует, и включим его как один из компонентов в процедуру тайного голосования.

Однако даже если анонимный канал связи между избирателями и существует, это не решает полностью задачу выполнения протокола тайного голосования. Дело в том, что, принимая результаты голосования, ИК должна убедиться, что они присланы от легитимных избирателей, но подтверждение их легитимности не должно идентифицировать самих избирателей. Это, казалось бы неразрешимое, противоречие, может быть тем не менее решено при выполнении протокола *слепой подписи,* описанной подробно в [27] и рассмотренной в следующем разделе применительно к задаче тайного голосования. Кроме того, необходимо обеспечивать выполнение важного требования 3, которое может быть, в свою очередь, разбито на два требования:

- каждый легитимный участник не должен проголосовать более одного раза;
- голос каждого легитимного участника должен быть учтен (т. е. не может быть удален) ИК.

Что же касается требования 4, то оно оказывается менее жестким, поскольку реализуется при выполнении повторной процедуры голосования, которая, однако, не должна открыть голоса ее участников.

Наконец, принципиальные трудности могут возникнуть при выполнении требования 5, когда ИК может попытаться сфальсифицировать выгодные для нее результаты голосования легитимных, но не принявших участие в голосовании избирателей. Если во время выполнения протокола избиратель не известит об отказе продолжить голосование, то ИК может попытаться воспользоваться его голосом. Это для нее легко выполнить, так как именно сама ИК выдает разрешения на голосование. В связи с этим в протоколе предусмотрены некоторые процедуры для сокращения возможностей ИК в использовании голосов отказавшихся голосовать избирателей. Однако полным решением этой проблемы было бы извещение избирателя со стороны ИК о его отказе от голосования.

Ниже описан протокол ТГ по телекоммуникационным каналам связи (ТКС) с использованием техники криптографии с открытым ключом.

Стойкость такой техники шифрования базируется на предположении о невозможности быстрого решения задач факторизации и логарифмирования. При выборе

соответствующих параметров (разд. 3) необходимая стойкость может быть достигнута относительно всех известных до настоящего времени методов криптоанализа.

Описание протокола ТГ по ТКС

Весь протокол может быть разбит на три основные части:

- инициализация протокола (распределение исходных данных);
- выполнение основного протокола;
- коррекция основного протокола.
- Рассмотрим далее подробно выполнение каждой из этих частей.

Инициализация протокола

На этом этапе (рис. 3.10) выполняются следующие процедуры:

1) ИК составляет список легитимных участников голосования;

2) каждый избиратель генерирует свою пару ключей $d_{\text{и}}$; $n_{\text{и}}$, $k_{\text{и}}$ (закрытый/открытый ключ) и публикует открытый ключ на общедоступном сайте в Интернете;

3) ИК генерирует тройку чисел $n_{\text{ИК}}$, $d_{\text{ИК}}$, $k_{\text{ИК}}$ (модуль, закрытый/открытый ключ) и публикует открытый ключ на общедоступном сайте в Интернете.

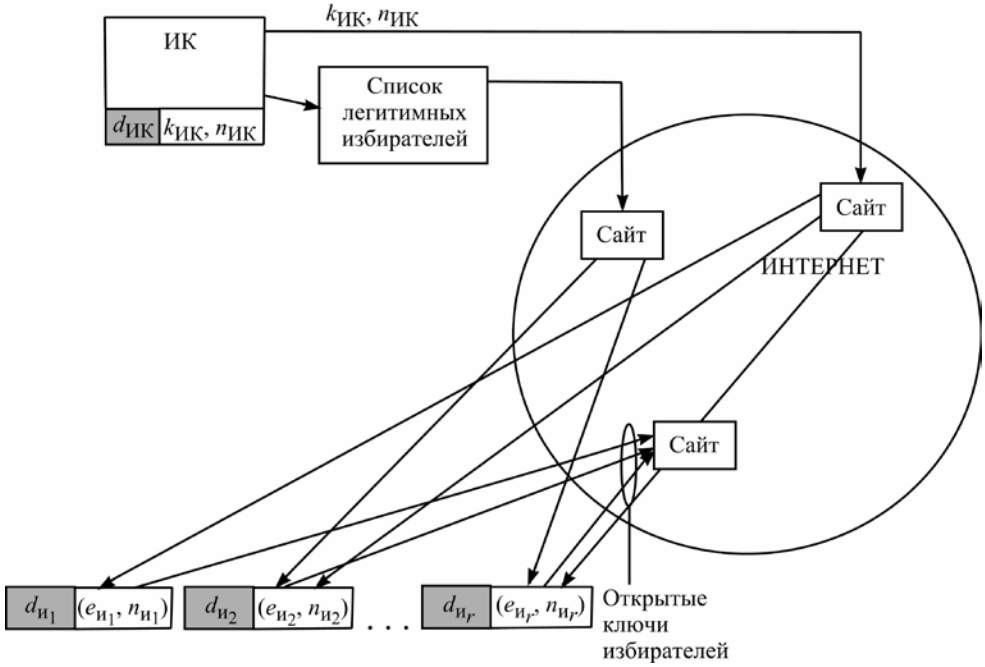


Рис. 3.10. Инициализация протокола тайного голосования

Выполнение основного протокола

Перечислим сначала шаги протокола ТГ (рис. 3.11, 3.12):

1) ИК публикует список всех правомочных избирателей;

2) избиратель, желающий принять участие в голосовании, публикует свой открытый ключ и генерирует свой *идентификационный номер I* (ИН);

3) избиратель маскирует свой идентификационный номер;

4) избиратель отправляет в избирательную комиссию *маскированный* ИН, зашифрованный дополнительно ключом избирателя;

5) ИК расшифровывает полученное сообщение;

6) ИК подписывает «вслепую» маскированный ИН;

7) избиратель демаскирует подписанный ИН;

8) избиратель создает бюллетень, зашифровывает его и отправляет в ИК по *анонимному каналу*;

9) ИК публикует полученное зашифрованное сообщение;

10) избиратель отправляет по анонимному каналу ключ для расшифровки сообщения;

11) ИК дешифрует сообщение и публикует результаты голосования.

Рассмотрим подробно шаги протокола ТГ.

1) ИК публикует на общедоступном сайте в Интернете пронумерованный список легитимных избирателей, подписанный своей цифровой подписью (ЦП), с использованием своего секретного ключа. (Заметим, что сквозная нумерация может быть заменена, для удобства, на какой-либо другой способ сортировки избирателей, обеспечивающий их быстрый поиск.) В дополнение к этому ИК публикует на общедоступном сайте заверенные своей ЦП сведения о порядке голосования, о времени выполнения каждого шага протокола и структуре сообщений, которыми должны обмениваться с ней избиратели при выполнении процедуры голосования.

На этом шаге обеспечивается *защита* от возможных недружественных действий участников протокола ТГ:

- *от ИК*. Так как ИК подписывает своей ЦП список избирателей, то она не заинтересована в том, чтобы внести в список нелегитимных избирателей. Также она не заинтересована в удалении из списка легитимных избирателей;

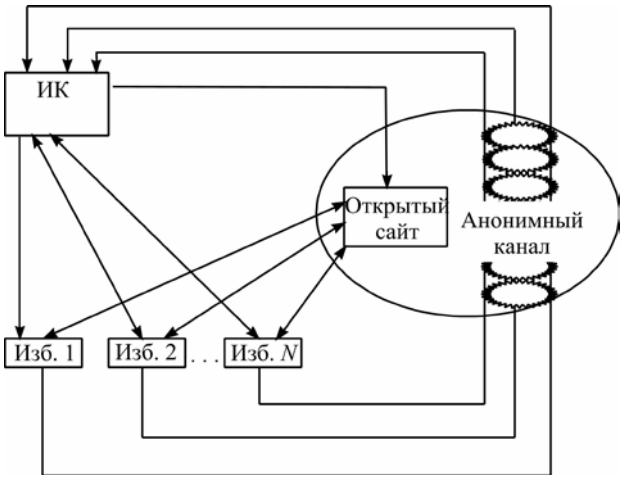


Рис. 3.11. Обмен информацией при выполнении протокола ТГ

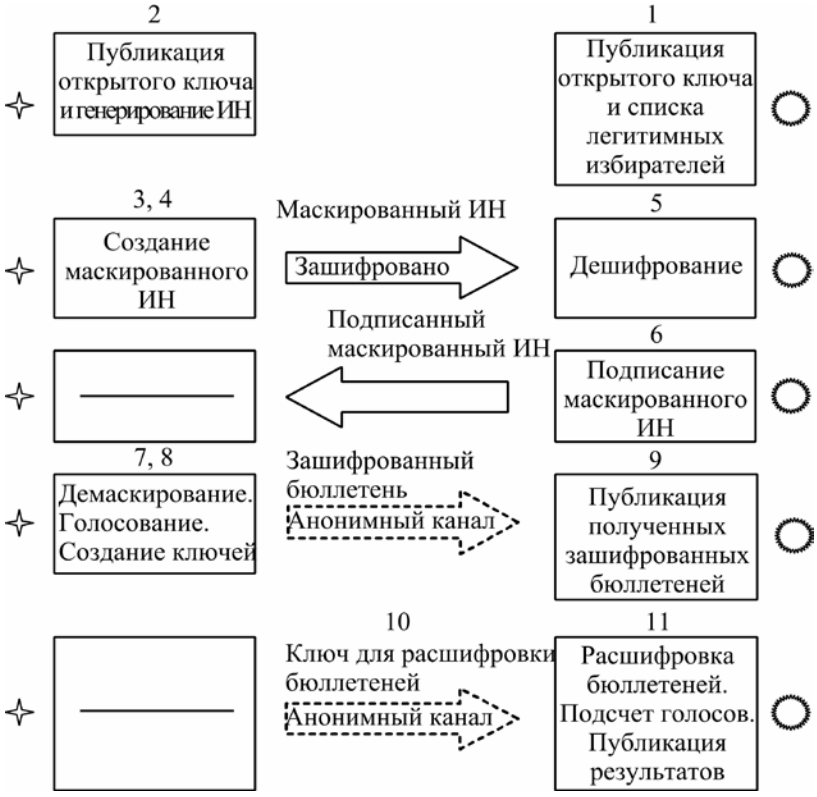


Рис. 3.12. Иллюстрация основных этапов протокола ТГ

- *от избирателей*. Избиратель не будет отказываться голосовать под предлогом того, что ИК игнорирует легитимных участников;

- *от посторонних*. Если ИК не будет подписывать список легитимных избирателей, то злоумышленник может подменить список. Это приведет к тому, что легитимные избиратели, которые не нашли себя в списке, будут протестовать о невнесении их в список, или часть из них откажется голосовать.

2) Каждый избиратель, который желает принять участие в голосовании, должен найти в списке легитимных избирателей свою фамилию и соответствующий ей номер. (Если он не нашел себя в этом списке, то он должен действовать в соответствии с процедурой коррекции выполнения основного протокола, описанной далее.) Если избиратель нашел свою фамилию и соответствующий ей номер в списке, то он должен сгенерировать случайной идентификацией номер I, который представляет собой цепочку цифр длиной l. Длина l выбирается таким образом, чтобы вероятность случайного выбора одного и того же идентификатора I у двух и более избирателей была пренебрежимо малой.

Этот ИН должен быть уникальным, чтобы ИК, получив впоследствии бюллетень с ним, была убеждена, что избиратель является одним из тех, кто имеет право на голосование. Однако ни ИК, ни кто-либо другой не должны знать, какой номер I имеется у данного избирателя, поскольку в противном случае результат голосования данного избирателя стал бы им известен.

Идентификационный номер I нужен для того, чтобы ИК знала, что данный избиратель имеет право на голос. В данном протоколе I создает избиратель и дает ИК подписать его. Подписанный идентификационный номер – это более чем достаточный признак права на голосование. Проблема заключается в том, что ИК не должна знать, какой идентификационный номер она подписала, хотя должна быть уверена, что обладателем этого ИН является легитимный избиратель. Эту проблему решает процедура слепой подписи.

3) Существует много способов выполнения подписи «вслепую». Рассмотрим далее один из самых простых. Идентификационный номер I умножается на *маскирующий множитель* (число) m. Однако так как его нужно будет в дальнейшем «сократить», этот множитель заранее возводится в степень открытого ключа ИК – $k_{ИК}$.

Избиратель маскирует сгенерированный им идентификационный номер I, подготавливая его для последующей слепой подписи со стороны ИК. Для этого он выполняет следующее преобразование:

$$I_m = m^{k_{ИК}} \cdot I \bmod n_{ИК}, \tag{3.27}$$

где $k_{ИК}$, $n_{ИК}$ – открытый ключ ИК, считанный избирателем с общедоступного сайта; m – случайно сгенерированное целое число из диапазона (1, 2, …, $n_{ИК} - 1$), которое является взаимно простым с $n_{ИК}$, т. е. при выполнении условия $\gcd(m, n_{ИК}) = 1$.

Для того чтобы найти значение ИН по маскированному номеру, ИК должна перебрать всевозможные m. Однако поскольку $n_{ИК}$ является очень большим, время для нахождения правильного I оказывается нереализуемым.

Посторонний наблюдатель, также как ИК, не знает маскирующего множителя и, значит, не сможет узнать истинное значение I.

4) Для того чтобы ИК подписала каждому легитимному избирателю по одному идентификационному номеру, она должна знать, что избиратели являются легитимными. Таким образом, каждый избиратель зашифровывает свой номер n и маскированный I (т. е. I_m) своим секретным ключом $d_и$.

Избиратель направляет к ИК по открытому каналу следующее сообщение:

$$M_1 = (n,\ E_{d_и}(n, I_m)), \tag{3.28}$$

где n – порядковый номер этого избирателя в списке легитимных избирателей, который он считал с общедоступного сайта; $E_{d_и}(n, I_m)$ означает процедуру асимметричного шифрования сообщений (n, I_m) с использованием секретного ключа избирателя $d_и$.

В данном сообщении для ИК скрыт только ИН избирателя, тогда как его номер в списке легитимных избирателей известен.

Если злоумышленник попытается от имени одного из избирателей отправить сообщение на подпись, то он не сможет создать подобное сообщение, так как не знает секретного ключа избирателя.

5**)** ИК публикует все принятые сообщения M_1 на открытом сайте. Далее ИК дешифрует криптограмму, содержащую сообщение M_1, используя для этого находящийся на общедоступном сайте открытый ключ избирателя ($k_{\text{и}}$, $n_{\text{и}}$) с номером n. Если значения n, содержащиеся в открытой части сообщения M_1 и в расшифрованной криптограмме, совпадут, то ИК может быть уверена, что это сообщение действительно получено от n-го избирателя.

Публикация сообщений M_1 необходима для того, чтобы знать, кто из легитимных избирателей не будет голосовать. ИК не сможет воспользоваться голосами тех, кто изначально не принимал участия в голосовании. Если сообщения M_1 не будут опубликованы, то ИК может имитировать избирателей. Так как в сообщении M_1 используется шифрование с секретным ключом избирателя, то подделать это сообщение невозможно. Значит, публикация этих сообщений определит голосующих и удалит те голоса, которыми могла бы воспользоваться нечестная ИК. Это весьма важно потому, что большая часть избирателей, отказавшихся от голосования, не станет генерировать ИН и посылать их в ИК.

6**)** Если некоторый избиратель является легитимным, то ИК подписывает его маскированный идентификационный номер I_m, выполняя, например, процедуру цифровой подписи RSA:

$$I_{sm} = {I_m}^{d_{\text{ИК}}} \bmod n_{\text{ИК}}, \tag{3.29}$$

После этого ИК отправляет I_{sm} по открытому каналу к n-му избирателю и помещает его (вместе с номером n) на общедоступном сайте.

Даже если бы ИК не хотела этого, ей пришлось бы подписать сообщение и отправить его избирателю. При любых других вариантах обман со стороны ИК тривиально раскрывается.

С другой стороны, избиратель не сможет отказаться от того, что он получил подписанные данные. Несмотря на то что подписанный маскированный ИН находится на общедоступном сайте, это не дает возможности определить истинный ИН без знания маскирующего множителя.

7**)** Избиратели производят демаскирование своих подписанных идентифицированных номеров, выполняя следующие преобразования над полученными от ИК значениями I_{sm}:

$$I_s = (I_{sm})/m \bmod n_{\text{ИК}}. \tag{3.30}$$

Действительно, подставляя (3.27) и (3.29) в (3.30), получаем

$$I_{sm}/m = (I_m)^{d_{\text{ИК}}} \bmod n_{\text{ИК}} \cdot m^{-1} \bmod n_{\text{ИК}} = \\ = (m^{k_{\text{ИК}}})^{d_{\text{ИК}}} \cdot I^{d_{\text{ИК}}} \cdot m^{-1} \bmod n_{\text{ИК}} = \\ = m \cdot I^{d_{\text{ИК}}} \cdot m^{-1} \bmod n_{\text{ИК}} = I^{d_{\text{ИК}}} \bmod n_{\text{ИК}}. \quad (3.31)$$

После выполнения преобразования (3.31), избиратель проверяет подлинность подписи ИК: $(I^{d_{\text{ИК}}})^{k_{\text{ИК}}} \bmod n_{\text{ик}} = I$, используя известный открытый ключ ИК. Если это равенство не выполняется, то он выражает протест (см. ниже коррекцию выполнения основного протокола).

Таким образом, все легитимные избиратели после выполнения шага 7 будут иметь свои идентифицированные номера *I*, подписанные ИК, что и удостоверит их легитимность при выполнении последующих шагов протокола. С другой стороны, ИК не знает ИН избирателей, поскольку она подписала их в маскированном виде.

8**)** Избиратели голосуют, включая результаты голосования в сообщение *V*. После этого каждый из избирателей генерирует пару собственных ключей (k_v, n_v и d_v) для шифрования/дешифрования результатов голосования. Далее каждый из избирателей формирует следующее сообщение:

$$M_2 = (P,\ E_{d_v}\ (I,\ I_s,\ V)), \quad (3.32)$$

где P – это любое число, используемое для удобного поиска M_2 на открытом сайте. (Оно может быть даже одинаковым у различных избирателей.) Избиратель посылает это сообщение по *анонимному каналу* к ИК. Необходимость в шифровании сообщения (I, I_s, V) объясняется следующим образом. Если бы его передавали в открытом виде, как, скажем, $M_2^{'} = (P,\ I,\ I_s,\ V)$, то ИК могла бы удостоверить свою подпись и учесть результат голосования V как легитимный, однако существовала бы опасность *нарушения целостности* сообщения $M_2^{'}$ со стороны злоумышленников, т. е. замены его на $M_2^{"} = (P,\ I,\ I_s,\ V'))$, где $V' \neq V$, что соответствовало бы другому результату голосования, причем эта подмена не была бы обнаружена ИК.

Таким образом, ИК не знает, кто послал сообщение M_2, так как избиратели используют анонимные каналы для отправки этих сообщений. Даже если сообщение было послано от нелегитимного избирателя, ИК узнает об этом только после его расшифровки.

Поскольку избиратель имеет только один подписанный ИН, сможет проголосовать всего один раз.

Только владельцы подписанных ИН смогут создать подобные сообщения, все же остальные сообщения будут на следующих шагах признаны ИК недействительными.

9**)** ИК публикует M_2 (в зашифрованном виде) на открытом сайте.

10**)** Избиратели направляют в ИК сообщение M_3 по анонимному каналу в следующем виде:

$$M_3 = (P,\ k_v,\ n_v). \qquad (3.33)$$

ИК, получив это сообщение, сможет расшифровать M_2 и, следовательно, учесть голос не известного ей, но легитимного (поскольку его идентификатор ею был подписан) избирателя. Если этот голос «не устроит» нечестную ИК, то она может попытаться проигнорировать это сообщение. (Решение данной проблемы дано в коррекции к протоколу ТГ, ситуация 4.)

Очевидно, что те, кто не имел ИН, подписанных ИК, могут послать подобное сообщение, но их голос не будет учтен ИК.

11**)** ИК дешифрует криптограмму, содержащуюся в M_2, используя для этого ключ дешифрирования k_v, n_v. В результате ИК, получая I, I_s и V, может убедиться в легитимности результата голосования и учесть результат голосования V, представленный теперь в открытом виде, который она затем публикует (совместно с ИН) на открытом сайте для проверки правильности учета результата голосования каждым избирателем.

Дополнительные пояснения к протоколу тайного голосования

Использование анонимного канала как на шаге 8, так и на шаге 10 обязательно, поскольку в противном случае ИК может связать I с конкретным избирателем и, следовательно, нарушить тайну голосования.

Если избиратель выполнил шаг 4, но затем решил не голосовать, то он должен известить об этом ИК. Избиратель создает в этом случае сообщение вида:

$$M_a = (n,\ E_{d_и}(n,\ R,\ I,\ I_s)),$$

где n – номер избирателя; R – признак отказа от голосования (значение его должно быть заранее определено); I – ИН (может отсутствовать в сообщении); I_s – подписанный ИН (может отсутствовать в сообщении); $E_{d_и}(n,\ r)$ – криптограмма сообщений n и r, зашифрованных секретным ключом избирателя.

Избиратель отправляет это сообщение в ИК, а кроме того выбирает по своему усмотрению избирателей, число которых должно быть заранее определено, и отправляет каждому из них то же самое сообщение M_a. ИК при публикации результатов голосования предоставляет дополнительную информацию об отказе избирателей от голосования, если такие имеются. Если избиратель уже получил подписанный ИН, то «раскрытие» подписанного ИН становится обязательным, чтобы предотвратить его использование.

Нечестная ИК не может сфабриковать результат голосования для тех избирателей, которые изначально не участвовали в голосовании, поскольку голосующий избиратель на шаге 4 отправляет сообщение M_1 и тем самым подтверждает желание

голосовать. Такое сообщение сам ИК создать не может, так как для этого используется секретный ключ избирателя. Разумно предположить, что бóльшая часть отказавшихся голосовать избирателей не будет голосовать изначально. Однако остается малая часть избирателей, которая выполнит шаг 4 и затем на любом из шагов протокола (за исключением шага 11) откажется продолжать голосование. После шага 4 избиратель обращается к ИК только по анонимному каналу и определить, кто посылает сообщения в ИК, невозможно. Это, казалось бы, означает, что нечестные пользователи могут попытаться послать сообщения в ИК, не являясь легитимными участниками. Однако для того чтобы их голоса были учтены, необходимы ИН, подписанные ИК. Поскольку ИК может создавать фальсифицированные ИН, она может попытаться проголосовать за избирателя, воспользовавшись отсутствием действий избирателя после шага 4. Обман ИК будет раскрыт, если бóльшая часть избирателей, отказавшихся от голосования после шага 4, предоставит свои демаскированные подписанные ИН как доказательство того, что они не приняли участия в голосовании. В то же время обман не раскроется, если избиратели, отказавшиеся продолжать голосование, не будут этого делать.

Следовательно, если избиратель отказывается голосовать, то узнать об этом можно только от самого избирателя, и ИК должна быть не единственной, кто узнает об этом. Исключения этой угрозы можно достигнуть, используя сообщение избирателя об отказе от голосования некоторому количеству других избирателей (число которых определяется до начала голосования). Представляется весьма вероятным, что кто-нибудь из этих «доверенных» избирателей проверит, учла ли ИК «отказы» от голосования, и если нет, то он пошлет протест (см. ниже *коррекцию протокола*), содержащий в качестве подтверждения сообщение M_a. (Заметим, что сообщение M_a является неопровержимым доказательством того, что определенный избиратель отказался голосовать.)

Совершенно необходимо, чтобы на всех шагах протокола ТГ (вплоть до шага 11) избиратель не открывал свой ИН. Предположим противное: на шаге 8 в сообщении M_2 вместо числа P открыто передается ИН, т. е.

$$M_2 = (I,\, E_{d_v}(I_s,\, V)).$$

Тогда у ИК, пока не закончится этот шаг, остается время на создание сообщения с таким же ИН. Она подписывает ИН, получая I_s, и затем создает V', в котором она заинтересована. Далее, имитируя избирателя, ИК создает ключи $k_v^{'}$ и $d_v^{'}$, шифрует сообщение и отправляет его *самой себе*. В результате таких нечестных действий ИК окажется, что у некоторых избирателей совпадут ИН, а это может означать, что некоторые избиратели решили проголосовать несколько раз с одним и тем же ИН. Так как никто не знает истинного владельца ИН, то доказать, что именно ИК созда-

ла это сообщение, оказывается невозможным. Избиратель мог попытаться дважды проголосовать, и одно из его сообщений необходимо исключить. Однако органом, который исключает такие сообщения, является именно ИК, и она может оставить свое сообщение, т. е. фактически проголосовать вместо избирателя. Впоследствии избиратель не сможет доказать, что он не отправлял два сообщения, и поэтому уличить ИК в нечестных действиях будет невозможно.

По окончании протокола может произойти конфликтная ситуация, состоящая в совпадении некоторых ИН. Это будет в трех случаях:

а) у избирателей случайно совпали ИН;

б) нечестный избиратель при помощи своего ИН решил проголосовать несколько раз;

в) нечестная ИК создала дополнительный ИН и решила проголосовать, но он случайно совпал с ИН избирателя.

Понятно, что изменять ИН в этих случаях не имеет смысла, так как если совпадение номеров произошло из-за случая «б», то при замене ИН один избиратель сможет проголосовать несколько раз, поэтому в данном протоколе нельзя требовать замены ИН.

Так как вероятность совпадений ИН очень мала, то принятый голос можно отнести к разряду «сомнительных». Если два (или более) сомнительных голоса сделали один и тот же выбор, то разумно учесть в результатах только один из них. Если же голоса отличаются, то не учитывается ни один из голосов. Все операции с «сомнительными» голосами должны быть опубликованы вместе с результатами голосования.

Если «сомнительные» голоса являются решающими, то наилучшим способом выхода из ситуации будет проведение *повторного голосования*. Вполне возможно, что какой-либо злоумышленник, зная открытый ключ ИК, может попытаться подделать подпись ИК. Допустим, злоумышленник подбирает такое a, что шифрование этого числа открытым ключом ИК приводит к идентификатору I, который можно представить как некоторый ИН:

$$(a)^e \bmod n = I. \tag{3.34}$$

Далее злоумышленник формирует на шаге 8 бюллетень, где вместо подписанного ИН I_s записывает значение a, т. е. формирует сообщение:

$$M_2 = (P,\ E_{d_v}(I,\ a,\ V)).$$

Поскольку ИК не знает отправителя этого сообщения, она считает, что оно пришло от легитимного избирателя. При проверке подписи ИК выполнит операцию (5.16) и, следовательно, подтвердит достоверность подписи. Таким образом, злоумышленник сможет проголосовать, даже не являясь легитимным избирателем. Ре-

шением данной проблемы является уменьшение вероятности выполнения соотношения (5.16), где I – допустимое значение ИН до такой степени, что таким событием можно пренебречь. Это вполне возможно при ограничении значений I и увеличении n.

Коррекция выполнения основного протокола

При выполнении данного протокола могут возникнуть следующие *конфликтные ситуации:*

1. Некоторые избиратели оказались не включенными в список легитимных избирателей.

2. Подписи I_s под ИН у некоторых избирателей оказались неверными.

3. ИК не учла некоторых из тех, кто отказался голосовать после шага 4.

4. Некоторые результаты голосования случайно не учтены или учтены неверно.

Если один из таких конфликтов будет иметь место, то избиратель (совместно с честной ИК) приступит к коррекции протокола голосования.

Коррекция ситуаций в случаях 1 и 2 тривиальна. Действительно, для этого избирателям достаточно обратиться к ИК по открытым каналам и выслать ей свои претензии, причем в случае 2 это сводится к просьбе повторения шагов протокола 2–7.

Рассмотрим ситуацию 3, когда ИК при публикации не учла голосов тех, кто отказался голосовать. (ИК должна была опубликовать список тех, кто перестал голосовать после шага 4.) Так как «отказавшийся» избиратель отсылает некоторому количеству «доверенных» избирателей сообщения об отказе от продолжения голосования, то избиратели, получившие сообщения M_a, обращаются к ИК с этими сообщениями и требуют повторного расчета результатов голосования. (Напомним, что сообщение M_a было подписано секретным ключом избирателя, а это является достаточным признаком его достоверности.)

При возникновении ситуации 4 только сам избиратель сможет узнать, правильно ли учтен его голос. Если это не так, то он выражает протест и повторно отправляет по анонимному каналу сообщения M_2 и M_3, которые были посланы ранее на шагах 8 и 10. Так как ИК публиковала все сообщения M_2 на общедоступном сайте, она не сможет проигнорировать эти сообщения. В случае если ИК на шаге 10 проигнорировала сообщение M_3 или сам избиратель не послал на этом шаге M_3, то он во время выполнения дополнительного протокола *протеста* имеет право на учет его голоса.

Как видно, все мыслимые пока конфликтные ситуации разрешаются без спорных вопросов. В целом протокол тайного голосования является достаточно сложной процедурой, требующей для своего выполнения наличия качественных каналов связи между избирателями и ИК, времени и вычислительных ресурсов. При наличии

большого количества избирателей (например, сотен тысяч или миллионов) практическая реализация подобных протоколов требует дальнейшей проработки. Что же касается процедуры тайного голосования по телекоммуникационным каналам связи при ограниченном контингенте избирателей (скажем, в рамках решения корпоративных проблем), то рассмотренный выше алгоритм электронного голосования по открытым каналам связи оказывается вполне приемлемым, правда, при надежном решении проблемы создания анонимного канала.

Краткие выводы по криптографическим протоколам

Выше были рассмотрены алгоритмы выполнения пяти КП из достаточно большого списка, содержащего 18 таких КП, который в действительности мог бы быть еще и продлен. Очевидно, что столь же подробное изложение объемного материала обо всех оставшихся КП вышло бы за рамки краткого учебного пособия по основам криптографии с открытым ключом. Описание способов выполнения большинства из оставшихся протоколов можно найти в монографиях [27, 28]. Важно отметить, что для обоснования «не учебной» секретности КП существует достаточно развитая теория, использующая понятие *семантической стойкости* и другие математические понятия [18], только применение которых может дать гарантии защищенности КП от всех мыслимых атак.

Отметим также, что из представленного в начале этого раздела списка КП наиболее важными в практическом отношении и не рассмотренными подробно выше представляются такие, как *варианты ЦП*, *обманчивая передача* и *секретные совместные вычисления*.

ЗАКЛЮЧЕНИЕ

Настоящее пособие, состоящее из трех разделов, предназначено для студентов, специализирующихся в области *информационной безопасности*.

Отметим, что студенты, хорошо изучившие данный материал, могут успешно решать следующие профессиональные задачи:

- реализовывать совершенные (одноразовые) шифры и понимать ограниченность их применения;
- реализовывать доказуемо стойкие к линейному и дифференциальному криптоанализу алгоритмы блокового шифрования на основе использования ППШ;
- понимать свойства и особенности применения различных мод шифрования;
- понимать стойкость методов многократного шифрования;
- реализовывать доказуемо стойкие к некоторым методам криптоанализа потоковые шифры;
- знать основные побочные атаки на блоковые и потоковые шифры и уметь защититься от них;
- понимать работу основных современных стандартов шифрования;
- реализовывать основные алгоритмы шифрования с открытым ключом при выборе параметров, обеспечивающих стойкость этих шифров к основным и побочным атакам;
- иметь представление о реализации алгоритмов шифрования с открытым ключом над эллиптическими кривыми;
- реализовывать некоторые стойкие алгоритмы цифровой подписи;
- понимать проблематику криптографических протоколов и алгоритмы выполнения некоторых из них.

Литература

1. Виноградов, И.М. Основы теории чисел. – М. : ФМ, 1965.
2. Питерсон, У. Коды, исправляющие ошибки / У. Питерсон, Э. Уэлдон. – М. : Мир, 1976.
3. Menezes, A.J. Handbook of Applied Cryptography. – New-York, London, Tokyo : CRC Press, 1997.
4. Bennet, C. et. al. Generalized privacy amplification // IEEE Trans. on IT. – 1995. – Vol. 41, n 6. – P. 1915–1923.
5. Шеннон К.Э. Работы по теории информации и кибернетике. М.: И.Л., 1963.
6. Schneier B. Applied Cryptography Second Edition: protocols, algorithms and source code in C. John Wiley & Sons Inc., 1996. (Русский перевод: Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. М.: ТРИУМФ, 2002.)
7. Spillman R. Cryptanalysis of knapsack ciphers using genetic algorithms // Cryptologia, 17(1), 1993.
8. Степашкин М.В., Котенко И.В., Богданов В.С. Оценка защищенности компьютерных сетей на основе анализа графов атак // Пятая Общероссийская Конференция «Математика и безопасность информационных технологий» (МаБИТ-06). Москва, МГУ, 2006.
9. Hong, S. Provable Security Against Differential and Linear Cryptanalysis for SPN Structure // FSE-2000, LNCS. – 2000. – Vol. 1978. – P. 273–278.
10. Изотов, Б.В. Методы конструирования блочных шифров на базе SP- и SL-сетей // Вопросы защиты информации. – 2004. – № 3. – С. 24–37.
11. Бабаш, А.В. Криптография. – М. : Солон, 2002.
12. Коржик, В.И. Теоретические основы информационной безопасности телекоммуникационных систем: учебное пособие / В.И. Коржик, Д.В. Кушнир; СПбГУТ. – СПб, 2000.
13. Зенин, О.С. Стандарт криптографической защиты AES / О.С. Зенин, М.А. Иванов. – М. : Кудиц-образ, 2002.
14. Henk, C.A. van Tilborg. Fundamentals of Cryptology. – Boston, Dordrecht, London : Kluwer, 2000.
15. Wegman, M. New Hash Functions and Their Use in Authentification and Set Equality / M. Wegman, L. Carter // Journal of Computer and System Sciences. – 1981. – Vol 22. – P. 265–279.
16. Preneel, B. Cryptographic Hash Functions. – Kluwer Academic Publishers, 1994.
17. Koblitz, N. A Course in Number Theory and Cryptography. – New-York : Springer, 1987.
18. Мао, В. Современная криптография. – М., СПб., Киев : Вильямс, 2005.
19. Henk C.A. van Tilborg. Fundamentals of Cryptography. – Boston, Dortecht, London : Kluwer, 2001.
20. Коржик, В.И. Основы защиты информации в компьютерных системах : методические указания к лабораторным работам. Часть 2 / В.И. Коржик, Д.В. Кушнир. – СПб. : СПбГУТ, 1999.
21. Berlecamp, E.R. et al. On the inherent intractability of certain coding problems // IEEE Trans. on IT. – 1978. Vol. 24. – P. 384–386.

22. Болотов, А.А. и др. Элементарное введение в эллиптическую криптографию. – М. : КомКнига, 2006.
23. Питерсон, У. Коды, исправляющие ошибки / У. Питерсон, Э. Уэлдон. – М. : Мир, 1976.
24. Молдавян, Н.А. Введение в криптосистемы с открытым ключом / Н.А. Молдавян, А.А. Молдавян. – СПб. : БХВ-Петербург, 2005.
25. ГОСТ Р 3410–2001. Информационная технология. Криптографическая защита информации. Процедуры выработки и проверки цифровой подписи на базе асимметричного криптографического алгоритма. – М. : Изд-во стандартов, 2001.
26. Феллер, В. Введение в теорию вероятностей и ее приложения : т.1. – М. : Мир, 1976.
27. Шнайер, Б. Прикладная криптография. – М. : Триумф, 2002.
28. Salomaa, A. Public – key Cryptography. – Berlin : Springer, 1990.
29. Schneier, Bruce. E-Mail Security. – New-York : Wiley, 1995.
30. Sherwood, Rob et al. A protocol for Scalable Anonymous Communication // Trans. Eurocrypt. – 2001. – P. 1–13.

УНИВЕРСИТЕТ ИТМО

Миссия университета – генерация передовых знаний, внедрение инновационных разработок и подготовка элитных кадров, способных действовать в условиях быстро меняющегося мира и обеспечивать опережающее развитие науки, технологий и других областей для содействия решению актуальных задач.

КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

На кафедре ВТ проводятся научные исследования в соответствии с программой развития научной школы кафедры «Организация вычислительных систем и сетей», включенной в реестр ведущих научных и научно-педагогических школ Санкт-Петербурга. Магистранты и аспиранты активно участвуют в научно-исследовательских работах по следующим основным направлениям.

Работы в области **вычислительных систем и сетей** направлены на разработку методов и средств системотехнического проектирования вычислительных систем и сетей на основе аналитических и имитационных моделей и измерений на реальных системах. Решаются задачи оценки эффективности и оптимизации отказоустойчивых высоконадежных систем, создания методологии комплексного обеспечения надежности, безопасности и устойчивости функционирования систем в условиях сбоев, отказов и внешних деструктивных воздействий. Разрабатываются методы и средства реализации информационной безопасности и защиты информации от несанкционированного доступа в вычислительных системах.

Исследования в области **параллельных и распределенных вычислений** связаны с решением задач анализа технологий параллельного программирования в системах с общей памятью и модификацией последовательных алгоритмов для выполнения в параллельном режиме. Исследования включают в себя анализ и разработку lock-free структур данных, оценку их эффективности при различных нагрузках. Исследуется эффективность организации параллельных вычислений на графических процессорах с использованием технологии CUDA. Значительное внимание уделяется инструментам и методам динамического анализа выполнения параллельных программ.

Исследования в области **обработки и распознавания цифровых изображений и аудио сигналов** направлены на разработку новых методов, алгоритмов и программных средств для решения задач обработки и анализа изображений с целью повышения их качества, анализа динамических изображений с целью формирования траектории движения объекта, распознавания изображений объектов независимо от их положения, ориентации и масштаба, спектральной обработки изображений. Проводятся исследования эффективности

существующих и разработка новых методов маркирования изображений встраиваемыми в них цифровыми водяными знаками и методов повышения робастности (устойчивости) распознавания в мультимодальных биометрических системах.

Более 30 лет на кафедре ведутся работы в области **микропроцессорной техники и встраиваемых вычислительных систем**, связанные с разработкой и исследованием распределенных информационно-управляющих систем с высокой надежностью, контроллерных сетей промышленной и транспортной автоматики, средств автоматизации программирования и отладки распределенных систем реального времени. В рамках научного направления "Автоматизация высокоуровневых этапов проектирования информационно-управляющих систем" решаются задачи создания встраиваемых систем, систем реального времени, реконфигурируемых систем и систем-на-кристалле. Ведутся работы по развитию методологии проектирования киберфизических систем. Стремительно развивается направление работ по созданию IP-ядер для систем-на-кристалле.

Работы в области **интеллектуальных информационных систем** нацелены насоздание быстрых алгоритмов поиска в базах знаний, организацию данных в базах знаний, обеспечивающую быстрый логический вывод, извлечение знаний из неформализованных источников, в том числе из социальных сетей. Кроме интеллектуальных информационных систем в сферу интересов кафедры входят также интеллектуальные методы управления в технических системах.

Кроме того, на кафедре проводятся научные исследования, связанные с проблемами **проектирования, разработки, сопровождения и реинжиниринга корпоративных информационных систем**, а также с разработкой **преобразователей перемещений на основе рекурсивных кодовых шкал**с улучшенными массогабаритными, технологическими и надежностными характеристиками.

Сотрудники кафедры участвуют в работе Международных научных лабораторий:

• «Архитектура и методы проектирования встраиваемых систем и систем на кристалле».

• «Лаборатория нелинейных и адаптивных систем управления».

• «Многомодальные биометрические и речевые системы».

Существующие международные программы

На кафедре реализуются совместные программы подготовки:

• бакалавров с Пекинским политехническим университетом;

• магистров по программе двойного диплома с Казахским национальным университетом им. аль-Фараби;

• магистров по программе двойного диплома с Восточно-Казахстанским государственным техническим университетом им. Серикбаева.

Ожиганов Александр Аркадьевич

Криптография

Учебное пособие

В авторской редакции
Редакционно-издательский отдел Университета ИТМО
Зав. РИО Н.Ф. Гусарова
Подписано к печати
Заказ №
Тираж
Отпечатано на ризографе

Редакционно-издательский отдел
Университета ИТМО
197101, Санкт-Петербург, Кронверкский пр., 49