Содержание

1. Асимметричные криптосистемы

В учебном пособии [3] было дано общее определение асимметричной криптосистемы, отличительной особенностью которой является то, что для зашифрования и расшифрования информации используются разные ключи. Криптосистема с открытым ключом определяется тремя алгоритмами: генерации ключей, шифрования и расшифрования. Алгоритм генерации ключей позволяет получить пару ключей $(k_o, k_з)$, причем $k_o \neq k_з$. Один из ключей k_o публикуется, он называется **открытым**, а второй $k_з$ называется **закрытым** (или секретным) и хранится в тайне.

1.1. Общая характеристика и классификация асимметричных криптосистем

Определение 1.1. Под криптосистемой с открытым ключом понимают систему [6,7]:

$$ACS = (X, Y, K, x, y, k_o, k_з, E, D), \tag{1.1}$$

где X - множество открытых текстов, Y - множество криптограмм, K - множество ключей, $x \in X$ - некоторый открытый текст, $y \in Y$ - некоторая криптограмма, $k_o \neq k_з$, $(k_o, k_з) \in K$ - пара ключей, E - функция шифрования, D - функция расшифрования.

Центральным понятием теории асимметричных криптосистем является понятие односторонней функции [6]. Пусть дана функция:

$$y = f(x), \tag{1.2}$$

определенная на множестве $X, x \in X$, для которой существует обратная функция:

$$x = f^{-1}(y). \tag{1.3}$$

Функция называется **односторонней**, если вычисление по формуле (1.2) является сравнительно простой задачей, требующей немного времени, а вычисление по (1.3) – задача сложная, требующая привлечения массы вычислительных ресурсов, например, 10^6-10^9 лет работы мощного суперкомпьютера. Данное определение не является строгим. Рассмотрим более точное определение односторонней функции [9].

Определение 1.2. Пусть X и Y - конечные множества. Односторонней функцией:

$$f : X \to Y \tag{1.4}$$

называется обратимая функция, удовлетворяющая следующим условиям:

1) f легко вычисляется, т.е. если дано $x \in X$, $y = f(x)$ вычислима за полиномиальное время (существует полиномиальный алгоритм вычисления y);

2) f^{-1} - обратная функция к f, трудно вычисляется, т.е. если дано $y \in Y$, $x = f^{-1}(y)$ является вычислительно неразрешимой (не существует полиномиального алгоритма вычисления x).

Собственно односторонняя функция в криптографии не используется. Применяется **односторонняя функция с секретом** (односторонняя функция с «лазейкой», односторонняя функция с «ловушкой»).

Определение 1.3. Пусть X и Y - конечные множества. Односторонней функцией с секретом

$$f : X \to Y \qquad (1.5)$$

называется обратимая функция, удовлетворяющая следующим условиям:

1) f легко вычисляется, т.е. если дано $x \in X$, $y = f(x)$ вычислима за полиномиальное время (существует полиномиальный алгоритм вычисления y);

2) f^{-1} - обратная функция к f, трудно вычисляется, т.е. если дано $y \in Y$, $x = f^{-1}(y)$ является вычислительно неразрешимой (не существует полиномиального алгоритма вычисления x).

3) f^{-1} легко вычисляется, если известен секрет, связанный с параметрами функции.

Таким параметром в асимметричных криптосистемах является, как правило, закрытый ключ $k_з$.

Рассмотрим односторонние функции, используемые в криптографии [6].

Дискретное экспоненцирование и логарифмирование. Пусть

$$y = a^x \bmod p, \qquad (1.6)$$

где p - некоторое простое число, а $x \in \{1, 2, ..., p-1\}$. Обратная функция обозначается:

$$x = \log_a y \bmod p \text{ и} \qquad (1.7)$$

называется **дискретным логарифмом**.

Рассмотрим на примере простоту вычисления (1.6). Пусть требуется вычислить $y = 3^{100} \bmod 7$. Введем величину $t = \lfloor \log_2 x \rfloor = \lfloor \log_2 100 \rfloor = 6$, представляющую собой целую часть $\log_2 x$ и определим числовой ряд $a^{2^t} \bmod p$:

a	a^2	a^4	a^8	a^{16}	a^{32}	a^{64}
3	2	4	2	4	2	4

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

$$100 = (1100100)_2$$

и в соответствии с выражением [6]:

$$y = \prod_{i=0}^{t} a^{x_i \cdot 2^i} \bmod p, \qquad (1.8)$$

вычислим значение $y = 4$. Для вычислений потребовалось всего восемь операций умножения.

Справедливо утверждение [6]: количество операций умножения при вычислении (1.6) по рассмотренному методу не превосходит $2\log_2 x$.

Задача вычисления (1.7) не является тривиальной. Для ее решения используются различные математические методы, при этом требуется $2\sqrt{p}$ операций. Следовательно, чем больше значение p, тем сложнее вычислить (1.7). Например, если количество десятичных знаков в записи числа p равно 90, то количество операций для вычисления (1.6) равно примерно 600. Для перспективных компьютеров (для современных компьютеров это пока недоступно) время на одну операцию составляет 10^{-14} сек., тогда вычисление (1.6) занимает $6 \cdot 10^{-12}$ сек., а вычисление (1.7) занимает 10^{31} сек.$\approx 10^{22}$ лет (количество операций - 10^{45}).

Таким образом, можно утверждать, что (1.7) является односторонней функцией. Вместе с тем, не существует строгого математического доказательства отсутствия возможности вычисления (1.7) также «быстро», как и (1.6).

Умножение и факторизация. Другим примером односторонней функции является задача факторизации. Существо ее базируется на двух фактах из теории чисел:

1) задача проверки чисел на простоту является сравнительно легкой;

2) задача разложения чисел вида:

$$n = pq, \qquad (1.9)$$

является очень трудновыполнимой, если известно только n, а p и q - большие простые числа.

Возведение в квадрат и извлечение квадратного корня по модулю. Пусть x и N два целых числа, причем $N = pq$, а p и q - простые числа. Тогда прямая функция вычисляется по формуле:

$$y = x^2 \bmod N. \qquad (1.10)$$

Обратная функция представляет собой операцию вычисления квадратного корня по $\bmod N$. Эта операция также вычислительно сложна, как и задача факторизации.

Задача об укладке рюкзака (задача о ранце). Пусть имеется n объектов, при которых можно составить n-компонентный вектор $\mathbf{B}$, так что

i-й компонент вектора $\mathbf{B}$ представляет собой место, занимаемое i-м объектом. Имеется рюкзак общим объемом V.

Задача укладки рюкзака может быть сформулирована следующим образом. Даны $\mathbf{B}$ и V, требуется найти битовый вектор $\mathbf{X}$, такой что

$$\mathbf{B} \cdot \mathbf{X} = V. \qquad (1.11)$$

В общем случае не существует эффективного алгоритма вычисления $\mathbf{X}$ по $\mathbf{B}$ и V. Таким образом, можно использовать вектор $\mathbf{B}$ для шифрования n-битового сообщения $\mathbf{X}$ путем вычисления произведения (1.11).

В настоящее время разработаны математические методы, позволяющие в задаче «об укладке рюкзака» эффективно вычислять как прямую, так и обратную функции.

Существуют и другие односторонние функции, основанные, например, на сложности декодирования случайных линейных кодов. Однако в асимметричной криптографии широко используются только две первые, рассмотренные нами, односторонние функции.

1.2. Элементы теории чисел

Алгоритмы асимметричной криптографии базируются на классической теории чисел. Рассмотрим минимум из этой теории, позволяющий уяснить суть методов и алгоритмов асимметричной криптографии [5,6].

Определение 1.4. Целое положительное число p называется простым, если оно делится без остатка только на единицу и само на себя.

Определение 1.5. Два числа называются взаимно простыми, если они не имеют ни одного общего делителя кроме единицы.

Теорема 1.1. Основная теорема арифметики. Любое целое положительное число может быть представлено в виде произведения простых чисел, причем единственным образом.

Определение 1.6. Пусть дано целое число $N \geq 1$. Значение функции Эйлера $\varphi(N)$ равно количеству чисел ряда 1, 2, 3, …, $N-1$, взаимно простых с N.

Рассмотрим пример: $\varphi(10)$. Сформируем числовой ряд: 1, 2, 3, 4, 5, 6, 7, 8, 9. Проверим числа ряда на взаимную простоту с 10 и получим следующий результат: **1**, 2, **3**, 4, 5, 6, **7**, 8, **9**. Выделены числа взаимно простые с 10. Таким образом - $\varphi(10)$=4.

Теорема 1.2. Пусть p - простое число, тогда $\varphi(p) = p-1$.

Рассмотренную теорему примем без доказательства.

Теорема 1.3. Пусть p и q - простые числа, причем $p \neq q$. Тогда $\varphi(pq) = (p-1)(q-1)$.

□ В числовом ряду 1, 2, …, $(pq-1)$ не взаимно простыми с pq будут числа:

$$p,2p,3p,...,(q-1)p \text{ и } q,2q,3q,...,(p-1)q.$$

Всего таких чисел будет $(q-1)+(p-1)$. Следовательно, количество чисел, взаимно простых с pq будет

$$pq-1-(p-1)-(q-1)=pq-q-p+1=(p-1)(q-1). \blacksquare$$

Теорема 1.4. Теорема Ферма. Пусть p - простое число и $0<a<p$. Тогда

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

Рассмотренную теорему примем без доказательства и рассмотрим поясняющий пример. Пусть p=13, a=2. Тогда:

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

Теорема 1.5. Теорема Эйлера. Пусть a и b - взаимно простые числа. Тогда

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

Данную теорему примем без доказательства, но рассмотрим поясняющий пример. Пусть a=5 и b=12. Тогда, учитывая, что $\varphi(12)=4$, получаем:

$$5^{\varphi(12)} \bmod 12 = 5^4 \bmod 12 = (5^2) \bmod 12 = (2^2) \bmod 12 = 1.$$

Теорема 1.6. Пусть p и q - простые числа, $p \neq q$ и k - произвольное целое число. Тогда

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

Рассмотрим пример. Пусть p=5, q=7, тогда pq=35. Функция Эйлера имеет значение $\varphi(pq)=\varphi(35)=24$. При $k=2$ и $a=9$, имеем

$$9^{2\cdot 24+1} \bmod 35 = 9.$$

Определение 1.7. Число d, удовлетворяющее выражению:

$$cd \bmod m = 1,$$

называется инверсией числа c по модулю m и обозначается $c^{-1} \bmod m$.

Данное обозначение для инверсии можно переписать и по другому:

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

Умножение на c^{-1} соответствует делению на c при вычислении по модулю m. Можно ввести отрицательные степени при вычислениях по модулю:

$$c^{-a} = (a^{-1}) = (c^{-1})^a \bmod m.$$

Вычислить инверсию числа можно с помощью обобщенного алгоритма Евклида.

Приведенных сведений из теории чисел вполне достаточно для описания основных криптографических алгоритмов.

1.3. Криптосистема RSA

Криптосистема RSA названа в честь ее разработчиков: Рона Ривеста (Ron Rivest), Ади Шамира (Adi Shamir) и Леонарда Адлемана (Leonard Adleman). Криптосистема RSA является одной из самых используемых в мире асимметричных криптосистем.

Определение 1.8. Криптосистема с открытым ключом RSA формально определяется следующим образом [6-9]:

$$RSA = (X, Y, K, x, y, k_o, k_з, N, E, D), \qquad (1.12)$$

где X - множество открытых текстов, Y - множество криптограмм, K - множество ключей, $x \in X$ - некоторый открытый текст, $y \in Y$ - некоторая криптограмма, $k_o \neq k_з$, $(k_o, k_з) \in K$ - ключи шифрования и расшифрования, удовлетворяющие условию $k_o k_з \bmod \varphi(N) = 1$, $N = pq$ - натуральное число, причем p и q - простые числа, E - функция шифрования, D - функция расшифрования.

Рассмотрим алгоритм работы криптосистемы RSA. Пусть имеется сеть, состоящая из M абонентов. Каждый m-й абонент, $m = \overline{1, M}$, сети случайно выбирает два больших простых числа p_m, q_m и затем вычисляется число $N_m = p_m q_m$. Число N_m является открытой информацией, доступной другим абонентам сети. Далее каждый абонент вычисляет функцию Эйлера $\varphi(N_m) = (p_m - 1)(q_m - 1) = \phi_m$ и выбирает число $k_{om} < \phi_m$, взаимно простое с ϕ_m, а затем по обобщенному алгоритму Евклида находит число $k_{зm}$, такое что:

$$k_{om} k_{зm} \bmod \phi_m = 1,\ m = \overline{1, M}.$$

Пара $\langle N_m, k_{om} \rangle$ является открытым ключом криптосистемы, а число $k_{зm}$ представляет собой закрытый ключ и хранится абонентом в тайне.

Рассмотренные действия являются подготовительными. Будем считать, что абонент m хочет передать абоненту $m+1$ сообщение x. Сообщение x должно удовлетворять условию $x < (N_{m+1})$.

Абонент m осуществляет шифрование в соответствии с выражением:

$$y = x^{k_{om+1}} \bmod N_{m+1}, \qquad (1.13)$$

при этом, как видно, используется открытый ключ абонента $m+1$.

Абонент $m+1$, получивший криптограмму y, расшифровывает ее:

$$x = y^{k_{зm+1}} \bmod N_{m+1}. \qquad (1.14)$$

Докажем справедливость (1.14).

□ $x = y^{k_{зm+1}} \bmod N_{m+1} = (x^{k_{om+1}})^{k_{зm+1}} \bmod N_{m+1} =$

$$= x^{k_{om+1} \cdot k_{зm+1}} \bmod N_{m+1}.$$

Равенство $k_{om}k_{зm} \bmod \phi_m$ означает, что для некоторого c справедливо равенство:

$$k_{om+1}k_{зm+1} = c\phi_{m+1} + 1.$$

Согласно теореме 1.3

$$\varphi(N_{m+1}) = (p_{m+1} - 1)(q_{m+1} - 1) = \phi_{m+1}.$$

На основании этого, и следуя теореме 1.6, имеем:

$$x^{c \cdot \phi_{m+1}+1} \bmod N_{m+1} = x. \blacksquare$$

Из вышесказанного можно сделать следующие выводы:

1) протокол криптосистемы RSA шифрует и расшифровывает информацию корректно;

2) злоумышленник, перехватывающий криптограммы и знающий все открытые ключи, не сможет найти исходное сообщение при больших p и q.

Действительно, злоумышленник знает только открытый ключ $\langle N_m, k_{om} \rangle$. Для того чтобы найти $k_{зm}$, злоумышленник должен знать значение функции Эйлера $\phi_m = (p_m - 1)(q_m - 1)$, а для этого следует определить параметры p_m и q_m, разложив N_m на множители, т.е. решить задачу факторизации. Однако для него это задача трудновыполнимая.

С другой стороны, односторонняя функция $y = x^{k_{om}} \bmod N_m$, также применяемая в RSA, обладает «секретом», позволяющим легко вычислить обратную функцию $x = \sqrt[k_{om}]{y} \bmod N_m$, если известно разложение на множители числа $N_m = p_m q_m$. Действительно, это легко сделать, вычислив сначала $\phi_m = (p_m - 1)(q_m - 1)$, а затем $k_{зm} = k_{om}^{-1} \bmod \phi_m$. Таким образом, параметры p_m и q_m являются «секретом» или, как еще говорят, «лазейкой».

Иногда рекомендуется выбирать открытый ключ k_{om} одинаковым для всех абонентов сети, например, k_{om}=3. Это обеспечивает увеличение скорости шифрования.

Данная криптосистема обладает одним существенным недостатком. Рассмотрим схему на рис. 1.1. Злоумышленник, под видом абонента $m+1$, либо самостоятельно инициирует обмен информацией между абонентами m и $m+1$, либо «встраивается» между абонентами в процессе обмена ими информацией. Абонент m шифрует сообщение для передачи его абоненту $m+1$, используя при этом «подставленные» злоумышленником параметры $\langle N_{зл}, k_{озл} \rangle$. Перехваченная криптограмма y_m расшифровывается злоумышленником с помощью его закрытого ключа $k_{ззл}$. Полученный открытый текст x_m злоумышленник снова зашифровывает, используя

параметры $\langle N_{m+1}, k_{om+1} \rangle$ абонента $m+1$. Абонент $m+1$ расшифровывает полученную криптограмму, используя закрытый ключ $k_{зm+1}$. Затем абонент $m+1$ формирует ответное сообщение, которое аналогично перехватывается злоумышленником. Таким образом, злоумышленник, находясь «посередине» между абонентами, читает передаваемые ими сообщения. Избежать такой ситуации можно за счет использования более сложных протоколов, например, следующего.

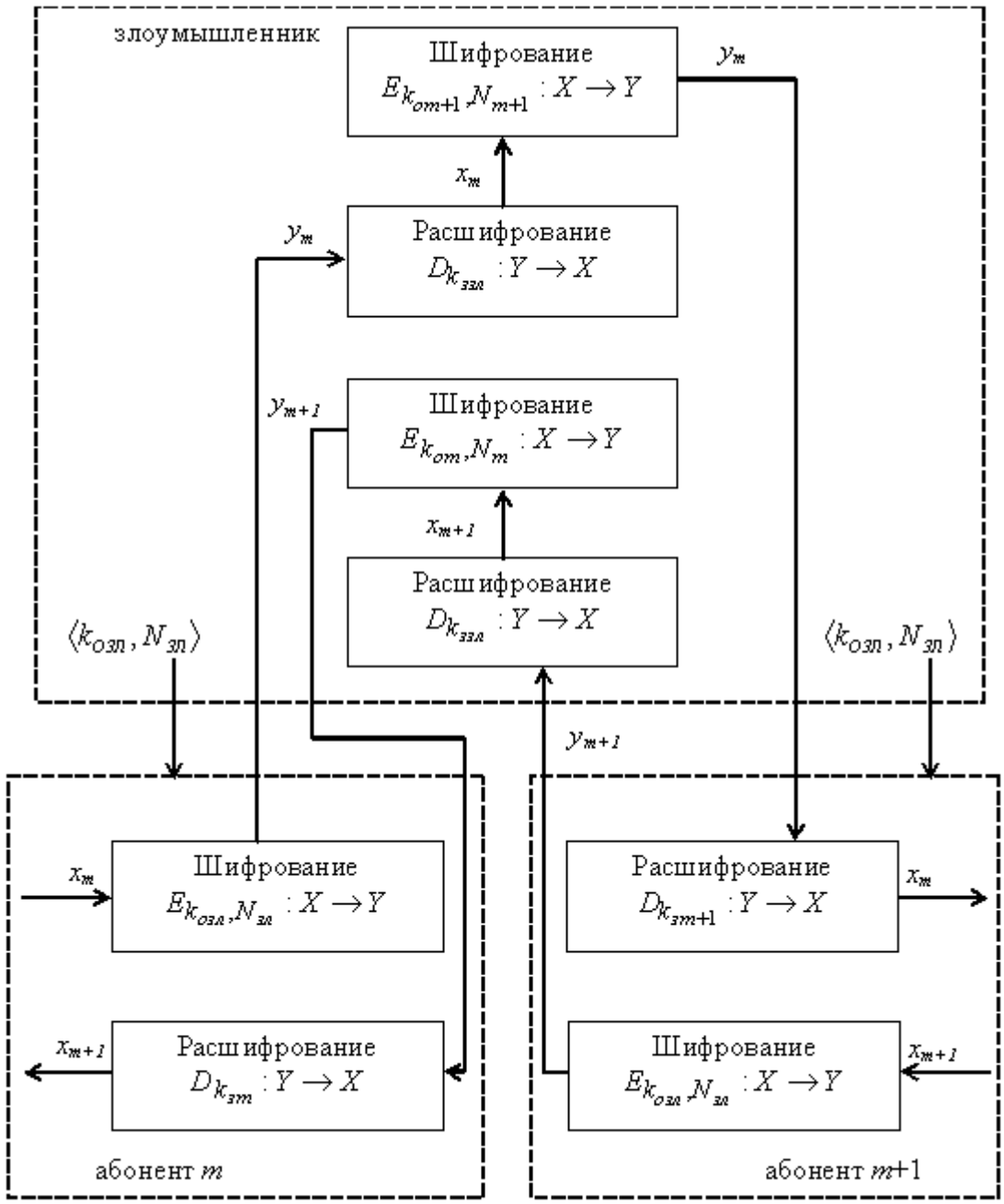

Рис. 1.1. Схема «активный злоумышленник посередине»

Пусть абонент m хочет передать сообщение абоненту $m+1$. Сначала абонент m вычисляет:

$$y = x^{k_{зт}} \bmod N_m.$$

Затем абонент m вычисляет

$$w = y^{k_{от+1}} \bmod N_{m+1}$$

и передает его абоненту $m+1$. Абонент $m+1$, получив криптограмму, последовательно вычисляет:

$$u = w^{k_{зт+1}} \bmod N_{m+1} \text{ и } x = u^{k_{от}} \bmod N_m.$$

При таком протоколе злоумышленник уже не может реализовать схему перехвата сообщений, представленную на рис. 1.1.

Рассмотрим пример шифрования при помощи алгоритма криптосистемы RSA. Пусть открытое сообщение, подлежащее шифрованию x=15. Сеть состоит из двух абонентов. Параметры второго абонента, которому адресуется шифрованное сообщение следующие: p_2=3, q_2=11, $N_2 = p_2 q_2$ =33, k_{o2}=3 (число 3 взаимно простое с $\varphi_2(33) = \phi_2 = 20$). Закрытый ключ, определенный с помощью алгоритма Евклида, $k_{з2}$=7 $(3 \cdot 7 \bmod 20 = 1)$. Шифруем открытый текст:

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

Расшифруем сообщение:

$$x = 9^7 \bmod 33 = (9^2)^2 \cdot 9^2 \cdot 9 \bmod 33 = 15^2 \cdot 15 \cdot 9 \bmod 33 = 15.$$

1.4. Криптосистема Шамира

Криптосистема, предложенная Ади Шамиром, была первой криптосистемой с открытым ключом.

Определение 1.9. Криптосистема Шамира формально определяется следующим образом [6-9]:

$$Shamir = (X, Y, K, x, y, k_o, k_з, p, F), \qquad (1.15)$$

где X - множество открытых текстов, Y - множество криптограмм, K - множество ключей, $x \in X$ - некоторый открытый текст, $y \in Y$ - некоторая криптограмма, $k_o \neq k_з$, $(k_o, k_з) \in K$ - ключи шифрования, удовлетворяющие условию $k_o k_з \bmod (p-1) = 1$, p - большое простое число, F - криптографическая функция.

Пусть два абонента сети m и $m+1$, $m = \overline{1, M}$, собираются обменяться шифрованными сообщениями, используя открытый канал связи. Абонент m выбирает большое простое число p и передает его абоненту $m+1$, а затем выбирает два числа $k_{от}$ и $k_{зт}$ такие, что

$$k_{om}k_{зт} \bmod (p-1) = 1.$$

Эти числа абонент m держит в секрете и передавать не будет. Абонент $m+1$ тоже выбирает два числа k_{om+1} и $k_{зт+1}$ такие, что

$$k_{om+1}k_{зт+1} \bmod (p-1) = 1$$

и держит их в секрете.

После этого абонент m передает сообщение, используя трехступенчатый протокол. При передаче сообщения проверяется условие $x < p$. Если условие $x < p$ выполняется, то сообщение сразу зашифровывается и передается по каналу связи. Если условие $x < p$ не выполняется, то сообщение представляется в виде блоков $x = (x_1, x_2, ..., x_i, ..., x_n)$, причем $x_i < p$. Затем каждый блок зашифровывается и передается по каналу связи. Для обеспечения криптостойкости абоненты выбираю для каждого $i-$го блока свою пару $\langle k_{om}, k_{зт} \rangle_i$ и $\langle k_{om+1}, k_{зт+1} \rangle_i$.

Дадим описание трехступенчатого протокола (см. рис. 1.2.). Абонент m вычисляет число

$$x_1 = x^{k_{зт}} \bmod p.$$

Абонент $m+1$, получив x_1, вычисляет

$$x_2 = x_1^{k_{зт+1}} \bmod p$$

и передает его абоненту m. Абонент m вычисляет следующее число

$$x_3 = x_2^{k_{om}} \bmod p$$

и передает его абоненту $m+1$. Абонент $m+1$, получив x_3, вычисляет

$$x_4 = x_3^{k_{om+1}} \bmod p = x,$$

которое является передаваемым исходным сообщением.

□ Для доказательства корректности протокола заметим, что любое целое число $z \geq 0$ может быть представлено в виде:

$$z = k(p-1) + r, \text{ где } r = z \bmod (p-1).$$

На основании теоремы 1.4 (теоремы Ферма) можно записать:

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

Тогда:

$$x_4 = x_3^{k_{om+1}} \bmod p = (x_2^{k_{om}})^{k_{om+1}} \bmod p = (x_1^{k_{зт+1}})^{k_{om}k_{om+1}} \bmod p =$$

$$= (x^{k_{зт}})^{k_{зт+1}k_{om}k_{om+1}} \bmod p = x^{k_{зт}k_{зт+1}k_{om}k_{om+1}} \bmod p =$$

$$= x^{k_{зт}k_{зт+1}k_{om}k_{om+1} \bmod (p-1)} \bmod p = \left| \begin{matrix} k_{om}k_{зт} \bmod (p-1) = 1, \\ k_{om+1}k_{зт+1} \bmod (p-1) = 1 \end{matrix} \right| = x. \blacksquare$$

Злоумышленник не может прочесть переданное сообщение. Действительно, злоумышленник, перехватывая передаваемые сообщения

x_1, x_2, x_3, должен решать задачу дискретного логарифмирования, что при больших p невозможно.

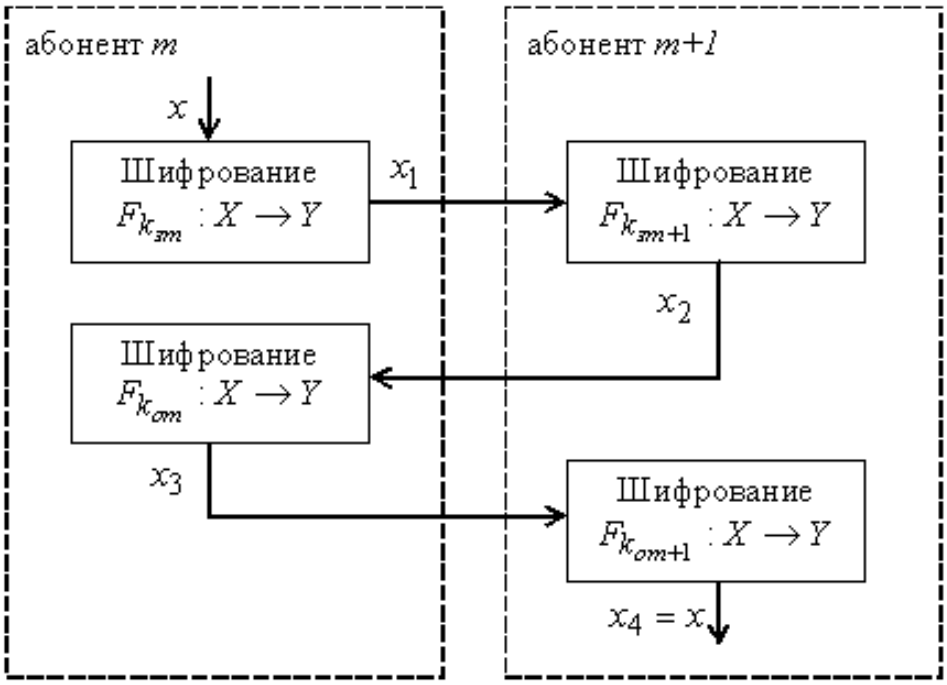

Рис. 1.2. Схема алгоритма Шамира

Рассмотрим пример шифрования при помощи алгоритма Шамира. Пусть открытое сообщение, подлежащее шифрованию, x=10. Сеть состоит из двух абонентов. Первый абонент выбирает $p = 23$ и параметры k_{o1}=19, $k_{з1}$=7 ($7 \cdot 19 \bmod 22 = 1$). Аналогично второй абонент выбирает параметры k_{o2}=9, $k_{з2}$=5 ($5 \cdot 9 \bmod 22 = 1$). Протокол Шамира:

$$x_1 = 10^7 \bmod 23 = 14,$$
$$x_2 = 14^5 \bmod 23 = 15,$$
$$x_3 = 15^{19} \bmod 23 = 19,$$
$$x_4 = 19^9 \bmod 23 = 10.$$

Очевидный недостаток криптосистемы Шамира заключается в необходимости реализации трехступенчатого протокола. Это влечет за собой увеличение времени на передачу сообщения, а также увеличивает вероятность искажения принятого сообщения из-за помех в канале передачи информации.

1.5. Криптосистема Эль Гамаля

Определение 1.10. Криптосистема с открытым ключом Эль Гамаля формально определяется следующим образом [6-9]:

$$ElGamal = \langle X, Y, K, x, y, k_o, k_з, p, g, E, D \rangle, \quad (1.16)$$

где X - множество открытых текстов, Y - множество криптограмм, K - множество ключей, $x \in X$ - некоторый открытый текст, $y \in Y$ - некоторая криптограмма, $k_o \neq k_з$, $\langle k_o, k_з \rangle \in K$ - ключи шифрования и расшифрования, удовлетворяющие условию: $k_o = g^{k_з} \bmod p$, p - большое простое число, g - число такое, что различные степени числа g суть различные числа по модулю p, E - функция шифрования, D - функция расшифрования.

Для всей группы $m = \overline{1, M}$ абонентов сети выбирается некоторое большое простое число p и число g. Выбор числа g может оказаться трудной задачей при произвольно заданном числе p, т.к. это связано с разложением на простые множители числа $p-1$. Дело в том, что для обеспечения высокой стойкости алгоритма шифрования число $p-1$ должно обязательно содержать большой простой множитель, в противном случае, с помощью алгоритма Полига-Хеллмана быстро вычисляется дискретный логарифм. В связи с этим простое число p выбирается таким, чтобы выполнялось равенство:

$$p = 2q + 1, \quad (1.17)$$

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

$$1 < g < p-1, \; g^q \bmod p \neq 1. \quad (1.18)$$

Числа p и g передаются абонентам сети в открытом виде. Затем каждый абонент сети выбирает секретный ключ $k_{зт}$, $m = \overline{1, M}$, удовлетворяющее условию: $1 < k_{зт} < p-1$, и вычисляет открытый ключ:

$$k_{от} = g^{k_{зт}} \bmod p, \; m = \overline{1, M}. \quad (1.19)$$

Пусть абонент m хочет передать абоненту $m+1$ сообщение x, при этом необходимо выполнение условия: $x < p$.

Шифрование исходного сообщения x происходит следующим образом. Абонент m формирует случайное число c, причем $1 \le c \le p-2$ (эту операцию выполняют все абоненты сети), и вычисляет числа

$$r = g^c \bmod p, \quad (1.20)$$

$$y = x \cdot k_{от+1}^c \bmod p, \quad (1.21)$$

а затем передает пару $\langle r, y \rangle$ абоненту $m+1$.

Абонент $m+1$, получив криптограмму $\langle r, y \rangle$, вычисляет исходный текст

$$x = y \cdot r^{p-1-k_{зт+1}} \bmod p. \quad (1.22)$$

□ Подставим в (1.22) выражение (1.21) и получим

$$x = x \cdot k_{от+1}^c \cdot r^{p-1-k_{зт+1}} \bmod p.$$

Теперь в полученное выражение подставим (1.19) и (1.20)

$$x = x \cdot \left(g^{k_{зт+1}}\right)^{c} \cdot \left(c^{p-1-k_{зт+1}}\right) \bmod p =$$

$$= x \cdot g^{k_{зт+1}c + c(p-1) - ck_{зт+1}} \bmod p = x \cdot g^{c(p-1)} \bmod p.$$

По теореме 1.4 (теореме Ферма) $a^{p-1} \bmod p = 1$, тогда

$$x \cdot g^{c(p-1)} \bmod p = x \cdot 1^{c} \bmod p = x. \blacksquare$$

Злоумышленник должен определить закрытый ключ $k_{зт}$ или отыскать число c, но для этого ему необходимо решить задачу дискретного логарифмирования.

Особенностью криптосистемы Эль Гамаля является то, что объем передаваемой криптограммы в два раза превышает объем исходного сообщения. Это объясняется тем, что для вычисления криптограммы требуется выполнить операции (1.18) и (1.19). Следствием этого является большее, по сравнению с алгоритмом RSA, время шифрования и больший объем вычислений.

Рассмотрим пример шифрования при помощи алгоритма Эль Гамаля. Пусть требуется передать исходный открытый текст $x = 15$ от абонента m к абоненту $m+1$. Выберем в соответствии с (1.17) параметр $p = 23$ ($q = 11$). Определим параметр g. Возьмем $g = 3$, проверим: $3^{11} \bmod 23 = 1$ и, значит, такое число g не подходит. Возьмем $g = 5$, проверим: $5^{11} \bmod 23 = 22$, такое число g подходит. Таким образом, параметрами криптосистемы являются $p = 23$ и $g = 5$. Пусть абонент $m+1$ выбрал закрытый ключ $k_{зт+1} = 13$ и вычислил открытый ключ $k_{от+1} = 5^{13} \bmod 23 = 21$.

Абонент m выбирает случайное число $c = 7$ и вычисляет

$$r = 5^{7} \bmod 23 = 17,$$

$$y = 15 \cdot 21^{7} \bmod 23 = 15 \cdot 10 \bmod 23 = 12.$$

Абонент m пересылает абоненту $m+1$ зашифрованное сообщение $\langle 17,12 \rangle$, которое абонент $m+1$ расшифровывает

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

1.6. Криптосистема Рабина

Определение 1.11. Криптосистема с открытым ключом Рабина формально определяется следующим образом [6]:

$$Rabin = (X, Y, x, y, N, p, q, E, D), \qquad (1.23)$$

где X - множество открытых текстов, Y - множество криптограмм, $x \in X$ - некоторый открытый текст, $y \in Y$ - некоторая криптограмма,

$N = pq$ - RSA-модуль (является открытым ключом криптосистемы), где p и q - секретные ключи криптосистемы, E - функция шифрования, D - функция расшифрования.

В криптосистеме Рабина используется RSA-модуль $N = pq$, в котором числа p и q сравнимы с числом 3 по модулю 4:

$$p \equiv 3 \bmod 4,\ q \equiv 3 \bmod 4,$$

что обеспечивает при знании разложения модуля возможность выполнения операции извлечения квадратного корня из квадратичных вычетов по модулю N.

Шифрование осуществляется по формуле

$$y = x^2 \bmod N,$$

причем должно выполняться условие: $x < N$.

Процедура расшифрования состоит в извлечении квадратного корня из криптограммы. Предварительно вычисляются корни из y по модулям p и q:

$$w_{p_1} = y^{\frac{p+1}{4}} \bmod p,\ w_{q_1} = y^{\frac{q+1}{4}} \bmod q;$$

$$w_{p_2} = p - w_{p_1} = p - y^{\frac{p+1}{4}} \bmod p,$$

$$w_{q_2} = q - w_{q_1} = q - y^{\frac{q+1}{4}} \bmod q.$$

На основе полученных значений вычисляются четыре возможных корня из y по модулю N:

$$x_1 = (aw_{p_1} + bw_{q_1}) \bmod N,$$
$$x_2 = (aw_{p_1} + bw_{q_2}) \bmod N,$$
$$x_3 = (aw_{p_2} + bw_{q_1}) \bmod N,$$
$$x_4 = (aw_{p_2} + bw_{q_2}) \bmod N,$$

где $a = q(q^{-1} \bmod p)$ и $b = p(p^{-1} \bmod q)$.

Как видно из представленных формул расшифрование неоднозначно, что является недостатком криптосистемы Рабина. Для обеспечения однозначности перед шифрованием к исходному открытому сообщению можно присоединить некоторую заранее оговоренную метку.

1.7. Криптосистемы на эллиптических кривых

Криптосистемы на эллиптических кривых – одно из новых направлений в криптографии. Эллиптические кривые давно изучаются математикой, но их использование в криптографии впервые было предложено Коблицом и

Миллером в 1985 году. Прошедшие два с половиной десятилетия подтвердили эффективность этих криптосистем и привели к открытию множества их реализации. Основным достоинством криптосистем на эллиптических кривых является их более высокая стойкость по сравнению с традиционными асимметричными криптосистемами, при равных вычислительных затратах.

Детальное изучение эллиптических кривых требует знаний высшей алгебры, в особенности алгебраической геометрии. Рассмотрим математические основы теории эллиптических кривых, достаточных для понимания принципов построения и работы криптосистем [6-9].

Математические основы криптосистем на эллиптических кривых

Определение 1.12. Кривая третьего порядка E, задаваемая уравнением вида:

$$E: y^2 = x^3 + ax + b, \qquad (1.24)$$

называется эллиптической кривой.

Наиболее общее определение эллиптической кривой дает уравнение Вейерштрасса [7].

Поскольку $y = \pm\sqrt{x^3 + ax + b}$, график кривой симметричен относительно оси абсцисс. Для нахождения точек пересечения кривой E с осью абсцисс необходимо решить кубическое уравнение

$$x^3 + ax + b = 0. \qquad (1.25)$$

Это можно сделать с помощью известных формул Кордано. Дискриминант (1.25) имеет вид

$$D = \left(\frac{a}{3}\right)^3 + \left(\frac{b}{2}\right)^2.$$

Если $D < 0$, то (1.25) имеет три различных действительных корня α, β, γ; если $D = 0$, то (1.25) имеет два различных действительных корня α, β; если $D > 0$, то (1.25) имеет один действительный корень α и два комплексно сопряженных корня (см. рис.1.3).

Эллиптическая кривая, для которой $D = 0$, называется **сингулярной.** Точка $(\beta, 0)$ называется **точкой сингулярности**. Сингулярные эллиптические кривые в криптографии не используются, т.к. для них проблема дискретного логарифма эффективно решается [6,7]. Таким образом, для криптографических эллиптических кривых требуется выполнение условия

$$D = 4a^3 + 27b^2 \neq 0. \qquad (1.26)$$

В дальнейшем будем считать, что эллиптическая кривая E задана уравнением (1.24) с ограничением на коэффициенты (1.26).

Пусть на эллиптической кривой заданы две точки $S(x_s, y_s)$ и $T(x_t, y_t)$ (см. рис. 1.4). Проведенная через эти точки прямая пересечет кривую E в

третьей точке $R'(x_r, y'_r)$. Получим точку $R(x_r, y_r = -y'_r)$ путем изменения знака ординаты точки $R'(x_r, y'_r)$.

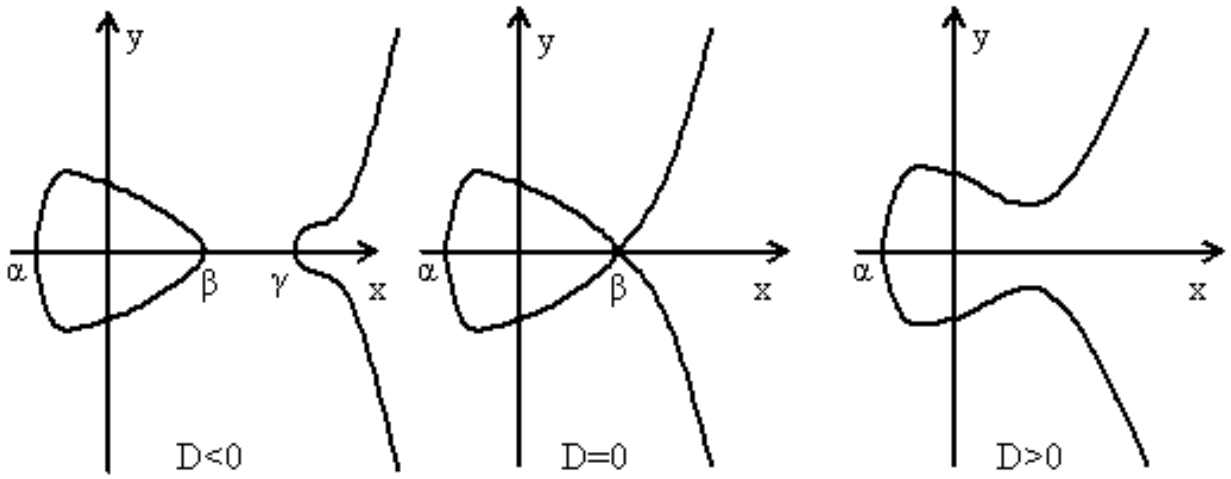

Рис. 1.3. Эллиптическая кривая

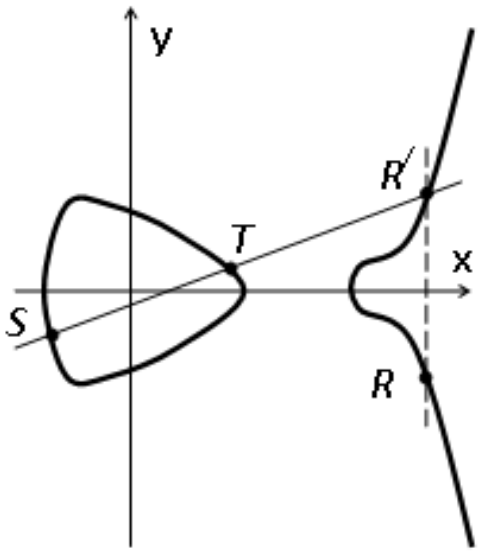

Рис. 1.4. Композиция точек на эллиптической кривой

Описанная операция имеет вид:

$$R = S + T, \qquad (1.27)$$

и называется **композицией точек** или **сложением точек** на эллиптической кривой.

Из (1.27) следует, что если точка $S(x_s, y_s) \in E$, то точка с измененным знаком ординаты $S(x_s, -y_s) = -S \in E$. Через эти точки проходит прямая параллельная оси ординат, которая, как считают, пересечет кривую E в бесконечно удаленной точке O, причем $S + (-S) = \mathrm{O}$. Как следствие из этого

справедливо утверждение, что $S + \mathrm{O} = \mathrm{O} + S = S$. Таким образом, точка O играет роль нуля в операциях на эллиптической кривой.

Представим, что точки $S(x_s, y_s)$ и $T(x_t, y_t)$ сближаются друг с другом и, наконец, сливаются в одну точку $S = T = (x_s, y_s)$ (см. рис. 1.5). Тогда композиция точек $R(x_r, y_r) = S + T = S + S$ будет получена путем проведения касательной в точке S и отражения ее пересечения с кривой в точке $R'(x_r, y'_r)$ относительно оси абсцисс:

$$R = S + S = [2]S. \qquad (1.28)$$

Операция (1.28) называется **удвоением точки**.

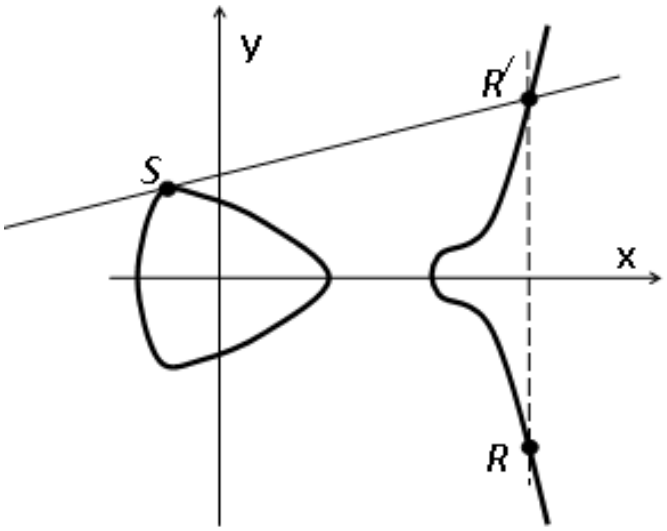

Рис. 1.5. Удвоение точки на эллиптической кривой

Определим координаты результирующей точки $R(x_r, y_r)$ на основе координат точек $S(x_s, y_s)$ и $T(x_t, y_t)$.

Рассмотрим случай, когда $S \neq \pm T$. Обозначим через k угловой коэффициент прямой, проходящей через S и T. Очевидно, что

$$k = \frac{y_t - y_s}{x_t - x_s}.$$

Тогда уравнение прямой будет иметь вид

$$y - y_s = k(x - x_s),$$

откуда

$$y = y_s + k(x - x_s). \qquad (1.29)$$

Если подставить (1.29) в (1.24), получим

$$(y_s + k(x - x_s))^2 = x^3 + ax + b.$$

Проведя необходимые преобразования, получаем кубическое уравнение

$$x^3 - k^2x^2 + a'x + b' = 0,$$

где $a' = a - 2k(y_s - x_s)$, $b' = b - y_s^2 - k(y_s x_s + kx_s^2)$.

По теореме Виета для кубических уравнений [6] имеем

$$x_1 + x_2 + x_3 = k^2,$$

откуда

$$x_3 = k^2 - x_1 - x_2. \qquad (1.30)$$

Подставив найденное значение x_3 в уравнение прямой (1.29), определим ординату точки $R'(x_r, y'_r)$:

$$y'_r = y_s + k(x_3 - x_s) = y_s + k(x_r - x_s),\ x_3 = x_r.$$

Изменив знак ординаты точки $R'(x_r, y'_r)$ получим

$$y_r = k(x_s - x_r) - y_s. \qquad (1.31)$$

Таким образом, получены координаты точки $R = S + T$.

Рассмотрим случай, когда $S = T$ и результирующая точка $R = 2S$. Дифференцируя обе части (1.24) по x, получаем

$$2yy' = 3x^2 + a.$$

Угловой коэффициент касательной равен значению производной в точке S

$$k = \frac{3x_s^2 + a}{2y_s}.$$

Дальнейшие рассуждения аналогичны рассмотренному выше случаю, координаты точки R определяются по тем же формулам (1.30) и (1.31).

Если ордината точки S равна нулю, то касательная проходит параллельно оси ординат и $2S = \mathrm{O}$.

Основные свойства точек на эллиптической кривой:

1) $S + T = T + S$;

2) $S + (T + V) = (S + T) + V$;

3) $S + \mathrm{O} = \mathrm{O} + S = S$;

4) $S + (-S) = \mathrm{O}$.

Как видно перечисленные свойства точек на эллиптической кривой совпадают со свойствами целых чисел при использовании операции сложения. Рассмотрим еще несколько свойств композиции точек

1) $mS = \underbrace{S + S + \ldots + S}_{m}$, для любого целого числа m;

2) $-mS = -\underbrace{(S + S + \ldots + S)}_{m}$;

3) $0S = \mathrm{O}$.

Из рассмотренного выше можно сделать следующий вывод. При вычислении композиции точек на эллиптической кривой используются только операции сложения, вычитания, умножения и деления. Это значит, что все приведенные выше выражения сохраняются при вычислении с целыми числами по модулю простого числа p. В этом случае сложение и умножение чисел выполняется по $\bmod p$, разность чисел $(\nu - \vartheta)$ вычисляется как $\nu + (p - \vartheta) \bmod p$, а деление $\frac{\nu}{\vartheta}$ выполняется как $\nu \cdot \vartheta^{-1} \bmod p$.

В результате получаем кривую:

$$E_p : y^2 = x^3 + ax + b \bmod p. \qquad (1.32)$$

В уравнении (1.32) коэффициенты a, b принимают целочисленные значения, а все вычисления выполняются по $\bmod p$. В соответствии с (1.26) на a, b налагается условие

$$(4a^3 + 27b^2) \bmod p \neq 0. \qquad (1.33)$$

Множество $E_p(a, b)$ состоит из точек (x, y), $0 < x, y < p$, удовлетворяющих (1.32) и точки в бесконечности O. Количество точек в $E_p(a, b)$ обозначается $\#E_p(a, b)$. Эта величина имеет важное значение для эллиптической криптографии.

Рассмотрим некоторые свойства множества точек $E_p(a, b)$. Очевидно, это множество конечно, т.к. в него входят только точки с целочисленными координатами $0 \le x, y < p$. Существует прямая аналогия между $E_p(a, b)$ и множеством степеней целых чисел, вычисляемых по модулю p. Действительно, для организации криптосистемы выбирается большое простое число p и некоторое число g, $1 \le g < p - 1$, такое, что все числа из множества $\{1, 2, ..., p - 1\}$ могут быть представлены как различные степени $g \bmod p$. Так и $E_p(a, b)$ имеет генератор, т.е. такую точку G, что ряд G, $2G$, $3G$, ..., nG, где $n = \#E_p(a, b)$, содержит все точки множества $E_p(a, b)$, причем $nG = \mathrm{O}$.

Число точек на кривой, при надлежащем выборе параметров a, b и p, может быть простым числом, $\#E_p(a, b) = q$. В этом случае любая точка (кроме точки O) является генератором всего множества точек. Такая кривая предпочтительна и может быть вычислена за приемлемое время. Если по каким-то причинам такую кривую не удалось отыскать и $\#E_p(a, b) = hq$, то в $E_p(a, b)$ существует подмножество из q точек, генератором которого может служить любая точка $G \neq \mathrm{O}$, такая что $qG = \mathrm{O}$.

Основная криптографическая операция на эллиптической кривой - m-кратная композиция, т.е. вычисление

$$T = [m]S = \underbrace{S + S + S + \ldots + S}_{m}.$$

Эта операция выполняется очень эффективно и требует не более $2\log m$ композиций точек. Подходы к ее реализации те же, что и возведение в степень. Например, чтобы получить точку $T = [21]S$, вычисляем $[2]S$, $[4]S$, $[8]S$, $[16]S$, каждый раз удваивая предыдущую точку, и складываем $S + [4]S + [16]S = T$.

Обратная задача называется **дискретным логарифмом на эллиптической кривой** и формулируется следующим образом. Зная точки S и T, требуется найти такое число m, что $[m]S = T$. Эта задача оказывается очень трудной.

Выбор параметров эллиптической кривой. Существуют две стратегии выбора эллиптической кривой. Первая - случайная стратегия. Эта стратегия отличается более высокой стойкостью криптосистемы, но с вычислительной точки зрения менее эффективна. Вторая стратегия, не рассматриваемая в данном пособии, заключается в конструировании эллиптической кривой с заданными свойствами. С вычислительной точки зрения такая стратегия более эффективна, но получаемые кривые фактически выбираются из относительно небольшого класса и возможно, что со временем будут получены алгоритмы, способные эффективно решать задачу дискретного логарифмирования на эллиптической кривой.

Процесс формирования случайной эллиптической кривой состоит в следующем.

1. Выбирается случайное число p.

Битовая длина числа p, $t = \lfloor \log p \rfloor + 1$ должна быть такой, чтобы сделать невозможным применение общих методов нахождения логарифмов на эллиптической кривой. Величина $t = 160$ бит в настоящее время вполне приемлема. С другой стороны, криптосистема на эллиптической кривой должна быть не менее стойкой, чем блочная криптосистема AES. Стойкость криптосистемы AES определяется длиной ее ключа (128, 196, 256 бит), а для криптосистем на эллиптических кривых при равной криптостойкости величина модуля должна быть в два раза больше, т.е. 256, 392, 512 бит соответственно [7,9].

2. Выбираются числа a, b такие, что $a, b \neq 0 \bmod p$ и $4a^3 + 27b^2 \neq 0 \bmod p$.

При вычислении композиции точек на эллиптической кривой параметр b нигде не фигурирует, поэтому b выбирают случайно, а число a принимают равным небольшому целому числу, например, $a = -3$. Это позволяет снизить вычислительные затраты.

3. Определяют число точек на эллиптической кривой $n = \#E_p(a, b)$.

Это самый трудоемкий этап. Базовый алгоритм (алгоритм Схоуфа) определения числа точек эллиптической кривой подробно рассмотрен в [7,9].

Важно, чтобы n имело большой простой делитель q, а лучше всего само было простым числом $n = q$.

Если n разлагается на маленькие множители, то в $E_p(a,b)$ существует много небольших подмножеств со своими генераторами, и алгоритм Полига-Хеллмана быстро вычисляет алгоритм на кривой через логарифмы этих подмножеств.

Если поиск кривой с $n = q$ занимает слишком много времени, то допускают, чтобы $n = hq$, где h - небольшое число.

Необходимо отметить, что стойкость криптосистемы на эллиптической кривой определяется не модулем p, а числом элементов q в подмножестве точек кривой. Однако если h - небольшое число, то q является величиной того же порядка что и p.

Если n не соответствует предъявленным требованиям, то необходимо вернуться ко второму этапу, т.е. к выбору чисел a, b.

4. Проверка выполнения неравенств $(p^k - 1) \bmod q \neq 0$ для всех k, $0 < k < 32$. Если неравенство не выполняется, то необходимо вернуться ко второму этапу.

Эта проверка предотвращает возможность MOV-атаки, названной в честь ее авторов - Menezes, Okamoto, Vanstone, а также исключает из рассмотрения суперсингулярные эллиптические кривые и кривые с $\#E_p(a,b) = p - 1$.

5. Проверка неравенства $q \neq p$. Если неравенство не выполняется, то необходимо вернуться к этапу 2. Данная проверка необходима, т.к. при выполнении условия $q = p$ получаются так называемые **аномальные эллиптические кривые,** для которых существуют методы эффективного решения задачи дискретного логарифмирования на кривой.

6. Определение точки G. На данном этапе искомая эллиптическая кривая найдена, определены параметры p, a, b, число точек n и размер подмножества точек q.

Если $q = n$, то любая точка, кроме точки O, является точкой-генератором G.

Если $q < n$, то выбирается случайная точка G' и проверяется условие $G = \left[\frac{n}{q}\right] G' \neq \mathrm{O}$. Если оно выполняется, то $G = G'$, в противном случае берется другая точка.

Криптосистема Эль Гамаля на эллиптической кривой. Любая криптосистема, построенная на основе дискретного логарифмирования, может быть перенесена на эллиптические кривые.

Основной принцип построения криптосистемы состоит в замене операции $y = g^x \bmod p$ на $Y = [x]G \bmod p$. Отличие заключается в том, что y - это число, а Y - это точка, поэтому требуется переходить от точки к числу. Обычно прибегают к простому способу – используют абсциссу точки.

На эллиптических кривых можно построить и аналог системы RSA. Однако в этом случае не получается выигрыша, т.к. длина модуля остается такой же как и в первоначальном варианте системы RSA, чтобы невозможно было разложить n на простые множители.

Рассмотрим технику использования эллиптических кривых на примере криптосистемы Эль Гамаля. Выбирается общая эллиптическая кривая $E_p(a, b)$ и точка G на ней такая, что G, $[2]G$, $[3]G$, …, $[q]G$ суть различные точки и $[q]G = \mathrm{O}$ для некоторого простого числа q.

Каждый абонент сети выбирает случайное число $k_з$, $0 < k_з < q$, которое является секретным ключом, и вычисляет точку

$$K = [k_з]G,$$

являющуюся открытым ключом. Параметры эллиптической кривой и открытые ключи передаются всем пользователям сети.

Абонент m желает передать сообщение абоненту $m+1$. Открытый текст должен удовлетворять условию $x < p$.

Абонент m выбирает случайное число s, $0 < s < q$. Затем вычисляет:

$$R = [s]G,$$
$$[s]K_{m+1} = P(x_{m+1}, y_{m+1})$$

и производит шифрование:

$$y = (x \cdot x_{m+1}) \bmod p.$$

Полученную криптограмму $\langle R, y \rangle$, абонент m передает абоненту $m+1$.

Абонент $m+1$ вычисляет

$$[k_{зm+1}]R = Q(x_{m+1}, y_{m+1}),$$
$$x = (y \cdot x_{m+1}^{-1}) \bmod p.$$

□ $[k_{зm+1}]R = [k_{зm+1}][s]G = [s][k_{зm+1}]G = [s]K$,

то есть $Q = P$. ■

Координата x_{m+1} точки Q для злоумышленника остается неизвестной, т.к. он не знает числа s. Для того, чтобы определить число s на основании точки R злоумышленнику требуется решить задачу дискретного логарифмирования на кривой.

Чаще всего такого рода криптосистемы используются для передачи ключа симметричной криптосистемы. В этом случае необходимо выбирать параметры кривой так, чтобы $\log q$ примерно вдвое превышал длину ключа криптосистемы.

1.8. Криптосистемы, основанные на задаче «об укладке рюкзака»

При рассмотрении односторонних функций в пункте 1.1 было отмечено, что криптосистемы, основанные на задаче «об укладке рюкзака», в настоящее время не находят применения. Однако в данном пункте мы все же рассмотрим кратко принцип построения криптосистем, основанных на использовании задачи «об укладке рюкзака». Такими криптосистемами являются криптосистемы Меркля-Хеллмана и Хора-Ривеста [2].

Выражение (1.11) представляет собой уравнение шифрования. Требуется сказать, что выбор $\mathbf{B} = (b_1, b_2, b_3, ..., b_n)$ является критическим. Например, предположим, что $\mathbf{B}$ является быстрорастущей последовательностью, для которой выполняется условие $b_i > \sum_{j=1}^{i-1} b_j$. В этом случае для данных $\mathbf{B}$, V вычислить $\mathbf{X}$ достаточно просто.

Действительно, проверяем, является ли V большим, чем последний элемент $\mathbf{B}$, и если да, то полагаем последний элемент $\mathbf{X}$ равным $x_n = 1$, вычитаем это значение из V и рекурсивно решаем меньшую проблему. Этот метод работает, т.к. $\mathbf{B}$ является быстрорастущей последовательностью.

Следовательно, выбор $\mathbf{B}$ - важная и непростая задача, ведь можно получить и не получить одностороннюю функцию. Вместе с тем, именно существование этой проблемы позволяет получить одностороннюю функцию с секретом, которую используют для построения криптосистем.

Абонент формирует открытый ключ следующим образом. Вначале он выбирает некоторую быстрорастущую последовательность $\mathbf{B}'$, затем выбирает случайное число m, с условием, что $m > \sum_{j=1}^{n} b'_j$. После этого абонент выбирает случайное целое число ϖ взаимно простое с m. Затем вычисляется последовательность

$$\mathbf{B}'' = \varpi \mathbf{B}' \bmod m.$$

После применения случайной перестановки $P(\ )$ элементов последовательности $\mathbf{B}''$ абонент получает открытый ключ $\mathbf{B}$, т.е. $\mathbf{B} = P(\mathbf{B}'')$.

Открытый ключ $\mathbf{B}$ абонент публикует в сети, а параметры $\langle m, \varpi, \mathbf{B}', P \rangle$ являются секретным (закрытым) ключом.

Когда один абонент хочет послать другому абоненту зашифрованное сообщение, он выполняет операцию $y = \mathbf{BX}$. Для злоумышленника вычисление $\mathbf{X}$ по y и публичному ключу $\mathbf{B}$ будет эквивалентно решению задачи «об укладке рюкзака» в общем случае. Легальный абонент, которому адресована

криптограмма y, применяя секретные параметры, решает задачу «об укладке рюкзака» в случае быстрорастущей последовательности.

Вначале абонент вычисляет

$$y' = \mathbf{B}'\mathbf{X} = \sum_{i=1}^{n} b_i' x_i = \varpi^{-1} \sum_{i=1}^{n} \varpi b_i' x_i = \varpi^{-1} \sum_{i=1}^{n} b_i x_i =$$
$$= \varpi^{-1} y \pmod{m}.$$

Таким образом, абонент просто умножает криптограмму на мультипликативное обратное ϖ^{-1}, а затем решает задачу «об укладке рюкзака» в случае быстрорастущей последовательности и отыскивает $\mathbf{X}$. Алгоритм решения задачи «об укладке рюкзака» в случае быстрорастущей последовательности подробно описан в [2].

В 1982 году Ади Шамир открыл атаку, позволяющую эффективно решать задачу «об укладке рюкзака». Это оказалось началом падения криптосистем, основанных на задаче «об укладке рюкзака». Единственная, используемая в настоящее время и не раскрытая криптосистема, основанная «на укладке рюкзака», – криптосистема Бен-Цион Хора [2,7].

Контрольные вопросы

1. Дайте определение криптосистемы с открытым ключом.
2. Дайте определение односторонней функции. Какие типы односторонних функций используются в ассиметричной криптографии?
3. Опишите алгоритмы шифрования и расшифрования основных типов криптосистем с открытым ключом.
4. Дайте сравнительную оценку криптосистем с открытым ключом.
5. В чем заключается различие и сходство криптосистемы Эль Гамаля и криптосистем на эллиптических кривых?
6. Назовите основные характеристики ассиметричных криптосистем.

2. Электронная подпись

Одна из важнейших проблем, решаемых с использованием асимметричных методов шифрования - проблема подтверждения авторства. Данная проблема возникает при следующих обстоятельствах:

- когда некоторый абонент m получает сообщение, предположительно от абонента $m+1$, как подтвердить, что получено сообщение именно от абонента m, а не от третьего лица;
- когда абонент m получает от абонента $m+1$ сообщение, как подтвердить, что оно не было изменено третьим лицом.

Для решения этой проблемы были разработаны алгоритмы электронной подписи. Определение электронной подписи дано федеральным законом №63-ФЗ от 6.04.2011 г.

Определение 2.1. Электронная подпись - информация в электронной форме, которая присоединена к другой информации в электронной форме (подписываемой информации) или иным образом связана с такой информацией и которая используется для определения лица, подписывающего информацию.

Во многих странах мира, в том числе в России, введены в действие стандарты на электронную подпись. В Европе и США вместо термина «электронная подпись» чаще используется термин «цифровая подпись». Оба термина означают одно и то же.

Электронная подпись (ЭП) должна обладать следующими свойствами:

1) подписать документ может только законный владелец ЭП;

2) автор ЭП не может от нее отказаться;

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

Из рассмотренных свойств можно определить злонамеренные действия, к которым относятся:

- **отказ (ренегатство)** – отправитель впоследствии отказывается от переданного сообщения;
- **фальсификация** – получатель (или третье лицо) подделывает сообщение;
- **изменение** - получатель (или третье лицо) вносит изменение в сообщение;
- **маскировка** – злоумышленник маскируется под легального пользователя.

Схема ЭП включает в себя:

- параметр безопасности n;
- пространство исходных сообщений;
- алгоритм G генерации пары ключей $(k_з, k_о)$;
- алгоритм S формирования подписи;
- алгоритм V проверки подписи.

Электронная подпись $s = S(k_з, x)$ называется **допустимой** для документа x, если она принимается алгоритмом V. **Подделкой** ЭП документа x называется нахождение злоумышленником, не имеющим секретного ключа, допустимой подписи для документа x.

Обобщенная схема ЭП имеет следующий вид (см. рис.2.1) [7-10]:

1. Отправитель А вычисляет $(k_з, k_о) = G(n)$ и посылает получателю В $k_о$.

2. Для получения подписи документа x отправитель вычисляет $s = S(k_з, x)$ и посылает $\langle x, s \rangle$ получателю.

3. Получатель вычисляет $V(x, s, k_о)$ и в зависимости от результата принимает или отвергает подпись s отправителя А.

В классической схеме ЭП предполагается, что отправитель знает содержание подписываемого документа, а получатель проверяет подлинность ЭП без какого-либо разрешения и участия отправителя А.

При формировании ЭП по классической схеме отправитель А вычисляет хеш-функцию (хеш-образ) документа $h_x = h(x)$ и при необходимости дополняет его до требуемой длины. Алгоритм вычисления хеш-функции известен всем абонентам сети. Не будем пока останавливаться на свойствах и способах вычисления хеш-функции, этот вопрос будет рассмотрен подробнее в следующем разделе. Отметим только важные для нас сейчас свойства:

1) хеш-функция обеспечивает преобразование входного массива данных любого размера в выходной массив данных (хеш) фиксированного размера;

2) практически невозможно внести изменения в входной массив данных, не изменив выходной массив данных (хеш).

Отправителю А достаточно снабдить подписью не сам документ x, а его хеш-образ h_x.

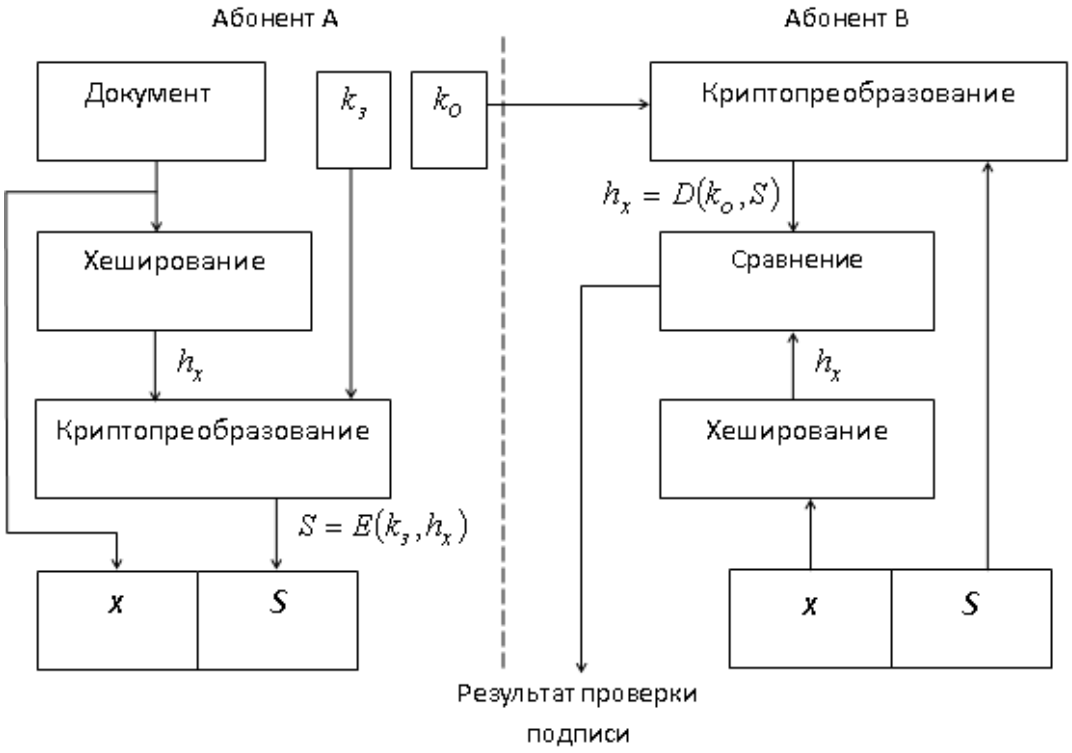

Рис. 2.1. Обобщенная схема формирования и проверки ЭП

2.1. Электронная подпись на основе криптосистемы RSA

Пусть некоторая сеть включает в себя M абонентов. Абонент m планирует подписывать документы. Вначале абонент должен сформировать параметры криптосистемы RSA (см. пункт 1.3). Для этого абонент выбирает два больших простых числа p_m, q_m и вычисляет $N_m = p_m q_m$, $\phi_m = (p_m - 1)(q_m - 1)$. Затем абонент выбирает число $k_{om} < \phi_m$, взаимно простое с ϕ_m, и находит $k_{зm} = k_{om}^{-1} \bmod \phi_m$.

Абонент публикует в сети, ассоциировав со своим именем, числа $\langle N_m, k_{om} \rangle$, а число $k_{зт}$ хранит в тайне. Числа p_m, q_m, ϕ_m в вычислениях больше не используются. На этом шаге формирование параметров криптосистемы заканчивается. Абонент готов подписывать документы.

На следующем шаге абонент вычисляет хеш-функцию (или просто - хеш) подписываемого документа $h_x = h(x_m)$. Далее абонент вычисляет число

$$s = h_x^{k_{зт}} \bmod N_m$$

которое представляет собой ЭП. Число s добавляется к документу x_m и абонент получает подписанный документ $\langle x_m, s \rangle$.

Каждый абонент сети, который знает параметры абонента m, может проверить подлинность его подписи. Для этого необходимо из подписанного документа $\langle x_m, s \rangle$ взять x_m и вычислить хеш-образ h_x. Затем вычислить число

$$\omega = s^{k_{om}} \bmod N_m$$

и проверить выполнение равенства $\omega = h_x$.

Утверждение 2.1. Если ЭП подлинная, то $\omega = h_x = h(x_m)$.

□ Из свойств RSA следует, что

$$\omega = s^{k_{om}} \bmod N_m = h_x^{k_{om} k_{зт}} \bmod N_m = h_x = h(x_m). \blacksquare$$

Первое свойство ЭП выполняется, т.к. никто кроме владельца подписи не может разложить N на простые множители p и q. Для злоумышленника это будет односторонняя функция. По состоянию на 2010 год, при N порядка 1024 бит, эта задача для злоумышленника практически неразрешима. Злоумышленник, зная N и k_o, не может определить $k_з$. Действительно, чтобы вычислить $k_з = k_o^{-1} \bmod \phi$, требуется знать $\phi = (p-1)(q-1)$, а, следовательно, p и q. Второе и третье свойства выполняются, т.к. выполняется первое.

Рассмотрим пример формирования ЭП с помощью алгоритма RSA. Пусть $p = 5$ и $q = 11$. Тогда $N = 55$, $\phi = 40$. Пусть $k_o = 3$, тогда $k_з = 3^{-1} \bmod 40 = 27$. Подписываемый документ $x = abbbaa$. Будем полагать, что значение хеш-образа документа равно $h_x = h(abbbaa) = 13$.

Вычисляется ЭП

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

Затем формируется подписанный документ $\langle abbbaa, 7 \rangle$. Зная открытый ключ подписавшего документ $\langle 55, 3 \rangle$. В соответствии с правилом проверки подлинности ЭП вычисляется $h_x = h(abbbaa) = 13$ и затем

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

Значение $h_x = \omega = 13$, следовательно, ЭП верна.

2.2. Электронная подпись на основе криптосистемы Эль Гамаля

Пусть, как и в предыдущем случае, абонент m собирается подписывать документы. На первом шаге формируются параметры криптосистемы Эль Гамаля. Абонент выбирает большое простое число p и число g, такое, что различные степени g суть различные числа по модулю p. Эти числа хранятся в открытом виде и могут быть общими для целой группы абонентов. Затем абонент m выбирает случайное число $k_{зт}$, $1 < k_{зт} < p-1$, которое держится в секрете. Затем абонент вычисляет число

$$k_{от} = g^{k_{зт}} \bmod p,$$

которое является открытым. Теперь абонент готов подписывать документы.

На следующем шаге абонент вычисляет хеш-функцию исходного документа $h_x = h(x_m)$, которая должна удовлетворять условию $1 < h_x < p$.

На третьем шаге абонент выбирает случайное число c, $1 < c < p-1$, взаимно простое с $p-1$ и вычисляет числа

$$r = g^c \bmod p,$$
$$u = (h_x - k_{зт} \cdot r) \bmod (p-1),$$
$$s = c^{-1} \cdot u \bmod (p-1),$$

где $c^{-1} c \bmod (p-1) = 1$. Числа $\langle s, r \rangle$ являются ЭП. Таким образом, подписанное сообщение имеет вид $\langle x_m; s, r \rangle$.

Получатель подписанного документа заново вычисляет хеш-функцию $h_x = h(x_m)$. Затем проверяет подлинность подписи, используя равенство

$$k_{от}^r \cdot r^s = g^{h_x} \bmod p.$$

Утверждение 2.2. Если ЭП верна, то условие $k_{от}^r \cdot r^s = g^{h_x} \bmod p$ выполняется.

□ Действительно,

$$k_{от}^r \cdot r^s = (g^{k_{зт}})^r (g^c)^s = g^{k_{зт} r} g^{c(c^{-1}(h_x - k_{зт} r))} = g^{k_{зт} r} g^{h_x} g^{-k_{зт} r} =$$
$$= g^{h_x} \bmod p. \blacksquare$$

Первое свойство ЭП выполняется, т.к. никто кроме законного владельца не знает $k_з$. По этой же причине выполняются второе и третье свойства ЭП.

Основное отличие ЭП на базе криптосистемы Эль Гамаля от ЭП на базе криптосистемы RSA заключается в длине подписи. ЭП на базе криптосистемы RSA $\langle x_m, s \rangle$ практически в два раза короче, чем ЭП на базе криптосистемы Эль

Гамаля $\langle x_m; s, r \rangle$, т.е., если длина ЭП RSA 1024 бит, то длина ЭП Эль Гамаля 2048 бит.

Рассмотрим пример. Пусть общие параметры $p = 23$, $g = 5$. Пользователь выбирает секретный ключ $k_з = 7$ и вычисляет открытый ключ

$$5^7 \bmod 23 = 17.$$

Подписываемый документ имеет вид $x = baaaab$. Вычисляется хеш-образ $h_x = h(baaaab) = 3$, затем генерируется число $c = 5$ $(1 < 5 < 22)$. Вычисляется ЭП:

$$r = 5^5 \bmod 23 = 20,$$
$$u = (3 - 7 \cdot 20) \bmod 22 = 17,$$
$$s = 9 \cdot 17 \bmod 22 = 21.$$

При этом $5^{-1} \bmod 22 = 9$. Подписанное сообщение имеет вид $\langle baaaab, 20, 21 \rangle$. Для проверки подлинности ЭП вычисляется хеш-образ документа $h_x = h(baaaab) = 3$, а затем вычисляется

$$17^{20} \cdot 20^{21} \bmod 23 = 5^3 \bmod 23,$$
$$10 = 10.$$

Следовательно, ЭП верна.

Рассмотрим методы сокращения длины ЭП Эль Гамаля [10].

1. Схема ЭП Эль Гамаля с сокращенной длиной параметра s.

Уравнение проверки подлинности ЭП $k_o^r \cdot r^s = g^{h_x} \bmod p$ может выполняться также в случае, когда в качестве g берется число, относящееся к простому показателю q, где $q \mid p - 1$. Для этого параметр s должен быть вычислен из соотношения

$$h_x = (k_з r + cs) \bmod q.$$

Можно выбрать простой модуль p таким образом, чтобы разложение $p - 1$ содержало бы простой множитель q, размер которого существенно меньше размера p. Например, для модуля p длиной 2048 бит длина q может составлять 160 бит. Тогда s будет иметь длину не более 160 бит.

2. Схема ЭП Эль Гамаля с сокращенной длиной параметров s и r.

Соотношение для проверки подписи $k_o^r \cdot r^s = g^{h_x} \bmod p$ может быть преобразовано к виду

$$r = g^{\left(\frac{h_x}{s}\right)} k_o^{\left(-\frac{r}{s}\right)} \bmod p.$$

При этом, вместо r в степени при k_o можно использовать значение его хеш-образ $h_r = h(r)$. В этом случае уравнение проверки подписи имеет вид

$$r = g^{\left(\frac{h_x}{s}\right)} k_o^{\left(-\frac{h_r}{s}\right)} \bmod p.$$

Чтобы ЭП была корректной, законный владелец подписи должен вычислить параметр s из следующего уравнения

$$h_x = (k_з h_r + cs) \bmod q.$$

Так как при проверке ЭП не требуется выполнять никаких вычислений с использованием параметра r, то проверка ЭП может быть осуществлена в соответствии с уравнением

$$h_r = h\left(g^{\left(\frac{h_x}{s}\right)} k_o^{\left(-\frac{h_r}{s}\right)} \bmod p \right).$$

В рассматриваемом случае нет необходимости предоставлять проверяющему параметр r, имеющий сравнительно большую длину. Достаточно для проверки представить значение h_r, размер которого равен примерно 160 бит. При применении метода сокращения длины ЭП за счет параметра s получаем общую длину ЭП порядка 320 бит. Таким образом, достигается существенное снижение длины ЭП.

Сокращение длины ЭП не уменьшает ее стойкости, т.к. сложность задачи дискретного логарифмирования для злоумышленника не изменяется, поскольку все вычисления ведутся по модулю исходного размера.

2.3. Стандарты электронных подписей

В многих странах мира существуют стандарты на ЭП. Рассмотрим российские стандарты ГОСТ Р34.10-94 и ГОСТ Р34.10-2012, а также особенности американского стандарта ЭП FIPS 186 [7,9-12].

Российский стандарт ГОСТ Р34.10-94 был принят в 1994 году, а американский стандарт FIPS 186 в 1991 году. В основе обоих стандартов лежит по сути один и тот же алгоритм, называемый DSA (Digital Signature Algorithm) и являющийся вариацией ЭП Эль Гамаля. Российский стандарт ГОСТ Р34.10-2012 введен в действие с 1 января 2013 года взамен утратившего силу стандарта ГОСТ Р 34.10-2001. Стандарт ГОСТ Р 34.10-2012 содержит описание процессов формирования и проверки ЭП, реализуемой с использованием операций в группе точек эллиптической кривой, определенной над конечным простым полем.

Российский стандарт ГОСТ Р 34.10-94. Для некоторого сообщества пользователей выбираются общие несекретные параметры. Прежде всего необходимо найти два простых числа: q длиной 256 бит и p длиной 1024 бит, между которыми выполняется соотношение $p = bq + 1$, где b - небольшое

целое число. Старшие биты q и p должны быть равны единице. Затем выбирается число $a > 1$, такое что

$$a^q \bmod p = 1.$$

В результате получают три общих параметра - q, p и a.

Равенство означает, что при возведении a в степени по модулю p показатели приводятся по модулю q, т.е. $a^q \bmod p = a^{b \bmod q} \bmod p$. Такое приведение будет постоянно выполняться при генерации и проверке подписи, в результате чего длина показателей степени в рамках рассматриваемого алгоритма не превышает 256 бит, что значительно упрощает вычисления.

Далее каждый пользователь выбирает случайное число x, $0 < x < q$ и вычисляет

$$y = a^x \bmod p.$$

Число x является секретным ключом, а y - открытым ключом. Приведенные обозначения параметров взяты из ГОСТ Р 34.11.-94. Открытые ключи пользователей указываются в сертифицированном справочнике, который есть у всех пользователей. На этом этап выбор параметров заканчивается.

Генерация ЭП выполняется следующим образом:

1. Вычисляется значение хеш-функции h для выбранного документа m, причем $0 < h < q$. В российском стандарте хеш-функция определяется ГОСТ Р 34.11-94.

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

3. Вычисляется

$$r = \left(a^k \bmod p\right) \bmod q.$$

Если $r = 0$, то требуется перейти к шагу 2.

4. Вычисляется

$$s = \left(kh + xr\right) \bmod q.$$

Если $s = 0$, то требуется перейти к шагу 2.

5. Формируется подписанное сообщение $\langle m; s, r \rangle$.

Для проверки ЭП выполняются следующие вычисления:

1. Вычисляется хеш-функция h.

2. Проверяется выполнение неравенств

$$0 < r < q,\ 0 < s < q.$$

3. Вычисляется

$$u_1 = s \cdot h^{-1} \bmod q,\ u_2 = -r \cdot h^{-1} \bmod q,$$

$$\vartheta = \left(a^{u_1} y^{u_2} \bmod p\right) \bmod q.$$

4. Проверяется выполнение равенства $\vartheta = r$.

Если хотя бы одна из проверок (шаги 2 и 4) дает неверный результат, то ЭП считается недействительной.

Утверждение 2.3. Если подпись была сформирована законным владельцем, то $\vartheta = r$.

□ Действительно,

$$\vartheta = \left(a^{sh^{-1}} y^{-rh^{-1}} \bmod p \right) \bmod q = \left(a^{(k+xr)h^{-1}} a^{-xrh^{-1}} \bmod p \right) \bmod q =$$

$$= \left(a^{k+xrh^{-1}-xrh^{-1}} \bmod p \right) \bmod q = \left(a^{k} \bmod p \right) \bmod q = r. \blacksquare$$

Американский стандарт FIPS 186. Рассмотрим только отличия американского стандарта ЭП от российского стандарта. Они сводятся к следующему:

1. Длина числа q берется равной 160 бит.

2. В качестве хеш-функции используется алгоритм SHA-1.

3. При генерации подписи на шаге 4 параметр s вычисляется по формуле:

$$s = k^{-1}(h + xr) \bmod q.$$

4. При проверке подписи на шаге 3 вычисления u_1 и u_2 выполняются по формулам:

$$u_1 = h \cdot s^{-1} \bmod q,\ u_2 = r \cdot s^{-1} \bmod q.$$

Российский стандарт ГОСТ Р 34.10-2012. На рис. 2.2 представлен алгоритм формирования и проверки ЭП. Каждый пользователь должен иметь ключ формирования подписи $k_з$ и ключ проверки подписи $Q(x_q, y_q)$. В алгоритме формирования ЭП вычисляется хеш-функция $h(\bar{x})$, на основании которой определяется число α, двоичным представлением которого является $h(\bar{x})$. Затем определяется число:

$$e = \begin{cases} \alpha \bmod q, & e \neq 0, \\ 1, & e = 0. \end{cases}.$$

Генерация случайного числа k, осуществляется при условии $0 < k < q$. Затем вычисляется точка $C = kG \bmod q$ и по ее абсциссе определяется число $r = x_C \bmod q$. Точка G удовлетворяет равенству $qG = O$. Далее вычисляется число

$$d = (k_з + ke) \bmod q.$$

ЭП представляет собой операцию конкатенации двух двоичных векторов $\vec{r}$ и $\vec{d}$, соответствующих r и d

$$s = (\vec{r} \| \vec{d}).$$

Для проверки подписи выполняются аналогичные действия, но при этом вычисляются:

$$\nu = e^{-1} \bmod q,$$

$$Z_1 = d\nu \bmod q,\ Z_2 = -r\nu \bmod q.$$

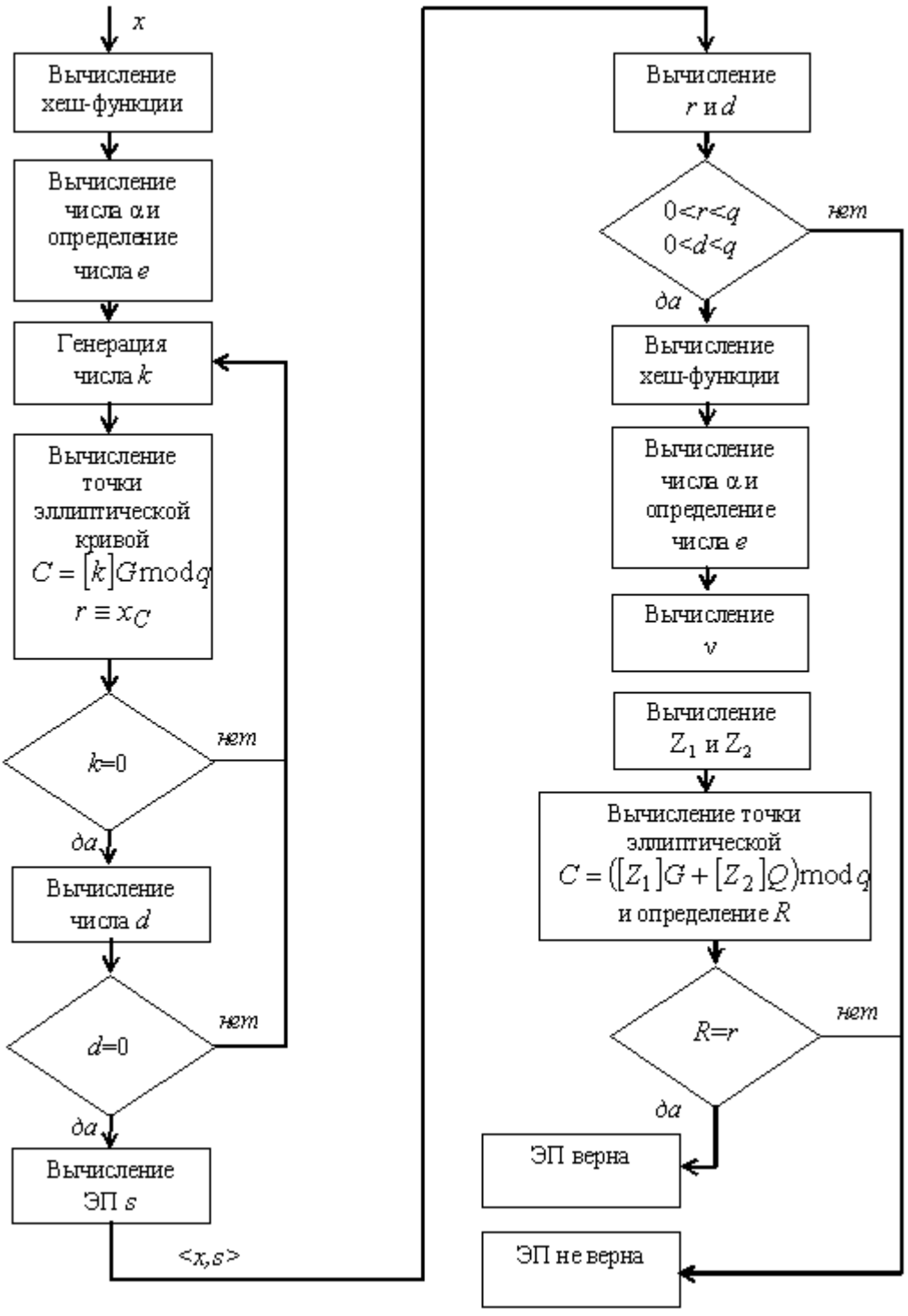

Рис. 2.2. Алгоритм формирования и проверки ЭП ГОСТ Р34.10-2012

На основании этих чисел вычисляется точка на эллиптической кривой $C = (Z_1 G + Z_2 Q) \bmod q$ и по ее абсциссе определяется число:

$$R = x_C \bmod q .$$

Если $R = r$, то ЭП верна.

Параметрами рассмотренной схемы формирования ЭП являются: простое число p - модуль эллиптической кривой; эллиптическая кривая E, задаваемая коэффициентами a, b; целое число m - порядок группы точек эллиптической кривой E; простое число q - порядок циклической подгруппы группы точек эллиптической кривой E, для которой выполнено:

$$\begin{cases} m = nq, n \in Z, n \geq 1; \\ 2^{254} < q < 2^{256} \vee 2^{508} < q < 2^{512}; \end{cases}$$

точка G, удовлетворяющая равенству $[q]G = O$; хеш-функция, определяемая стандартом ГОСТ Р.34.11-2012.

2.4. Электронная подпись на основе решения системы сравнений

Генерация ЭП, основанной на сложности дискретного логарифмирования, может быть осуществлена с использованием нового подхода [10], в котором оба элемента подписи r и s представляются в одинаковом виде

$$r = g^{c_1} \bmod p, \; s = g^{c_2} \bmod p, \qquad (2.1)$$

где c_1 и c_2 вычисляются одновременно как одно из решений системы двух уравнений, записываемых в зависимости от вида проверочного соотношения.

Идея этого подхода состоит в том, чтобы сделать вычислительно невозможным вычисление одного из параметров r и s при заранее заданном значении второго параметра. Параметры r и s используются как аргументы двух различных функций $F_1(r, s)$ и $F_2(r, s)$. При определенных ограничениях на значения аргументов их можно изменять таким образом, что значение функции $F_2(r, s)$ будет оставаться неизменным. При этом значение функции $F_1(r, s)$ должно изменяться таким образом, что можно подобрать пару значений r и s, при которых будет выполняться некоторое проверочное соотношение. Таким образом, в определенной области пар значений r и s имеем $F_2 = Z = \text{const}$, поэтому проверочное соотношение в принципе может быть упрощено так, что при определенном его виде можно вычислить подпись $r(Z)$ и $s(Z)$, зависящую от Z.

При составлении конкретных вариантов ЭП могут быть использованы следующие пары функций $F_1(r, s)$, $F_2(r, s)$:

$$\frac{r}{s} \bmod p, \; rs \bmod p \text{ или } rs \bmod p, \; \frac{r}{s} \bmod p;$$

$$rs^h \bmod p, \; rs \bmod p;$$

$$rs \bmod p, \; rs^2 \bmod p;$$

$$rs^Z \bmod p, \; rs \bmod p,$$

где $Z = rs \bmod p$.

При этом значения r и s предполагается выражать через c_1 и c_2 в виде (2.1).

Если функция $F_2(\cdot, s)$ имеет вид: $F_2(\cdot, s) = \frac{r}{s} \bmod p$, то условие постоянства значения функции запишется в виде: $c_1 - c_2 = U \bmod \gamma$, где γ - некоторый показатель, к которому число g относится по модулю p; U - случайно выбираемое число. В случае если $F_2(\cdot, s) = rs \bmod p$ или $F_2(\cdot, s) = rs^h \bmod p$ условие постоянства значения функции запишется в виде: $c_1 + c_2 = U \bmod \gamma$ и $c_1 + hc_2 = U \bmod \gamma$ соответственно. При этом предполагаются следующие варианты проверочных соотношений:

$$\frac{r}{s} = k_o^{(s \bmod p)} g^{(s \bmod p)} \bmod p,$$

$$r = s k_o^{h(s \bmod p)} g^{(s \bmod p) \bmod \delta} \bmod p,$$

$$\left(\frac{r}{s}\right)^{(s \bmod p)} = k_o^{(s \bmod p) \bmod \delta} g^h \bmod p,$$

где δ - произвольное простое число длины $|\delta| \approx 0{,}25|p|$. Операция $F_2 \bmod p$ определяет сжимающую функцию F_2', значение которой остается постоянной, если значение $F_2 = rs \bmod p$ не изменяется. Это позволяет получить и использовать в проверочном соотношении две функции, зависящие от параметров r и s, причем такие, что их значения фиксируются одновременно при условии, что параметры c_1, c_2 удовлетворяют определенным условиям.

Необходимость использования пары одновременно фиксируемых функций F_2 и F_2' связана с тем, что в проверочном соотношении требуется задать показатели степени элементов k_o и g, зависящие от c_1, c_2. Если это условие не выполнено, то ЭП можно легко подделать путем включения фиксированной степени k_o или g как дополнительного множителя в представлении одного из параметров r и s.

Для примера рассмотрим ЭП с проверочным соотношением

$$rk_o^{(s \bmod p)} = sg^{h(s \bmod p)} \bmod p,$$

которое может быть представлено в другом виде

$$r = s\left(g^h k_o^{-1}\right)^{(s \bmod p)} \bmod p.$$

Элементы подписи определяются на основе соотношений

$$r = \left(g^h k_o^{-1}\right)^{c_1} \bmod p, \qquad (2.2)$$

$$s = \left(g^h k_o^{-1}\right)^{c_2} \bmod p. \qquad (2.3)$$

При

$$c_1 + c_2 = U \bmod \gamma \qquad (2.4)$$

значение $Z = rs \bmod p = \left(g^h k_o^{-1}\right)^{\alpha} \bmod p$ является фиксированным и условием выполнимости проверочного соотношения является

$$c_1 = c_2 + Z \bmod p. \qquad (2.5)$$

Действительно, если (2.4) и (2.5) выполняются, то получаем

$$\left(g^h k_o^{-1}\right)^{c_1} = \left(g^h k_o^{-1}\right)^{c_2} \left(g^h k_o^{-1}\right)^{\alpha} \bmod p \Rightarrow$$

$$\Rightarrow r k_o^{(s \bmod p)} = s g^{h (s \bmod p)} \bmod p.$$

Таким образом, ЭП может быть вычислена без использования секретного ключа путем совместного решения уравнения сравнений (2.4) и (2.5) и последующего вычисления элементов ЭП по формулам (2.2) и (2.3).

В рассмотренном подходе по ключу и подписи $\langle r, s \rangle$ вычислительно крайне сложно найти U, которое выбирается произвольно. Для вычисления U требуется вычислить и c_1, и c_2, однако, для этого необходимо решить задачу дискретного логарифмирования.

Дальнейшим развитием ЭП на основе решения системы сравнений является ЭП вида $\langle c_1, s \rangle$. Отказ от использования значения r в качестве элемента подписи в ЭП вида $\langle c_1, s \rangle$ не только устраняет возможность подделки подписи на основе замены переменных, но и дает ряд преимуществ, которые заключаются в следующем:

1. Отказ от одной из двух фиксируемых функций F_2 и F_2', поскольку число g возводится непосредственно в степень c_1, которая приобретает конкретное значение только после решения системы сравнений. Теперь достаточно включения в проверочные соотношения только одного множителя $k_o^{(c_1 \cdot s \bmod p)} \bmod p$ с показателем степени, определяемым после решения системы сравнений. Это обеспечивает упрощение вида проверочных соотношений.

2. Размер ЭП может быть существенно сокращен, если в качестве числа g использовать число, относящееся по модулю p к простому показателю γ, длина которого существенно меньше длины p, например, 160…256 бит.

3. Появляется возможность реализации ЭП, в которой условие фиксирования задается сравнениями, включающими произведение или отношение значений c_1, и c_2.

Примером ЭП вида $\langle c_1, s \rangle$ может служить подпись, вычисляемая на основе соотношений

$$c_1 = \frac{U - k_з hZ}{k_з + 1} \bmod \gamma,\ c_2 = \frac{U k_з + k_з hZ}{k_з + 1} \bmod \gamma,$$

$$s = g^{c_2} \bmod p.$$

Система решаемых сравнений и формула вычисления Z имеют вид

$$\begin{cases} c_1 + c_2 = U \bmod \gamma, \\ c_2 = k_з hZ + k_з c_1 \bmod \gamma; \end{cases}$$

$$Z = g^U \bmod p.$$

Проверочное соотношение

$$s = k_o^{h} \left(g^{c_1} \bmod p \right)^{c_1} \bmod p.$$

2.5. Коллективная и композиционная электронная подпись

Одной из актуальных задач в области аутентификации электронных документов с помощью ЭП является проблема одновременного подписания документа (например, контракта) группой людей на расстоянии. Эта задача естественным образом решается путем применения алгоритмов коллективной ЭП [10]. **Коллективная ЭП** формируется на основе параметров, вырабатываемых всеми подписывающими, причем подлинная коллективная ЭП может быть сформирована только для случая, когда документ подписан всеми участниками протокола коллективной ЭП. Вычислительно невозможно сформировать урезанную коллективную ЭП, т.е. создать такую ЭП, которая соответствовала бы меньшему числу пользователей. Это свойство решает проблему подписания одного документа несколькими пользователями.

На практике возникают ситуации, когда требуется одновременно подписать пакет документов, причем коллектив подписывающих для одного документа может отличаться от коллектива подписывающих для другого документа. Решение этой задачи требует другого протокола ЭП. Таким протоколом является протокол композиционной ЭП [10]. Композиционная ЭП является, по существу, дальнейшим развитием схемы коллективной ЭП. **Композиционная ЭП** позволяет различным пользователям или группам пользователей подписывать одновременно различное количество разных документов.

Главное отличие коллективной и композиционной ЭП состоит в различии процедур формирования коллективного открытого ключа. В случае коллективной ЭП коллективный открытый ключ является функцией открытых ключей отдельных пользователей, а в случае композиционной ЭП – функцией открытых ключей отдельных пользователей и значений хеш-функций подписываемых документов.

Протоколы коллективной и композиционной ЭП реализовываются на основе различных алгоритмов и стандартов ЭП. Например, на основе

алгоритмов ЭП, основанных на задаче дискретного логарифмирования, на сложности задачи факторизации, на основе стандарта ГОСТ 34.10.94 и других стандартов.

Для примера рассмотрим алгоритмы коллективной и композиционной ЭП, основанных на сложности задачи факторизации.

В пункте 2.1 рассмотрена ЭП с использованием алгоритма RSA. Открытым ключом является пара чисел $\langle N, k_o \rangle$, а тройка чисел $\langle p, q, k_з \rangle$ является секретной. Формирование и проверка подписи выполняется по формулам:

$$s = h_x^{k_з} \bmod N,\ h_x = s^{k_o} \bmod N.$$

Размер хеш-функции составляет обычно 160…256 бит, что значительно меньше размера модуля N.

Пусть коллектив пользователей I, $i = \overline{1, I}$ желает подписать документ x, представленный хеш-функцией $h_x = h(x)$. Коллективная подпись может быть сформирована следующим образом:

1. Открытые ключи пользователей упорядочиваются по возрастанию модуля, т.е. $N_1 < N_2 < ... < N_I$.

2. Последовательно формируются подписи: $s_1 = h_x^{k_{з1}} \bmod N_1$, $s_2 = s_1^{k_{з2}} \bmod N_2$, …, $s_i = s_{i-1}^{k_{зi}} \bmod N_i$,…, $s_I = s_{I-1}^{k_{зI}} \bmod N_I$. Значение s_I является коллективной ЭП.

3. Проверка коллективной подписи выполняется в обратном порядке: $s_{I-1} = s_I^{k_{oI}} \bmod N_I$, $s_{I-2} = s_{I-1}^{k_{oI-1}} \bmod N_{I-1}$, …, $s_1 = s_2^{k_{o2}} \bmod N_2$, $h_x = s_1^{k_{o1}} \bmod N_1$. Если $h_x = s_1^{k_{o1}} \bmod N_1$ справедливо, то коллективная ЭП подлинная.

Композиционная подпись к документам x_1, x_2, …, x_J, представленным хеш-функциями h_{x1}, h_{x2}, …, h_{xJ}, может быть сформирована следующим образом.

1. Открытые ключи пользователей упорядочиваются по возрастанию модуля, т.е. $N_1 < N_2 < ... < N_I$.

2. Последовательно формируются подписи: $s_1 = h_x^{k_{з1}} \bmod N_1$, $s_2 = s_1 h_{x2}^{k_{з2}} \bmod N_2, .., s_j = s_{j-1} h_{xj}^{k_{зj}} \bmod N_j, ..., s_J = s_{J-1} h_{xJ}^{k_{зJ}} \bmod N_J$. Значение s_J является композиционной ЭП.

3. Проверка композиционной ЭП выполняется следующим образом: $s_{J-1} = s_J^{k_{oJ}} h_{xJ}^{-1} \bmod N_J$, $s_{J-2} = s_{J-1}^{k_{oJ-1}} h_{xJ-1}^{-1} \bmod N_{J-1}$, …, $s_1 = s_2^{k_{o2}} h_{x2}^{-1} \bmod N_2$, $h_x = s_1^{k_{o1}} \bmod N_1$. Если $h_x = s_1^{k_{o1}} \bmod N_1$ выполняется, то композиционная

подпись подлинная. Каждый пользователь в этом случае подписывает соответствующий ему документ. Размер композиционной подписи равен длине модуля N, хранить хеш-функции от документов не нужно, т.к. их можно всегда вычислить.

В рассмотренном протоколе композиционной подписи подпись может быть сформирована любым подмножеством пользователей I.

2.6. Слепая электронная подпись

Слепая электронная подпись является разновидностью ЭП и отличается тем, что подписывающая сторона не может точно знать содержимое подписываемого документа [2,10]. Понятие слепой подписи введено Дэвидом Чаумом, им же предложена первая реализация алгоритма слепой ЭП.

Основная идея слепой электронной подписи заключается в следующем. Отправитель А посылает документ стороне В, который подписывает его и возвращает А. Используя полученную подпись, сторона А может вычислить подпись стороны В. По завершении этого протокола сторона В ничего не знает ни о документе, ни о подписи под этим документом. Эту схему можно сравнить с конвертом, в котором размещён документ и копировальный лист. Если подписать конверт, то подпись отпечатается на документе, и при вскрытии конверта документ уже будет подписан.

Таким образом цель алгоритма слепой электронной подписи состоит в том, чтобы воспрепятствовать подписывающему лицу В ознакомиться с сообщением стороны А, которое он подписывает, и с соответствующей подписью под этим сообщением.

Слепая электронная подпись Чаума. Слепая электронная подпись Чаума основана на криптосистеме RSA. Пусть пользователь А желает подписать документ x у пользователя В таким образом, чтобы последний не мог прочесть подписываемый документ. Для этого необходимо выполнить следующее:

1. Пользователь А генерирует случайное простое число c взаимно простое с N_B, где N_B - часть открытого ключа пользователя В.

Затем пользователь А вычисляет значение:

$$x' = c^{k_{oB}} x \bmod N_B$$

и предъявляет его на подпись пользователю В, чтобы последний подписал x' в соответствии со стандартной процедурой ЭП RSA. Пользователь В не может прочесть документ x, поскольку он преобразован путем наложения «разового» ключа $c^{k_{oB}}$ с использованием операции модульного умножения. Таким образом, пользователь А «маскирует» документ x при помощи множителя, который иногда называют **маскирующим множителем.**

2. Пользователь В подписывает документ x'

$$s' = \left(k_{oB} x\right)^{k_{зB}} \bmod N_B = c x^{k_{зB}} \bmod N_B .$$

Заметим, что по значению s' к сообщению x' пользователь В не имеет возможности вычислить $x^{k_{зB}} \bmod N_B$. Однако по значению $x^{k_{зB}} \bmod N_B$ пользователю А легко вычислить x

$$\left(k_{зB}\right)^{k_{oB}} \bmod N_B = x .$$

Следовательно, после получения значения

$$s = x^{k_{зB}} \bmod N_B$$

пользователь А должен держать его в секрете от подписавшего.

3. После получения от пользователя В значения s', используя расширенный алгоритм Евклида, пользователь А вычисляет для числа c мультипликативно обратный элемент c^{-1} по модулю N_B и формирует подпись пользователя В к исходному документу

$$s = c^{-1} s' = c^{-1} c x^{k_{зB}} = x^{k_{зB}} \left(\bmod N_B\right).$$

В настоящее время слепые ЭП находят широкое применение в сфере цифровых денег, в системах тайного электронного голосования. Для примера рассмотрим применение слепой ЭП в системах тайного электронного голосования.

Схема применения слепой ЭП состоит в следующем. Избиратель подготавливает избирательный бюллетень со своим выбором, который он сделал, шифрует его секретным ключом и маскирует. Далее избиратель подписывает избирательный бюллетень и посылает его валидатору. Валидатор проверяет, что подпись принадлежит зарегистрированному избирателю, который еще не голосовал. Если избирательный бюллетень действителен, валидатор подписывает избирательный бюллетень и возвращает его избирателю. Избиратель удаляет маскировку, раскрывая таким образом зашифрованный избирательный бюллетень, подписанный валидатором. Далее избиратель посылает избирательный бюллетень счётчику, который проверяет подпись на зашифрованном избирательном бюллетене. Если избирательный бюллетень действителен, счётчик размещает его в списке, который будет издан после всего голосования. После того, как список издан, избиратели проверяют, что их избирательные бюллетени находятся в списке и посылают счётчику ключи расшифрования, необходимые, чтобы открыть их избирательные бюллетени. Счётчик использует эти ключи для расшифрования избирательных бюллетеней и добавляет голос к общему числу. После выборов счётчик издает ключи расшифрования наряду с зашифрованными избирательными бюллетенями, чтобы избиратели могли независимо проверить выбор.

Существует много алгоритмов слепой ЭП, отличающихся различной степенью сложности и возможностями по применению. Чаум, например, разработал целое семейство более сложных алгоритмов слепой подписи под

общим названием *неожиданные слепые подписи***.** Подробно о слепых ЭП можно прочесть в [10].

Контрольные вопросы

1. Дайте определение электронной подписи. Каковы требования к электронной подписи?
2. Поясните алгоритмы формирования и проверки электронной подписи RSA и Эль Гамаля. В чем их отличие?
3. Какие стандарты электронных подписей существуют в настоящее время?
4. Дайте определение слепой электронной подписи.
5. Назовите способы сокращения длины электронной подписи. В чем их суть?
6. Дайте определение коллективной и композиционной электронной подписи. В чем их отличие?

3. Хеш-функции

Определение 3.1. Хеш-функцией называется преобразование h, превращающее последовательность x произвольной длины в информационную последовательность h_x фиксированной длины [1,2,6,7,9].

В общем случае h_x гораздо меньше, чем x, например, h_x может быть 128 или 256 бит, тогда как x может быть размером в мегабайты и более.

Хеширование иногда считают видом криптографического преобразования. Вместе с тем, криптографическое преобразование, по определению, является обратимым, а хеширование представляет собой необратимое преобразование.

3.1. Требования к хеш-функции

Для того чтобы хеш-функция могла быть использована в криптографических алгоритмах она должна обладать следующими свойствами [2,9]:

- преобразование h может быть применено к x любого размера;
- выходное значение h_x должно иметь фиксированный размер;
- значение h_x достаточно просто вычисляется для любого x;
- для любого значения h_x с вычислительной точки зрения невозможно найти x;
- для любого значения x с вычислительной точки зрения невозможно найти $x' \neq x$ такое, что $h(x) = h(x')$;

- значение хеш-функции h_x должно быть чувствительным к любым изменениям входной информационной последовательности x.

Если хеш-функция обладает перечисленными свойствами, то она считается качественной. Для качественной хеш-функции три следующие задачи являются вычислительно неразрешимыми.

1. **Задача нахождения прообраза** – это задача нахождения входной последовательности x по заданному хеш-образу h_x. Хеш-функция должна быть стойкой в смысле обращения.

2. **Задача нахождения коллизий** – это задача нахождения последовательностей x' и x'', причем $x' \neq x''$, для которых $h(x') = h(x'')$. Хеш-функция должна быть стойкой в смысле нахождения коллизий.

3. **Задача нахождения второго прообраза** – это задача нахождения для заданной входной последовательности x другой входной последовательности x', причем $x' \neq x$ такой, что $h(x) = h(x')$. Эта задача является разновидностью задачи нахождения коллизий.

В табл. 3.1 представлены атаки на хеш-функции и приведена оценка их вычислительной сложности.

Таблица 3.1

Оценка сложности атак на хеш-функцию

Атака	Вычислительная сложность
Нахождение прообраза	2^n
Нахождение второго прообраза	2^n
Нахождение коллизии	$2^{\frac{n}{2}}$

Если n - разрядность хеш-образа, то вычислительная сложность первой и второй атаки пропорциональная 2^n, а для третьей атаки - $2^{\frac{n}{2}}$. Необходимо отметить, что задача нахождения коллизий бывает двух типов. Задача нахождения коллизий первого типа заключается в поиске двух произвольных сообщений, имеющих одинаковое значение хеш-образа. Существует точка зрения, что такая атака бесполезна для атакующего. Однако существуют атаки на конкретные алгоритмы хеширования с использованием уже найденных коллизий алгоритма хеширования MD5. Во втором случае предполагается, что одно сообщение реальное, а другое фальсифицированное, при этом оба сообщения должны быть осмысленными. Решение состоит в создании двух списков осмысленных сообщений путем внесения избыточности или модификации содержимого сообщения без изменения его смысла.

Подавляющее большинство современных алгоритмов хеширования строится или по итерационной схеме, или на основе симметричных блочных криптосистем.

3.2. Итерационные хеш-функции

Схема итерационной хеш-функции представлена на рис. 3.1. На схеме: x_1, x_2, …, x_l - блоки дополненного сообщения, l - число блоков, h_0 - фиксированный вектор, называемый стартовым или вектором инициализации, h_1, h_2, …, h_l - промежуточные результаты вычисления хеш-образа.

Входная информационная последовательность дополняется до длины, кратной n, где n - разрядность блока данных, обрабатываемого функцией сжатия. **Функция сжатия** $f_{сж}$ на i-м итерационном шаге принимает результат вычисления h_{i-1} на предыдущем шаге и x_i блок входных данных, а затем формирует результат $h_i = f_{сж}(h_{i-1}, x_i)$. На последнем итерационном шаге получают значение хеш-образа $h_x = h_l = f_{сж}(h_{l-1}, x_l)$.

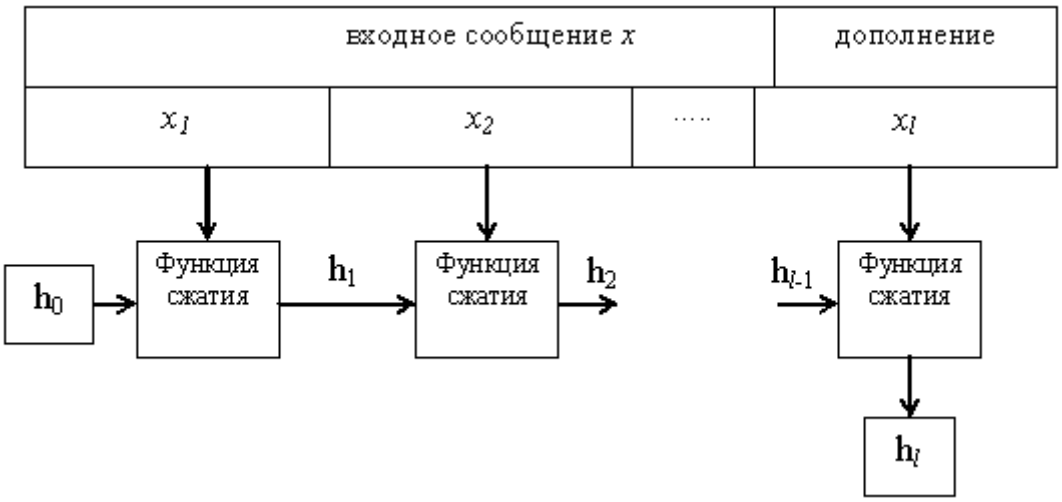

Рис. 3.1. Схема итерационной хеш-функции

Итерационная хеш-функция является стойкой в смысле нахождения коллизий, если аналогичным свойством обладает функция сжатия $f_{сж}$.

Рассмотрим в качестве примера итерационный алгоритм хеширования SHA (Secure Hash Algorithm), являющегося частью стандарта SHS (Secure Hash Standard), разработанный Национальным институтом стандартов и технологий США в 1993 году. Алгоритм был предложен Р. Ривестом и в своем первоначальном виде обозначался как SHA-1. В 1995 году стандарт был пересмотрен и определены четыре версии алгоритма: SHA-224, SHA-256, SHA-384, SHA-512. Все версии имеют одинаковую структуру, поэтому их называют общим именем SHA-2. В табл. 3.2 представлены характеристики SHA.

Рассмотрим версию SHA-1 (см. рис. 3.2). На первом этапе входная информационная последовательность дополняется до длины 512 бит.

Таблица 3.2

Характеристики SHA

Характеристика	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Максимальная длина сообщения	2^{64}-1	2^{64}-1	2^{64}-1	2^{128}-1	2^{128}-1
Длина блока	512	512	512	1024	1024
Длина хеш-образа	160	224	256	384	512
Число раундов	80	64	64	80	80

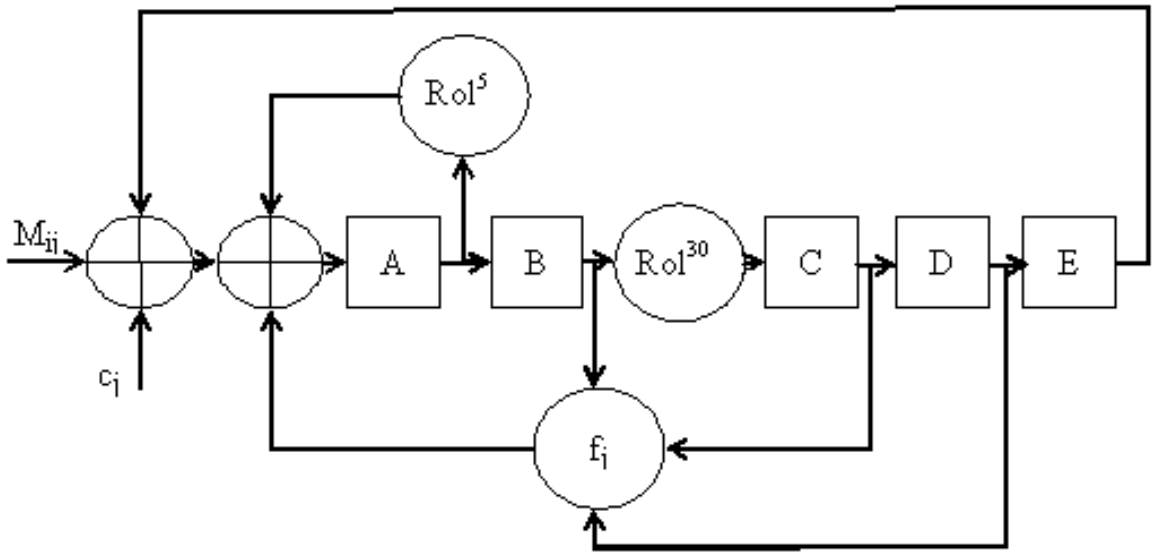

Рис. 3.2. Схема одного шага алгоритма SHA

Сначала информационная последовательность дополняется до длины на 64 двоичных разряда меньше числа, кратного 512: к концу последовательности приписываются 1, а затем необходимое количество нулей. После этого приписывается 64-разрядный код длины сообщения. Если длина исходного сообщения больше 2^{64}, используются только 64 младших разряда этого кода. Пусть после дополнения получена информационная последовательность

$$x = x_1 x_2 x_3 ... x_i ... x_l,\ i = \overline{1,l},\ |x_i| = 512.$$

На вход i-го основного цикла SHA преобразования поступает i-й блок информационной последовательности и результат работы предыдущего цикла SHA, т.е.

$$h_{SHAi} = h(x_i, h_{SHAi-1}).$$

Перед началом каждого цикла соответствующий блок расширяется до 80 слов по 32 разряда в каждом. Расширение происходит следующим образом. Пусть w - исходный блок, $\tilde{w}_1, \tilde{w}_2, ..., \tilde{w}_{80}$ - расширенный блок, при этом

$$\tilde{w}_j = w_j \text{ для } j = \overline{1,16},$$

$$\tilde{w}_j = Rol(w_{j-3} \oplus w_{j-8} \oplus w_{j-14} \oplus w_{j-16}) \text{ для } j = \overline{17,80},$$

где Rol - операция циклического сдвига на один разряд влево.

Перед началом первого цикла инициализируются пять 32-разрядных переменных: $A = 67452301\text{h}$, $B = \text{EFCDAB89h}$, $C = 98\text{BADCFEh}$, $D = 10325476\text{h}$, $E = \text{C3D2E1F0h}$; при этом вектор инициализации является результатом конкатенации этих переменных

$$h_{SHA0} = A \| B \| C \| D \| E,$$

где $\|$ - символ операции конкатенации.

Конкатенация новых значений этих переменных, полученных в конце i-го цикла, объявляется результатом работы цикла h_{SHAi}.

В начале каждого цикла создаются копии входных переменных: $AA = A$, $BB = B$, $CC = C$, $DD = D$, $EE = E$. Затем выполняется 80 шагов алгоритма, на каждом из которых происходит выполнение следующих операций:

$$Temp = Rol^5 A + f_j(B, C, D) + E + x_{ij} + c_j,\ j = \overline{1,80},\ i = \overline{1,l},$$
$$E = D,\ D = C,$$
$$C = Rol^{30} B,\ A = Temp,$$

где Rol^* - операция циклического сдвига на $*$ разрядов влево, $f_j(\ )$ - шаговая функция, x_{ij} - j-е слово i-го блока x_i, c_j - шаговая константа.

В первом раунде (при $j = \overline{1,20}$) используются функции и константа:

$$f_j(X, Y, Z) = XY \vee \bar{X}Z,\ c_j = 5\text{A827999h},$$

во втором раунде (при $j = \overline{21,40}$) -

$$f_j(X, Y, Z) = X \oplus Y \oplus Z,\ c_j = 6\text{ED9EBA1h},$$

в третьем раунде (при $j = \overline{41,60}$) -

$$f_j(X, Y, Z) = XZ \oplus XY \oplus ZY,\ c_j = 8\text{F1BBCDCh},$$

в четвертом раунде (при $j = \overline{61,80}$) -

$$f_j(X, Y, Z) = X \oplus Y \oplus Z,\ c_j = \text{CA62C1D6h}.$$

Цикл завершается сложением по модулю 2^{32}:

$$A = AA + A,\ B = BB + B,\ C = CC + C,\ D = DD + D,\ E = EE + E,$$

конкатенация полученных значений A, B, C, D, E является результатом работы основного цикла.

Алгоритмы семейства SHA-2 значительно отличаются от версии SHA-1. Рассмотрим алгоритм SHA-256 [9].

Перед хешированием сообщение дополняется до длины, кратной 512 битам, аналогично SHA-1. После этого полученная последовательность разделяется на блоки по 512 бит (16 32-разрядных слов), каждый из которых поступает на вход функции сжатия SHA-256. В этом смысле мы имеем обычную итерационную хеш-функцию.

Функция сжатия имеет похожую итерационную структуру. Функция расширения блока на основе 16 слов исходного сообщения формирует расширенное сообщение - 64 слова, поступающие на вход 64 раундов функции сжатия (РФС) (см. рис. 3.3).

Функция расширения блока MS описывается следующим образом. Первые 16 слов расширенного сообщения соответствуют исходным 16 словам блока, дальнейшие слова формируются по рекуррентной формуле:

$$\sigma_0^{(256)}(x) = ROTR^7(x) \oplus ROTR^{18}(x) \oplus SHR^3(x),$$

$$\sigma_1^{(256)}(x) = ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x),$$

$$w_i = \sigma_1^{(256)}(w_{i-2}) + w_{i-7} + \sigma_0^{(256)}(w_{i-15}) + w_{i-16},$$

где w_i - i-е слово расширенного сообщения, $ROTR^a(x)$ - циклический сдвиг слова x вправо на a позиций, $SHR^b(x)$ - сдвиг слова x вправо на b позиций.

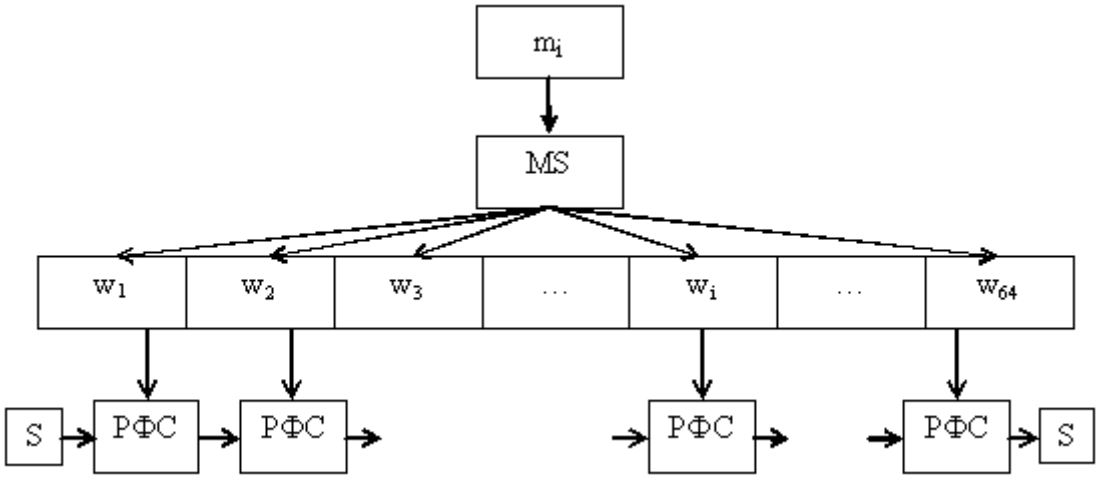

Рис. 3.3. Функция сжатия SHA-256

Структура раунда представлена на рис. 3.4., где приняты следующие обозначения:

$$Ch(X,Y,Z) = XY \sqrt{XZ},$$

$$Maj(X,Y,Z) = XY \sqrt{XZ} \sqrt{YZ},$$

$$\sum_0^{(256)}(x) = ROTR^2(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x),$$

$$\sum_1^{(256)}(x) = ROTR^6(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x).$$

Используется фиксированная последовательность из 64 32-разрядных констант (по количеству раундов). На каждом раунде используется одна константа из этого массива K_i. Более подробно семейство SHA-2 описано в [2,9].

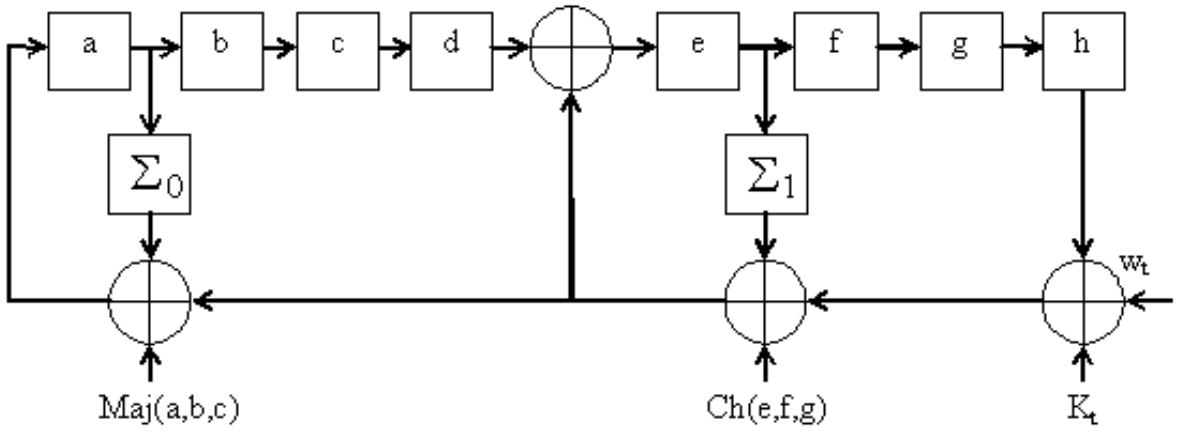

Рис. 3.4. Один раунд функции сжатия SHA-256

К итерационным алгоритмам хеширования относятся ГОСТ Р 34.11-94, MD5 и др.

3.3. Хеш-функции на основе симметричных блочных криптоалгоритмов

При использовании для построения хеш-функции симметричных блочных криптосистем стойкость хеш-функции гарантируется стойкостью применяемой блочной криптосистемы.

Пусть $x = x_1 x_2 x_3 ... x_i ... x_l$, $i = \overline{1,l}$ - последовательность, состоящая из блоков, размер которых равен размеру ключа блочного шифра. Блоки x_i суть результат расширения блоков исходного сообщения меньшей длины.

При использовании для вычисления текущего хеш-значения h_i функции шифрования E получают различные схемы хеш-функций. Наиболее известны следующие схемы:

$$h_i = E(h_{i-1}; x_i) \oplus x_i,$$
$$h_i = E(h_{i-1}; x_i \oplus h_{i-1}) \oplus x_i \oplus h_{i-1};$$
$$h_i = E(h_{i-1}; x_i) \oplus x_i \oplus h_{i-1};$$
$$h_i = E(h_{i-1}; x_i \oplus h_{i-1}) \oplus x_i.$$

Здесь в качестве ключа используется хеш-значение, вычисленное на предыдущем шаге h_{i-1}. Третья формула реализует схему хеширования, которая используется в алгоритме Whirlpool. В качестве симметричной блочной криптосистемы используется алгоритм AES.

В заключение этого раздела необходимо сказать, что в настоящее время в РФ принят стандарт хеширования ГОСТ Р 34.11-2012, проектное название представленного стандартом семейства хеш-функций носит название «Стрибог». Основными отличиями ГОСТ Р 34.11-2012 («Стрибог») и ГОСТ Р 34.11-94 являются:

- размер блоков сообщения и внутреннего состояния в ГОСТ Р 34.11-2012 составляет 512 бит, а в ГОСТ Р 34.11-94 – 256 бит;

- ГОСТ Р 34.11-2012 состоит из двух хеш-функций, с длинами результирующего значения хеш-образа 256 и 512 бит;

- в ГОСТ Р 34.11-94 в качестве функции сжатия используется блочный симметричный криптоалгоритм ГОСТ 28147-89, а в ГОСТ Р 34.11-2012 функция сжатия другая и это - главное отличие. В ГОСТ Р 34.11-2012 функция сжатия состоит из трех основных преобразований: подстановки на байтах, транспонирования матрицы байт, умножения 64-битных векторов на матрицу 64×64.

Результаты тестирования хеш-функций показали, что ГОСТ Р 34.11-2012 практические в два раза быстрее, чем ГОСТ Р 34.11-94. К тому же ГОСТ Р 34.11-2012 проще в реализации и оптимизации как алгоритмически, так и для конкретной платформы.

Контрольные вопросы

1. Дайте определение хеш-функции.
2. Назовите свойства, которыми должна обладать хеш-функция.
3. В чем суть задач: нахождения прообраза, нахождения коллизий, нахождения второго прообраза.
4. Что такое итерационная хеш-функция. Приведете примеры.
5. Поясните методику построения хеш-функций на основе блочных криптоалгоритмов. Приведите примеры.

4. Методы криптоанализа ассиметричных криптосистем

4.1. Методы, основанные на алгоритмах разложения на множители

В этом пункте рассматриваются методы «взлома», основанные на решении задачи факторизации (IFP). Очевидно, что зная разложение (1.9), можно решить задачу RSA путем вычисления обратной функции. Если это разложение неизвестно, то требуется определить такой алгоритм, который позволил бы решить задачу RSA за полиномиальное время, т.е.

$$IFP \overset{\mathrm{P}}{\Rightarrow} RSA,$$

где P - класс задач, решаемых за полиномиальное время детерминированной машиной Тьюринга, $\Rightarrow$ - символ операции импликации.

Самые эффективные алгоритмы факторизации можно разделить на две группы [3,7,8]:

1. Алгоритмы, время выполнения которых зависит главным образом от размера n. К таким алгоритмам относятся: алгоритм Ферма, метод Лемана, метод квадратичных форм Шенкса, метод цепных дробей, метод решета, метод решета числового поля.

2. Алгоритмы, время выполнения которых зависит главным образом от размера множителя p. При выполнении условия $p \le \sqrt{n}$ эффективными алгоритмами факторизации являются: $(p-1)$-метод Полларда, ρ-метод Полларда, метод Ленстры, метод пробного деления.

Кроме того, алгоритмы факторизации можно разделить на детерминированные и недетерминированные, условные и безусловные. Условными являются те алгоритмы, в которых на раскладываемое число n налагаются дополнительные условия, например, n - нечестное составное число длиной β бит, p и q - простые числа длиной $\frac{\beta}{2}$.

Рассмотрим некоторые из перечисленных методов.

Алгоритм факторизации Ферма. Для того, чтобы было сложнее факторизовать n и решить задачу RSA, p и q выбираются одинаковой битовой длины. Однако если p и q слишком близки друг к другу, то становится возможным достаточно быстро найти их. Пусть $n = pq$, где $p \le q$ и оба нечетные, тогда положив $z_1 = \frac{1}{2}(p+q)$ и $z_2 = \frac{1}{2}(q-p)$, получаем, что $n = z_1^2 - z_2^2 = (z_1 - z_2)(z_1 + z_2)$ или $z_2^2 = n - z_1^2$. На этой идее основан алгоритм Ферма. При известном нечетном целом $n > 1$ алгоритм находит наибольший множитель $\le \sqrt{n}$. Существо алгоритма в следующем:

1. Вводится n и $k \leftarrow \lfloor \sqrt{n} \rfloor + 1$, $z_2 \leftarrow k \cdot k - n$, $d \leftarrow 1$.

2. Если $\lfloor \sqrt{z_2} \rfloor = \sqrt{z_2}$, то требуется перейти сразу к четвертому шагу алгоритма, иначе $z_2 \leftarrow z_2 + 2 \cdot k - d$ и $d \leftarrow d + 2$.

3. $\lfloor \sqrt{z_2} \rfloor \le \frac{n}{2}$, то требуется перейти ко второму шагу алгоритма, иначе задача нахождения множителей считается нерешаемой и алгоритм заканчивается.

4. $z_1 \leftarrow \sqrt{n + z_2}$, $z_2 \leftarrow \sqrt{z_2}$. Определяются значения нетривиальных множителей $z_1 - z_2$ и $z_1 + z_2$ числа n.

Алгоритм Ферма эффективен для небольших n. Количество циклов C, необходимых для достижения результата, оценивается неравенством [8]

$$\frac{q-p}{2} - 2\sqrt{n} - 1 < C < \frac{q-p}{2}.$$

Следовательно, для обеспечения стойкости RSA необходимо, чтобы разность $q - p$ была велика.

$(p-1)$-метод Полларда. Простые числа p и q в RSA необходимо выбирать, исходя из тех соображений, чтобы $p \pm 1$, $q \pm 1$ имели по крайней

мере один простой делитель, больший 10^{20} [8], в противном случае p можно эффективно найти, используя $(p-1)$-метод Полларда (или $(p+1)$-метод Вильямса).

Пусть $n>1$ составное число. Рассмотрим алгоритм по шагам.

1. Шаг инициализации. Случайно выбирается $a \in Z_n$, где Z_n - класс остатков по модулю n: $Z_n = Z/nZ = \{1,...,n-1\}$, Z - множество целых чисел. Выбирается положительное целое число k, которое делилось бы на множество простых степеней, например, $k = НОК(2,...,B)$ для соответствующей границы B. Чем больше B, тем более вероятно, что найдется множитель, но тем дольше будет работать алгоритм.

2. Шаг возведения в степень. Вычисляется $a_k \equiv a^k \bmod n$.

3. Шаг вычисления НОД. Вычисляется $f = НОД(a_k - 1, n)$.

4. Шаг проверки условия $1 < f < n$. Если условие выполняется, то f - нетривиальный делитель n и алгоритм заканчивается. Один из множителей n найден. Если условие не выполняется, то f - тривиальный делитель. Тогда требуется перейти к шагу 2, выбрав новое a и/или новое k.

Алгоритм эффективен только при малом B. Сложность алгоритма составляет $B \log B (\log n)^2$.

Пример. Используем $(p-1)$-метод Полларда для факторизации $n = 540143$. Выберем $B = 8$, отсюда $k = НОК(2,3,4,5,6,7,8) = 840$. Выберем $a = 2$. Тогда

$$f = НОД(2^{840} - 1 \bmod 540143, 540143) = НОД(3046, 540143) = 421.$$

Выполняется условие $1 < f < 540143$, т.е. один из множителей n найден. Действительно, $540143 = 421 \cdot 1283$.

4.2. Методы, основанные на алгоритмах вычисления дискретного логарифма

Рассматриваемые в этом пункте методы основаны на решении задачи дискретного логарифмирования (DPL). Если существует возможность решить задачу DPL за полиномиальное время, то тогда за полиномиальное время можно «взломать» и криптосистему, основанную на задаче DPL, например, криптосистему Эль Гамаля:

$$DPL \overset{P}{\Rightarrow} ElGamal.$$

Существуют три принципиально различных класса алгоритмов, которые используются для вычисления дискретного логарифма [3,6,8]:

1. Алгоритмы, которые работают с произвольными группами. Они не используют никаких специальных свойств групп. К этому классу относятся:

алгоритм больших и малых шагов (алгоритм Шенкса), ρ-метод Полларда (аналог ρ-метода факторизации Полларда), λ-метод (метод «кенгуру»).

2. Алгоритмы, работающие с конечными группами, порядок которых не имеет больших простых делителей; точнее сказать, с группами, порядок которых является гладким числом. К этому классу принадлежит хорошо известный алгоритм Сильвера-Полига-Хеллмана, основанный на теореме об остатках.

3. Алгоритмы, которые используют метод представления элементов группы в виде произведения элементов, принадлежащих множеству небольшой мощности. Типичным представителем этого класса являются алгоритмы исчисления порядка Адлемана и NFS алгоритм Говарда.

Рассмотрим алгоритмы каждого класса.

Метод малых и больших шагов. Этот метод впервые был описан Д. Шенксом в 1973 году. Метод является одним из первых, который показал, что решить задачу DPL можно быстрее, чем методом полного перебора. Метод Шенкса широко известен и под другим названием – «шаг младенца, шаг великана». Существо метода заключается в следующем.

Выбирается два целых числа m и k такие, что $mk > p$. Затем вычисляются два ряда чисел

$$\left.\begin{array}{l} y, ay, a^2y, ..., a^{m-1}y; \\ a^m, a^{2m}, ..., a^{km}. \end{array}\right\} \pmod p \qquad (4.1)$$

Из равенства

$$a^{im} = a^j y$$

определяются числа i и j и затем определяется

$$x = im - j. \qquad (4.2)$$

Докажем, что (4.2) справедливо. Действительно,

$$a^x = a^{im-j} = \frac{a^{im}}{a^j} = \frac{a^{im}y}{a^j y} = \frac{a^{im}y}{a^{im}} = y. \pmod p$$

В [6] приведено доказательство существования чисел i и j.

Метод больших и малых шагов назван так по следующей причине. В (4.1) первый числовой ряд увеличивается медленнее второго, так как степень числа a увеличивается на единицу (малый шаг), а во втором числовом ряду степень увеличивается на m (большой шаг).

Пример. Требуется найти решение уравнения $2^x \bmod 23 = 9$. Выберем $m = 6$ и $k = 4$ (24>23). Вычислим два ряда чисел

$$\left.\begin{array}{l} 9, 18, 13, 3, 6, 12; \\ 18. \end{array}\right\} \pmod{23}$$

Таким образом, из равенства $2^{1\cdot 6} = 2^1 \cdot 9$ получаем $i = 1$ и $j = 1$. Тогда $x = 1 \cdot 6 - 1 = 5$.

Алгоритм исчисления порядка. Алгоритм исчисления порядка известен с начала двадцатого века, однако только в 1979 году Адлеман показал, что его можно использовать для решения задачи DPL.

Для описания алгоритма введем некоторые понятия [6].

Определение 4.1. Число называется **гладким**, если оно не имеет больших простых делителей.

Определение 4.2. Число называется p**-гладким**, если оно разлагается на простые множители, меньшие или равные p.

Алгоритм исчисления порядка состоит в следующем. Формируется множество базовых множителей

$$S = \{p_1, p_2, ..., p_t\},$$

состоящее из первых t простых чисел.

Задавая последовательно значения $k = 1,2,3,\ldots$, находятся $t + \varepsilon$ (ε - небольшое целое число) p_t-гладких чисел вида $a^k \bmod p$, гладкость проверяется путем деления на элементы множества S. Каждое из p_t-гладких чисел записывается через произведение базовых множителей

$$a^k \bmod p = \prod_{i=1}^{t} p_i^{c_i}, \; c_i \geq 0. \qquad (4.3)$$

Для каждого k получается свой набор c_i.

Логарифмируем (4.3)

$$k = \sum_{i=1}^{t} c_i \log_a p_i \qquad (4.4)$$

для каждого p_t-гладкого числа. Таким образом, получена система $t + \varepsilon$ уравнений вида (4.3) с t неизвестными. В качестве неизвестных выступают величины $\log_a p_i$, при этом число уравнений на ε больше числа неизвестных, что повышает вероятность получения решения системы уравнений в случае, если некоторые уравнения окажутся линейно зависимыми.

Решая систему уравнений, все вычисления проводят по $\bmod (p-1)$. Напомним, что показатели степени, а следовательно и логарифмы, приводятся по $\bmod (p-1)$. В результате получаем значения логарифмов чисел из множества S: $\log_a p_1$, $\log_a p_2$, …, $\log_a p_t$.

Случайным образом выбирается число r, находим p_t-гладкое число вида $(y \cdot a^r)$

$$y \cdot a^r \bmod p = \prod_{i=1}^{t} p_i^{e_i}, \; e_i \geq 0. \qquad (4.5)$$

Логарифмируя (4.5), получаем конечный результат

$$x = \log_a y = \left(\sum_{i=1}^{t} e_i \log_a p_i - r \right) \mod (p-1). \qquad (4.6)$$

Пример. Требуется решить уравнение $10^x \bmod 47 = 37$. Формируем множество базовых множителей $S = \{2,3,5\}$, тогда $t = 3$.

Пусть $\varepsilon = 1$, для простоты записи вводим обозначения $u_1 = \log_{10} 2 \bmod 47$, $u_2 = \log_{10} 3 \bmod 47$, $u_3 = \log_{10} 5 \bmod 47$.

Проводим поиск 5-гладких чисел:

$$10^1 \bmod 47 = 10 = 2 \cdot 5,$$
$$10^2 \bmod 47 = 6 = 2 \cdot 3,$$
$$10^3 \bmod 47 = 13,$$
$$10^4 \bmod 47 = 36 = 2 \cdot 2 \cdot 3 \cdot 3,$$
$$10^5 \bmod 47 = 31,$$
$$10^6 \bmod 47 = 28 = 2 \cdot 2 \cdot 7,$$
$$10^7 \bmod 47 = 45 = 3 \cdot 3 \cdot 5.$$

Значение k изменялось от 1 до 7. Дальнейший поиск 5-гладких чисел не имеет смысла, т.к. определены $t + \varepsilon = 4$ 5-гладких числа, соответствующие степеням 1,2,4 и 7.

Прологарифмировав полученные уравнения, и, с учетом введенных обозначений, имеем систему уравнений:

$$\begin{cases} u_1 + u_3 = 1, \\ u_1 + u_2 = 2, \\ 2u_1 + 2u_2 = 4, \\ 2u_1 + u_3 = 7. \end{cases}$$

Исключая линейно зависимое уравнение и решая систему, получаем

$$u_2 = \frac{8}{3} \bmod 46 = 8 \cdot 3^{-1} \bmod 46 = 8 \cdot 31 \bmod 46 = 18,$$
$$u_1 = 2 - u_2 = (2 - 18) \bmod 46 = -16 \bmod 46 = 30,$$
$$u_3 = u_2 - 1 = (18 - 1) \bmod 46 = 17.$$

Таким образом, в результате вычислений определены логарифмы чисел $S = \{2,3,5\}$. В соответствии с (4.5) и начиная с $r = 3$, вычисляем

$$37 \cdot 10^3 \bmod 47 = 11,$$
$$37 \cdot 10^4 \bmod 47 = 37 \cdot 36 \bmod 47 = 16 = 2 \cdot 2 \cdot 2 \cdot 2.$$

В последнем равенстве переходим к логарифмам и получаем конечный результат

$$x = \log_{10} 37 = 4 \log_{10} 2 - 4 = (4 \cdot 30 - 4) \bmod 46 = 24.$$

Алгоритм Сильвера-Полига-Хеллмана. В 1978 году Полиг и Хеллман предложили алгоритм для решения задачи DPL. Суть алгоритма состоит в следующем [8].

Число $\varphi(p-1)$ раскладывается на простые множители

$$\varphi(p-1) = \prod_{i=1}^{k} g_i^{\alpha_i},$$

где g_i - различные простые числа, α_i - натуральные числа.

Затем составляется таблица $r_{g_i,j}$ по правилу

$$r_{g_i,j} = a^{\frac{j\varphi(p-1)}{g_i}} \bmod p,\ 0 \le j < g_i. \qquad (4.7)$$

Аналогично алгоритму больших и малых шагов определяются отдельные дискретные логарифмы $x \bmod g_i^{\alpha_i}$. Для этого рассматривают представление $x \bmod g_i^{\alpha_i}$ по базе g_i

$$x \bmod g_i^{\alpha_i} = x_0 + x_1 g_1 + \ldots + x_{\alpha_i - 1} g^{\alpha_i - 1},\ 0 \le x_n < g_i - 1. \qquad (4.8)$$

Для нахождения x_0 вычислить $y^{\frac{\varphi(p-1)}{g_i}}$, что эквивалентно $r_{g_i,j}$ для некоторого j, и положить $x_0 = j$. Это возможно, т.к.

$$y^{\frac{\varphi(p-1)}{g_i}} \equiv a^{\frac{x\varphi(p-1)}{g}} \equiv a^{\frac{x_0\varphi(p-1)}{g}} \bmod p = r_{g_i,x_0}.$$

Для нахождения x_1 вычисляется $y_1 = ya^{-x_0}$. Если

$$y_1^{\frac{\varphi(p-1)}{g_i^2}} \bmod p = r_{g_i,j},$$

то $x_1 = j$. Это возможно, т.к.

$$y_1^{\frac{\varphi(p-1)}{g_i^2}} \equiv a^{\frac{(x-x_0)\varphi(p-1)}{g_i^2}} \equiv a^{\frac{(x_1+x_2g_2+\ldots)\varphi(p-1)}{g_i}} \equiv a^{\frac{x_1\varphi(p-1)}{g}} \bmod p = r_{g_i,x_1}.$$

Для нахождения x_2 вычисляется $y_2 = ya^{-x_0-x_1g_i}$ и $y_2^{\frac{\varphi(p-1)}{g_i^3}}$.

Таким же образом итеративно вычисляются x_3, x_4, …, $x_{\alpha-1}$. Далее, используя китайскую теорему об остатках, определяется значение x.

Пример. Требуется решить уравнение $x = \log_2 62 \bmod 181$.

Разложим $\varphi(p-1)$ на простые множители: $180 = 2^2 \cdot 3^2 \cdot 5$. Используя формулу (4.7), составляется таблица.

Вычисляем $r_{g_1,j} = a^{\frac{j(p-1)}{g_1}} \bmod 181 = 2^{90j} \bmod 181$ для $0 \le j < 2$:

$$r_{2,0} = 2^{90\cdot 0} \bmod 181 = 1,$$

$$r_{2,1} = 2^{90\cdot 1} \bmod 181 = 180.$$

Вычисляем $r_{g_2,j} = a^{\frac{j(p-1)}{g_2}} \bmod 181 = 2^{60j} \bmod 181$ для $0 \le j < 3$:

$$r_{3,0} = 2^{60\cdot 0} \bmod 181 = 1,$$

$$r_{3,1} = 2^{60\cdot 1} \bmod 181 = 48,$$

$$r_{3,2} = 2^{60\cdot 2} \bmod 181 = 132.$$

Вычисляем $r_{g_3,j} = a^{\frac{j(p-1)}{g_3}} \bmod 181 = 2^{36j} \bmod 181$ для $0 \le j < 5$:

$$r_{5,0} = 2^{36\cdot 0} \bmod 181 = 1,$$

$$r_{5,1} = 2^{36\cdot 1} \bmod 181 = 59,$$

$$r_{5,2} = 2^{36\cdot 2} \bmod 181 = 42,$$

$$r_{5,3} = 2^{36\cdot 3} \bmod 181 = 125,$$

$$r_{5,4} = 2^{36\cdot 4} \bmod 181 = 135.$$

Таким образом, получаем таблицу (см. табл. 4.1).

Таблица 4.1

	j				
g_i	0	1	2	3	4
2	1	180			
3	1	48	132		
4	1	59	42	125	135

Составлять таблицу имеет смысл только если все g_i малы.

Затем определяем дискретные логарифмы $x \bmod g_i^{\alpha_i}$ (4.8). Найдем дискретный логарифм $x \bmod g_1^{\alpha_1}$, то есть $x \bmod 2^2$:

$$x \bmod 181 \Leftrightarrow x \bmod 2^2 = x_0 + 2x_1.$$

Здесь $\Leftrightarrow$ - символ эквивалентности.

Для нахождения x_0 вычислим

$$a^{\frac{(p-1)}{g_1}} \bmod p = 62^{\frac{180}{2}} \bmod 181 = 1 = r_{2,0},$$

следовательно, $x_0 = 0$.

Для нахождения x_1 сначала вычисляем $y_1 = ya^{-x_0} = 62 \cdot a^0 = 62$, а затем

$$y_1^{\frac{(p-1)}{g_1^2}} \bmod p = 62^{\frac{180}{4}} \bmod 181 = 1 = r_{2,0},$$

следовательно, $x_0 = 1$.

Тогда $x \bmod 2^2 = x_0 + 2x_1 \Leftrightarrow x \bmod 4 = 0$.

Аналогичным образом вычисляются дискретные логарифмы $x \bmod 3^2$ и $x \bmod 5^1$.

В результате получаем систему уравнений

$$\begin{cases} x \bmod 2^2 = 0, \\ x \bmod 3^2 = 1, \\ x \bmod 5 = 0. \end{cases}$$

Решение системы уравнений находится с помощью китайской теоремы об остатках. Системе удовлетворяет единственное решение $x = 100$.

В настоящее время самым быстрым и эффективным алгоритмом решения задачи DPL является алгоритм NFS (Number Field Sieve). Именно этот алгоритм диктует условия выбора длин модулей криптосистем, стойкость которых основана на задаче DPL.

В табл. 4.2 приведены результаты сравнительного анализа трудоемкости решения задачи DPL.

Таблица 4.2

Трудоемкость решения задачи DPL

Метод	**Трудоемкость**
Полный перебор	$\frac{p}{2}$
Метод больших и малых шагов	$2\sqrt{p}$
Алгоритм исчисления порядка	$c_1 \cdot 2^{c_2\sqrt{\log p \log\log p}}$
Алгоритм Сильвера-Полига-Хеллмана	$\sum_{i=1}^{k}\left(\log p + g_i^{1-r_i}\left(+\log g_i^{r_i}\right)\right.$

Контрольные вопросы

1. Назовите группы эффективных алгоритмов факторизации.

2. Опишите алгоритм Ферма. Какие требования он предъявляет к стойкости ассиметричных криптосистем?

3. Опишите $p-1$-метод Полларда. Какие требования он предъявляет к стойкости ассиметричных криптосистем?

4. Назовите классы алгоритмов, которые используются для вычисления дискретного логарифма.

5. В чем суть метода больших и малых шагов?

6. Опишите алгоритм исчисления порядка. Дайте определение гладкости чисел.

7. Опишите алгоритм Сильвера-Полига-Хеллмана. Какие требования он предъявляет к стойкости ассиметричных криптосистем?

8. Проведите сравнительный анализ методов криптоанализа ассиметричных криптосистем.

5. Методы аутентификации

Для предотвращения работы с системой незаконных пользователей необходима процедура распознавания каждого законного пользователя (или групп пользователей). Для этого в системе хранится информация, по которой можно опознать пользователя, а пользователь при входе в систему, при выполнении определенных действий, при доступе к ресурсам обязан себя **идентифицировать**, т.е. указать идентификатор, присвоенный ему в данной системе. Получив идентификатор, система проводит его **аутентификацию,** т.е. проверяет его подлинность - принадлежность множеству идентификаторов. Если бы идентификация не дополнялась аутентификацией, то сама идентификация теряла бы всякий смысл. Обычно устанавливается ограничение на число попыток предъявления некорректного идентификатора.

Аутентификация пользователя основана на следующих принципах [9]:

- на предъявлении пользователем пароля;
- на предъявлении пользователем доказательств, что он обладает секретной ключевой информацией;
- на ответах на некоторые тестовые вопросы;
- на предъявлении пользователем некоторых неизменных признаков, неразрывно связанных с ним;
- на предоставлении доказательств того, что он находится в определенном месте в определенное время;
- на установлении подлинности пользователя некоторой третьей доверенной стороной.

Процедуры аутентификации должны быть устойчивы к подлогу, подбору и подделке.

После распознавания пользователя система должна выяснить, какие права предоставлены этому пользователю, какую информацию он может использовать и каким образом (читать, записывать, модифицировать или

удалять), какие программы может выполнять, какие ресурсы ему доступны, а также другие вопросы подобного рода. Этот процесс называется **авторизацией.** Таким образом, вход пользователя в систему состоит из идентификации, аутентификации и авторизации. В процессе дальнейшей работы иногда может появиться необходимость дополнительной авторизации в отношении каких-либо действий.

Методы аутентификации можно разделить в зависимости от целей аутентификации. Если целью аутентификации является доказательство подлинности предъявленного субъектом идентификатора, то такая аутентификация является **аутентификацией субъекта**. Если в процессе аутентификации проверяется подлинность идентификатора, представленного с некоторыми данными, то такая аутентификация является а**утентификацией объекта.** В отличие от аутентификации субъекта в этой ситуации субъекту не нужно быть активным участником процесса аутентификации.

Рассмотрим методы аутентификации субъекта и объекта [9].

5.1. Аутентификация субъекта

Классическим средством аутентификации субъекта являются парольные схемы. При этом для устранения последствий несанкционированного доступа злоумышленника к информации, хранящейся в памяти системы, передаваться может не сам пароль pw (password), а его хеш-образ $h_{pw} = h(pw)$ (см. рис. 5.1). Функция h_{pw} в этой ситуации может быть определена как:

$$h_{pw} = h(pw) = E_{pw}(ID),$$

если длина пароля и длина ключа функции шифрования E одинаковы $|pw| = |k|$, или как

$$h_{pw} = h(pw) = E_{pw \oplus k}(ID),$$

если длина пароля меньше длины ключа $|pw| < |k|$.

Верификатор системы заранее вычисляет значения $h_{pw} = h(pw)$ и для каждого идентификатора ID хранит эти значения.

Пользователь, прошедший идентификацию, вводит пароль pw^*, затем вычисляет его хеш-образ $h^*_{pw} = h(pw^*)$, а верификатор системы проверяет выполнение условия $pw = pw^*$. При выполнении условия принимается решение о правильности пароля и разрешается доступ пользователя в систему.

Вместе с тем такая схема не позволяет защищаться от злоумышленника, который может передавать информацию, подключаясь непосредственно к линии связи. В этом случае применяется другая схема (см. рис. 5.2.). В данной

схеме процедура вычисления $h^*_{pw} = h(pw^*)$ возложена на верификатора системы.

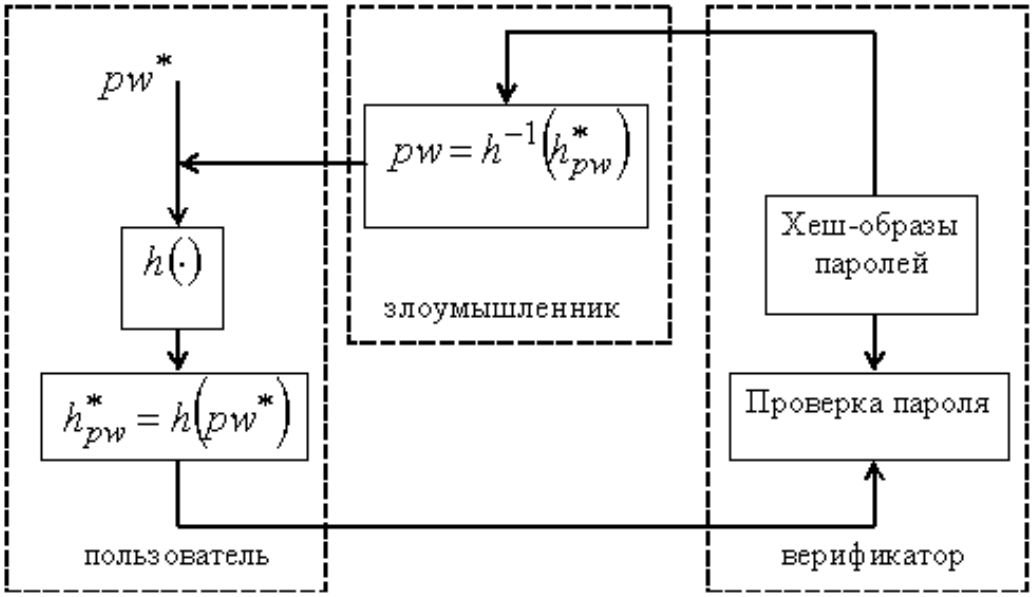

Рис. 5.1. Парольная схема аутентификации (вариант 1)

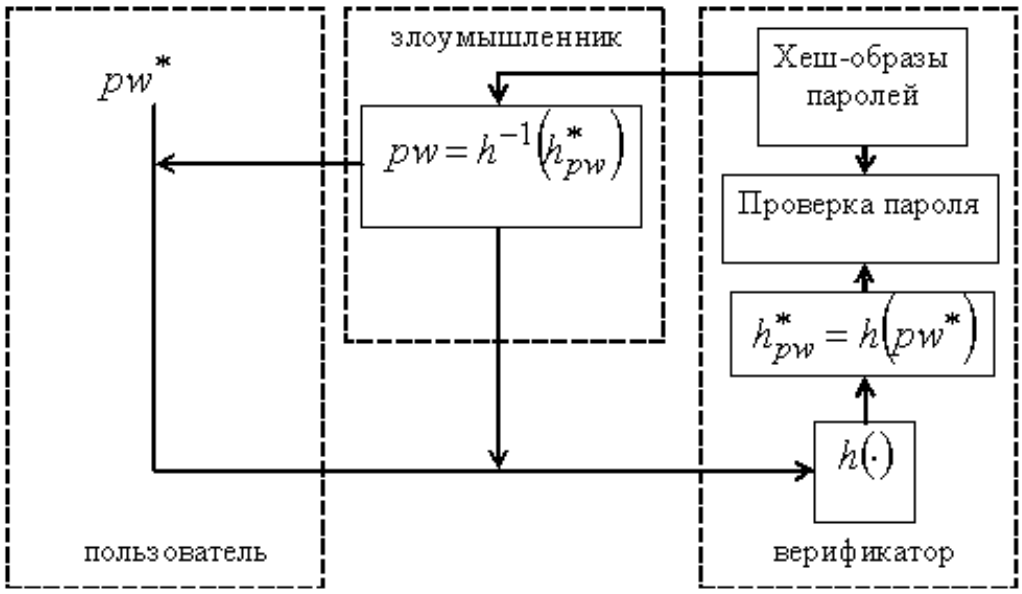

Рис. 5.2. Парольная схема аутентификации (вариант 2)

Обе рассмотренные схемы парольной аутентификации не защищают от атак перехвата и повтора, когда злоумышленник записывает информацию, передаваемую пользователем, и организует ее повторение для входа в систему.

Для устранения последствий перехвата информации, передаваемой пользователем, или несанкционированного доступа злоумышленника к

информации, хранящейся в памяти верификатора, может быть рекомендована схема, представленная на рис. 5.3, предполагающая использование двух хеш-функций h_1 и h_2, причем результат работы второй из них зависит от неповторяющегося блока данных RB.

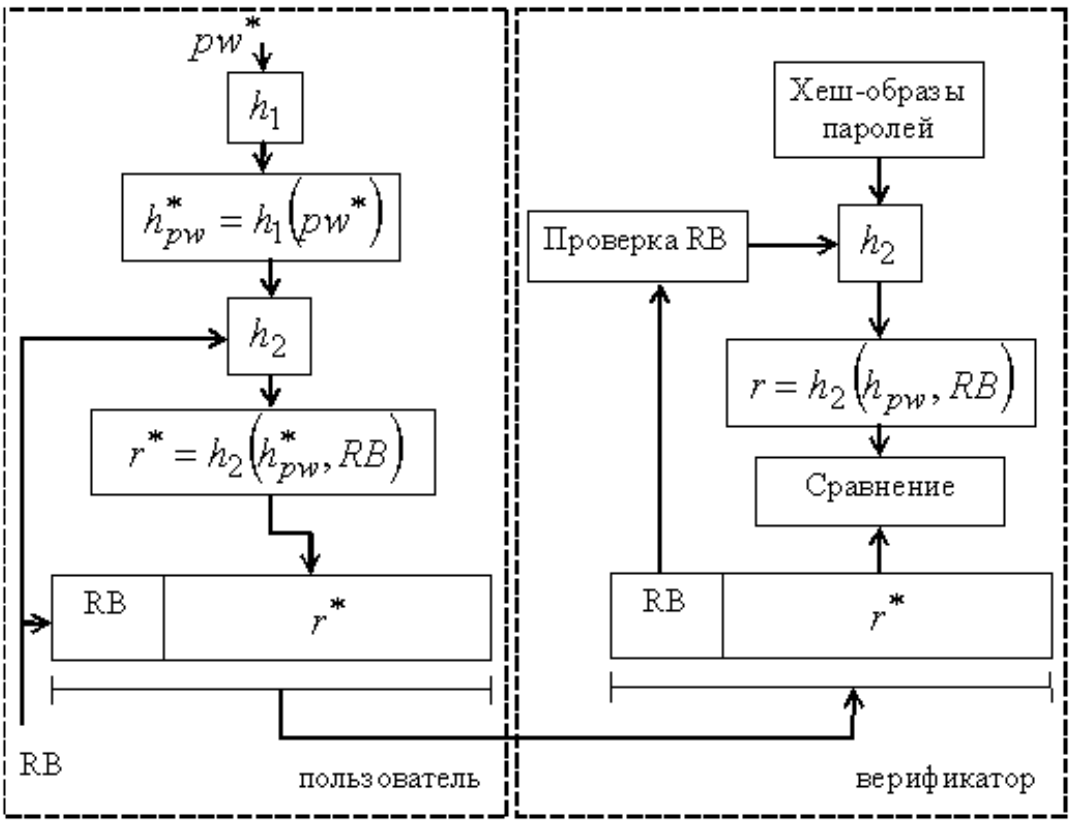

Рис. 5.3. Парольная схема аутентификации, защищенная от повтора

Аутентификация субъекта может быть как **односторонней**, так и **взаимной**. В первом случае процедуру аутентификации проходит один субъект, во втором случае аутентифицируют друг друга два взаимодействующих субъекта, например, связывающиеся между собой по линиям связи. Взаимная аутентификация не есть простое объединение двух сеансов односторонней аутентификации, так как в последнем случае противник легко может осуществить атаку перехвата и повтора, выдавая себя за верификатора перед пользователем и за пользователя перед верификатором.

Проверка подлинности предполагает применение неповторяющихся блоков данных, в качестве которых используются временные метки, механизмы запрос-ответ и процедуры рукопожатия (см. рис. 5.4).

Использование **меток времени** позволяет регистрировать время отправки конкретного сообщения, что дает возможность получателю определить, насколько «устарело» пришедшее сообщение, а значит защититься от повтора. При использовании меток времени возникает проблема допустимого времени

задержки, связанная, во-первых, с невозможностью мгновенной передачи сообщения, а, во-вторых, с невозможностью абсолютной синхронизации хода часов получателя и отправителя.

Механизм **запрос-ответ** предполагает включение пользователем А в сообщение для пользователя В запроса x_A - некоторого случайного числа. Перед ответом пользователь В обязан выполнить над числом x_A некоторую операцию, например, вычислить хеш-образ $h(x_A)$. Получив ответ с правильным результатом вычислений, пользователь А может быть уверен, что В - подлинный.

Процедура **рукопожатия** заключается во взаимной проверке ключей, используемых субъектами взаимодействия. Последние признают друг друга законными партнерами, если докажут друг другу, что обладают правильными ключами.

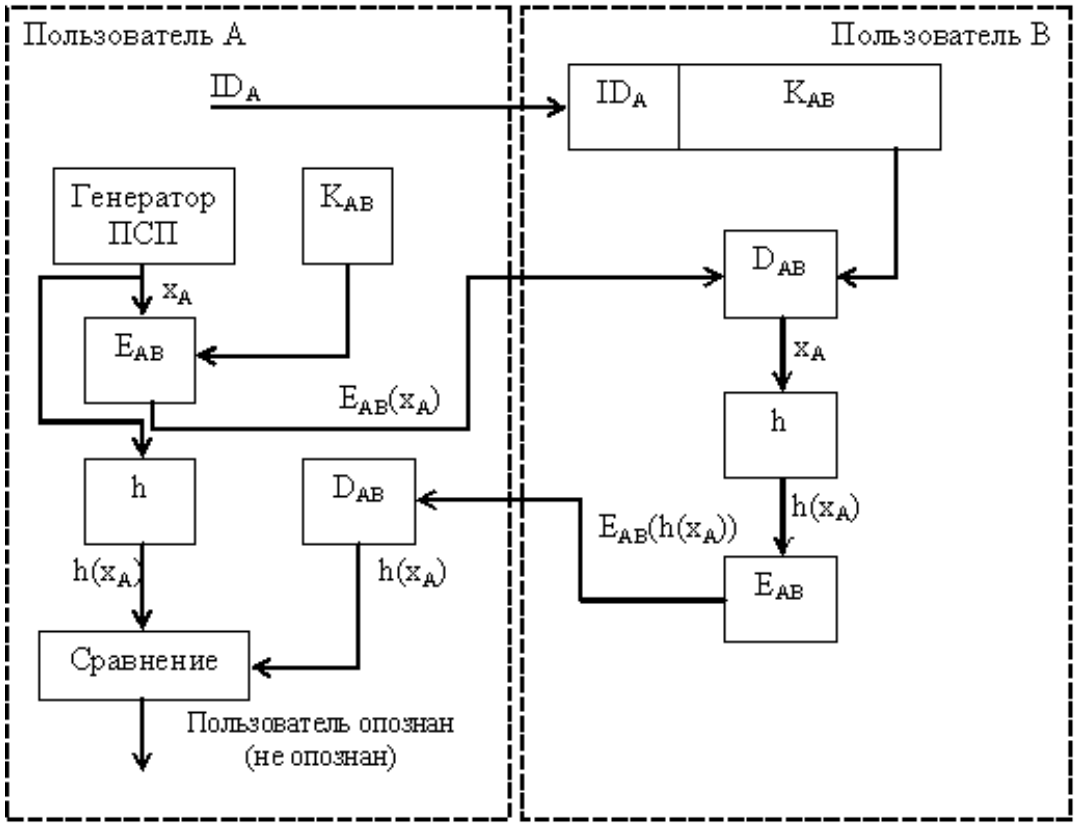

Рис. 5.4. Парольная схема с механизмом запрос-ответ и рукопожатием

Аутентификация субъекта может осуществляться с помощью симметричных и ассиметричных криптосистем.

Аутентификация с использованием симметричных криптосистем в системах с большим числом пользователей требует введения в сеанс связи доверенной стороны. Такая схема аутентификации носит название **схемы аутентификации Нидхэма-Шредера** (см. рис. 5.5). На рисунке показана

процедура взаимной аутентификации двух пользователей A и B с использованием третьей доверенной стороны – C, обладающей секретными ключами k_{AC} и k_{BC}.

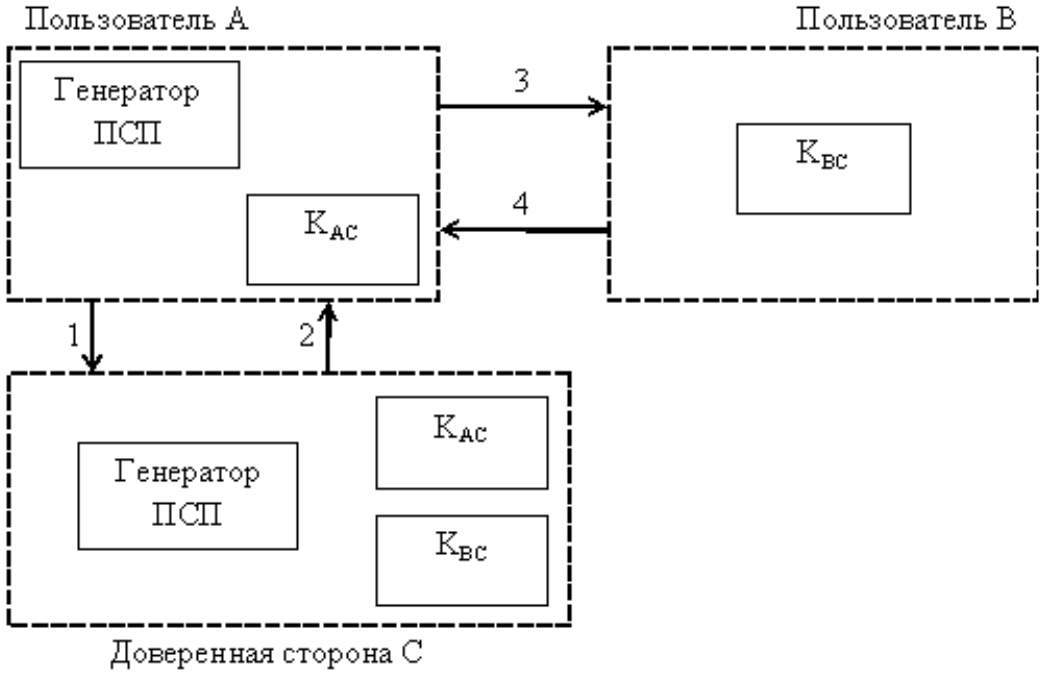

Рис. 5.5. Схема аутентификации Нидхэма-Шредера

Последовательность шагов процедуры следующая:

1) пользователь A, который хочет взаимодействовать с пользователем B, посылает C сообщение, содержащее идентификаторы субъектов запрашиваемого взаимодействия

$$ID_A, ID_B;$$

2) C, получив сообщение, формирует сеансовый ключ k_{AB} для взаимодействия пользователей A и B и посылает A зашифрованное сообщение

$$E_{k_{AC}}(ID_B, k_{AB}, E_{k_{BC}}(ID_A, k_{AB})),$$

содержащее сеансовый ключ для работы с B и криптограмму, которая по сути является разрешением для A на работу с B;

3) пользователь A, расшифровав полученное сообщение, определяет ключ k_{AB} и разрешение

$$E_{k_{BC}}(ID_A, k_{AB}),$$

которое он расшифровать не может, так как не знает ключа k_{BC}; после этого A отправляет B сообщение

$$E_{k_{AC}}(ID_A, x_A), E_{k_{BC}}(ID_A, k_{AB}),$$

содержащее зашифрованный запрос x_A и разрешение, полученное от C;

4) B, расшифровав криптограмму

$$E_{k_{BC}}(ID_A, k_{AB}),$$

узнает идентификатор пользователя взаимодействия и сеансовый ключ k_{AB} для работы с ним, читает запрос x_A, после этого В формирует ответ на запрос $h(x_A)$ и отправляет А сообщение

$$E_{k_{AB}}(ID_B, h(x_A));$$

5) пользователь А, получив сообщение, расшифровывает его и проверяет ответ В; в случае положительного результата проверки процесс аутентификации успешно завершается.

В качестве третьей доверительной стороны выступает центр генерации и распределения ключей (ЦГРК). Данная трехсторонняя схема, по сути, является аутентифицированной схемой распределения сеансовых колючей. ЦГРК должен обладать высокой защищенностью, т.к. его компрометация приводит к компрометации всей сети.

Рассмотрим двусторонние схемы аутентифицированного распределения ключей.

Пусть пользователи А и В располагают заранее известным ключом k_{AB}, используемым не для шифрования, а для пересылки ключевой информации. Передача разового сеансового ключа k от А к В осуществляется с помощью пересылки криптограммы $E_{k_{AB}}(I)$, где

$$I = (k, t, ID_B).$$

Здесь t - метка времени, ID_B - идентификатор абонента В.

Для аутентификации сеанса можно использовать следующую схему, основанную на методе запрос-ответ.

Пользователь В генерирует случайное число x_B и отправляет его пользователю А. Пользователь А вычисляет криптограмму

$$E_{k_{AB}}(\ ), \text{ где } I = (k, x_B, ID_B)$$

и отправляет ее В, который расшифровав криптограмму убеждается, что имеет дело с пользователем А.

Взаимная аутентификация сеанса может быть осуществлена по протоколу, являющемуся модификацией предыдущего.

Пусть k_A и k_B - случайные числа, сгенерированные пользователями А и В. Последовательность шагов процедуры следующая:

1) пользователь В генерирует случайное число x_B и отправляет его абоненту А (впоследствии абонент А должен предъявить это число абоненту В);

2) пользователь А генерирует случайное число x_A (впоследствии абонент В должен предъявить это число абоненту А), вычисляет криптограмму

$$E_{k_{AB}}(\ ), \text{ где } I = (k_A, x_A, x_B, ID_B)$$

и отправляет ее пользователю В. Последний расшифровывает криптограмму, и тем самым, А предъявляет число x_B абоненту В;

3) пользователь B вычисляет криптограмму

$$E_{k_{AB}}(J), \text{ где } J = (k_B, x_A, x_B, ID_A)$$

и отправляет ее абоненту A. Последний расшифровывает криптограмму и, тем самым, B предъявляет число x_A абоненту A;

4) каждая из сторон вычисляет сеансовый ключ с помощью заданной функции

$$k = f(k_A, k_B).$$

Ни один из пользователей заранее не знает сеансового ключа.

Аутентификация с использованием ассиметричных криптосистем.

Использование ассиметричных криптосистем позволяет отказаться от серверов аутентификации, однако в системе должен существовать сервер, выдающий сертификаты на используемые пользователями сети открытые ключи. **Сертификатом** принято называть электронный документ, удостоверяющий принадлежность данного открытого ключа данному пользователю сети, иначе говоря, аутентичность ключа.

Классическим примером несимметричной аутентификации может служить схема Диффи-Хэллмана, представляющая собой совокупность процедуры выработки общего секретного ключа и взаимной аутентификации пользователей сети. **Схема Диффи-Хеллмана** была предложена авторами в середине 70-х годов прошлого века и привела к настоящей революции в криптографии и ее практических применениях.

При использовании симметричных криптосистем для сети, имеющей N абонентов, причем N - достаточно большое число, количество секретных ключей должно быть $\approx \frac{N^2}{2}$. Таким образом система, обеспечивающая сеть ключами, является достаточно громоздкой и дорогостоящей. Диффи и Хеллман решили эту проблему за счет использования открытого распределения и вычисления ключей.

Пусть система связи состоит из трех пользователей A, B и C. Для организации системы связи выбирается большое простое число p и некоторое число g, $1 < g < p-1$ такое, что все числа из множества $\{1,2,3,..., p-1\}$ могут быть представлены как различные степени $g \bmod p$. Другими словами g - примитивный элемент поля Галуа $GF(p)$. Числа p и g известны всем пользователям.

Пользователи выбирают числа $k_{зA}$, $k_{зB}$, $k_{зC}$, которые являются закрытыми ключами. Затем пользователи вычисляют открытые ключи:

$$k_{oA} = g^{k_{зA}} \bmod p,\ k_{oB} = g^{k_{зB}} \bmod p,\ k_{oC} = g^{k_{зC}} \bmod p.$$

Пусть пользователь A решил организовать сеанс связи с B. Пользователь A сообщает B по открытому каналу, что он хочет передать ему сообщение. Затем пользователь A вычисляет:

$$Z_{AB} = \left(k_{oB}\right)^{k_{зA}} \bmod p.$$

В свою очередь, абонент В вычисляет число

$$Z_{BA} = \left(k_{oA}\right)^{k_{зB}} \bmod p.$$

Утверждение 4.1. $Z_{AB} = Z_{BA}$.

□ Действительно, $Z_{AB} = \left(k_{oB}\right)^{k_{зA}} \bmod p = \left(g^{k_{зB}}\right)^{k_{зA}} \bmod p =$

$$= g^{k_{зA}k_{зB}} \bmod p = \left(k_{oA}\right)^{k_{зB}} \bmod p = Z_{BA}.\blacksquare$$

Таким образом, пользователи А и В получают одно и то же число $k = Z_{AB} = Z_{BA}$, являющееся сеансовым ключом, причем это число не передавалось по линии связи.

Рассмотренный протокол не является аутентичным. Пользователи никак не подтверждают подлинность друг друга. Злоумышленник может замаскироваться под одного из пользователей системы, предъявив свой открытый ключ. Рассмотрим аутентичную схему распределения ключей Диффи-Хеллмана:

1) пользователь А вырабатывает случайное число x_A и отправляет пользователю В сообщение

$$g^{x_A} \bmod p;$$

2) пользователь В вырабатывает случайное число x_B, вычисляет

$$g^{x_B} \bmod p,$$

и на своем закрытом ключе создает подпись

$$S_B\left(g^{x_A} \bmod p, g^{x_B} \bmod p\right)$$

сообщения $\left(g^{x_A} \bmod p, g^{x_B} \bmod p\right)$. Затем В вычисляет сеансовый ключ

$$k_{AB} = g^{x_A x_B} \bmod p,$$

зашифровывает подпись на этом ключе и отправляет А сообщение

$$\left\langle g^{x_B} \bmod p; E_{k_{AB}}\left(S_B\left(g^{x_A} \bmod p, g^{x_B} \bmod p\right)\right)\right\rangle;$$

3) пользователь А вычисляет сеансовый ключ

$$k_{AB} = g^{x_A x_B} \bmod p,$$

с помощью своего секретного ключа создает подпись

$$S_A\left(g^{x_A} \bmod p, g^{x_B} \bmod p\right),$$

зашифровывает ее и отправляет В сообщение

$$\left\langle E_{k_{AB}}\left(S_A\left(g^{x_A} \bmod p, g^{x_B} \bmod p\right)\right)\right\rangle.$$

Если проверка подписи В абонентом А завершилась успешно, т.е. А убедился в справедливости равенства

$S_B^{-1}\left(E_{k_{AB}}\left(S_B\left(g^{x_A} \bmod p, g^{x_B} \bmod p\right)\right)\right) = \left(g^{x_A} \bmod p, g^{x_B} \bmod p\right)$,

он может быть уверен в подлинности пользователя В. Если проверка подписи А пользователем В завершилась успешно, т.е. В убедился в справедливости равенства

$S_A^{-1}\left(E_{k_{AB}}\left(S_A\left(g^{x_A} \bmod p, g^{x_B} \bmod p\right)\right)\right) = \left(g^{x_A} \bmod p, g^{x_B} \bmod p\right)$,

он в свою очередь может быть уверен в подлинности пользователя А.

Одним из наиболее эффективных протоколов аутентификации является **протокол Шнорра**.

Пусть p и q - простые числа, такие, что q делит $(p-1)$. Шнорр предлагал использовать p разрядностью не менее 512 битов и q разрядностью не менее 140 битов. Пусть $g \in Z_p$, Z_p - множество целых чисел от 0 до $(p-1)$ такое, что

$$g^q \bmod p = 1, g \neq 1.$$

Протокол Шнорра основан на проблеме дискретного логарифма. В качестве закрытого ключа пользователь выбирает некоторое случайное число $k_з \in Z_p$, а открытый ключ вычисляет $k_o = g^{k_з} \bmod p$.

Схема аутентификации Шнорра состоит в следующем:

1) пользователь А выбирает случайное число $x_A \in Z_p$, вычисляет

$$y_A = g^{x_A} \bmod p$$

и посылает его пользователю В;

2) пользователь В выбирает случайное t-разрядное число $x_B \in \{0,1,..., (2^t - 1)\}$ и посылает его пользователю А;

3) пользователь А вычисляет

$$s = x_A + k_{зA} x_b \bmod q$$

и посылает его пользователю В;

4) пользователь В проверяет соотношение

$$y_A = g^s (k_{oA})^{x_B} \bmod p,$$

и, если оно выполняется, признает подлинность пользователя А.

Злоумышленник, знающий только открытый ключ k_o, p, q и g, не может пройти аутентификацию. Шнорр рекомендовал использовать t разрядностью не менее 72 битов.

5.2. Аутентификация объекта

В процессе аутентификации объекта, иногда называемой аутентификацией источника данных, проверяется подлинность идентификатора, представленного с некоторыми данными. В отличие от

аутентификации субъекта в этой ситуации пользователю не нужно быть активным участником процесса аутентификации. Данный тип аутентификации (см. рис. 5.6) по сути ничем не отличается от процедуры контроля целостности. Для аутентификации объекта применяется шифрование симметричным алгоритмом, выработка имитовставки или электронной подписи. Первые два варианта применяются в том случае, когда пользователь и верификатор доверяют друг другу. Если необходимо иметь возможность доказательства подлинности идентификатора третьей стороне (при условии, что верификатор не имеет возможности изменить массив данных), например, необходима юридическая значимость пересылаемых электронных документов, требуется электронная подпись.

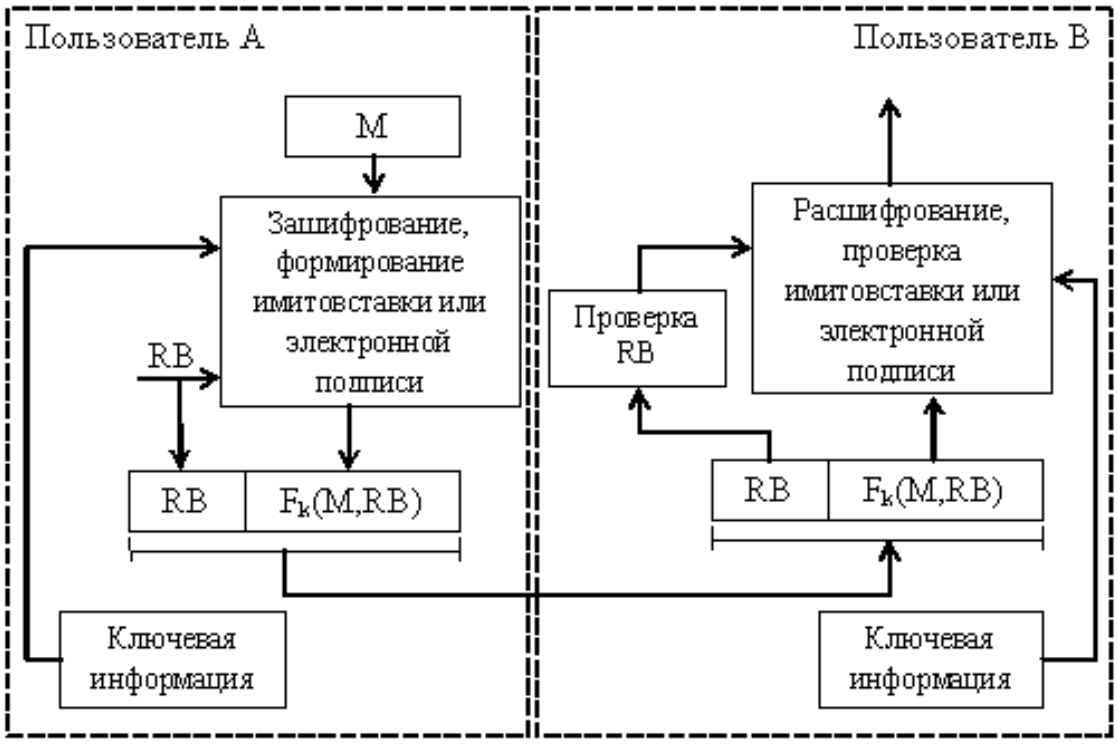

Рис. 5.6. Схема аутентификации объекта

6. Контроль целостности информации

6.1. Имитозащита информации

На жизненном цикле информация подвергается случайным и умышленным деструктивным воздействиям. Для обнаружения случайных искажений информации применяются корректирующие коды, которые в некоторых случаях позволяют не только зафиксировать факт наличия искажений информации, но локализовать и исправить эти искажения. Умышленные деструктивные воздействия чаще всего имеют место при хранении информации в памяти компьютера и при ее передаче по каналам

связи. При этом полностью исключить возможность несанкционированных изменений в массивах данных не представляется возможным. Следовательно, крайне важно оперативно обнаружить такие изменения, так как в этом случае ущерб, нанесенный законным пользователям, будет минимальным.

Как правило, целью злоумышленника, навязывающего ложную информацию, является выдача ее за подлинную, поэтому своевременная фиксация факта наличия искажений в массиве данных сводит на нет все усилия злоумышленника.

Под **имитозащитой** понимают не только исключение возможности несанкционированных изменений информации, а совокупность методов, позволяющих достоверно зафиксировать факты изменений, если они имели место.

Для обнаружения искажений в распоряжении законного пользователя (например, получателя информации при ее передаче) должна быть некая процедура проверки $F(x)$, дающая на выходе 1, если в массиве данных x отсутствуют искажения, или 0, если такие искажения имеют место. Идеальная процедура такой проверки должна обладать следующими свойствами [9]:

- невозможно найти такое сообщение x' способом, более эффективным, чем полный перебор по множеству допустимых значений x (такая возможность в распоряжении противника имеется всегда);
- вероятность успешно пройти проверку у случайно выбранного сообщения x' не должна превышать заранее установленного значения.

Учитывая, что в общем случае все возможные значения x могут являться допустимыми, второе требование - внесения избыточности в защищаемый массив данных. При этом чем больше разница между размером преобразованного избыточного x' и размером исходного x массива, тем меньше вероятность принять искаженные данные за подлинные.

На рис. 6.1-6.3 показаны некоторые возможные варианты внесения такой избыточности. В роли неповторяющегося блока данных RB могут выступать метка времени, порядковый номер сообщения и т.п. В роли контрольного кода могут выступать имитовставки или электронная подпись. **Имитовставкой** принято называть контрольный код, который формируется и проверяется с помощью одного и того же секретного ключа.

Использование блока *RB* позволяет контролировать целостность потока сообщений, защищая от повтора, задержки, переупорядочивания или их утраты. При использовании в качестве *RB* порядкового номера получатель, получив $(i+1)$-е сообщение, проверяет равенство $RB_{i+1} = RB_i + 1$, т.е. что его номер на единицу больше номера предыдущего i-го сообщения. При использовании в качестве *RB* метки времени получатель контролирует, чтобы времена отправки и приема сообщений соответствовали друг другу с учетом задержки в канале связи и разности показаний часов отправителя и получателя.

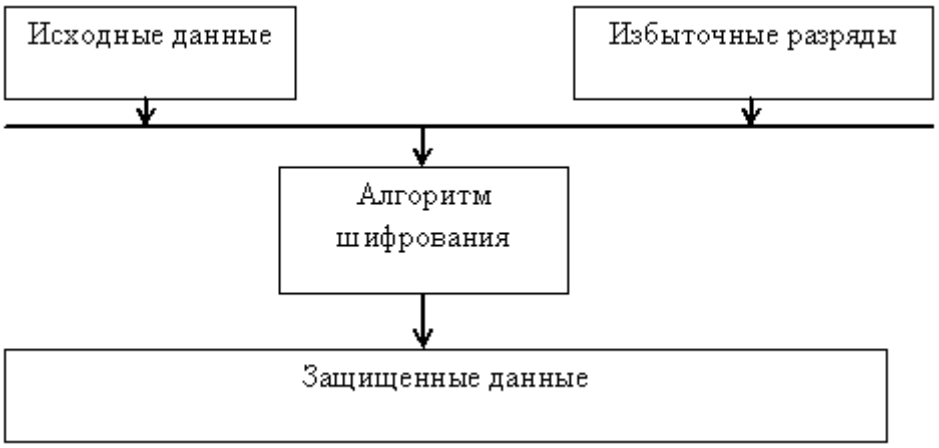

Рис. 6.1. Схема контроля целостности с использованием шифрования

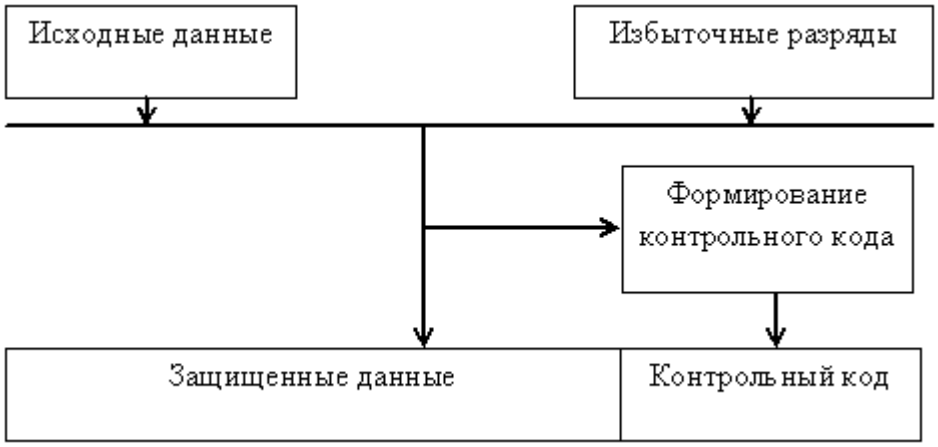

Рис. 6.2. Схема контроля целостности с контрольным кодом

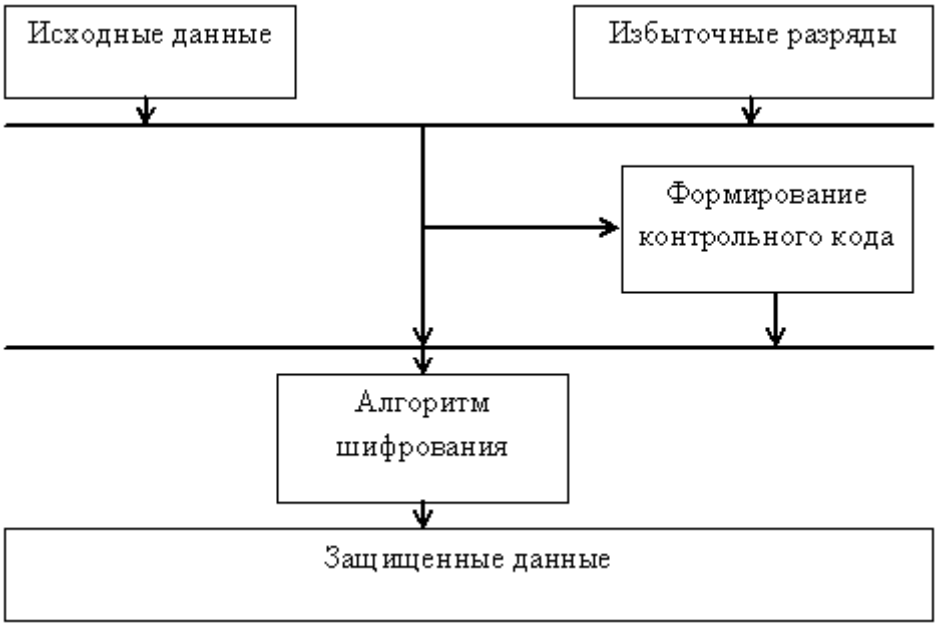

Рис. 6.3. Схема контроля целостности с контрольным кодом и шифрованием

Целостность потока сообщений можно также контролировать, используя шифрование со сцеплением сообщений (см. рис. 6.4).

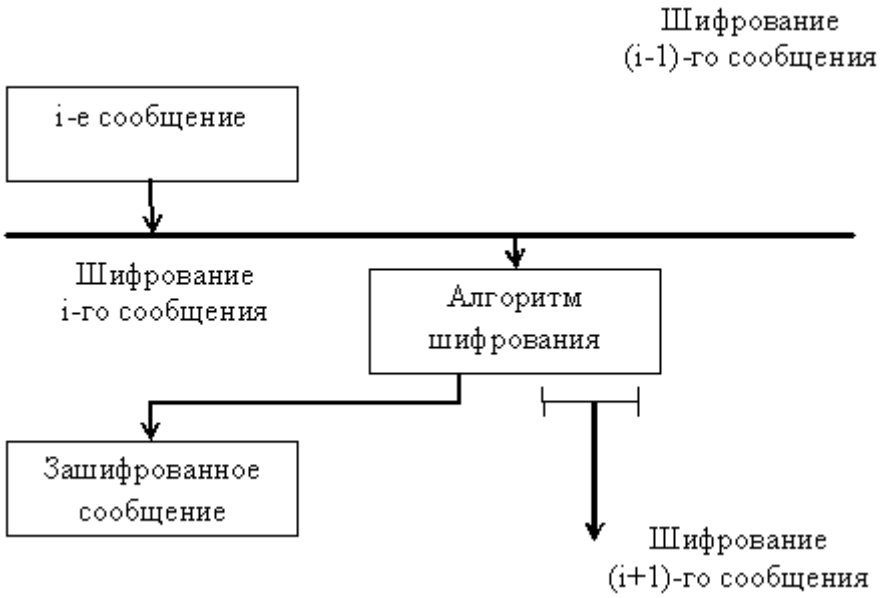

Рис. 6.4. Контроль целостности потока сообщений

Рассмотрим схему на рис. 6.2. Самый естественный способ преобразования информации с внесением избыточности - это добавление к исходным данным контрольного кода s фиксированной разрядности N, вычисляемого как некоторая функция от этих данных:

$$x' = (x, s) = (x, f(x)), \quad |s| = N.$$

В этой ситуации выделение исходных данных из преобразованного массива x' суть простое отбрасывание контрольного кода s. Проверка же целостности заключается в вычислении для содержательной части x' полученного массива данных контрольного кода $s' = f(x')$ и сравнении его с переданным значением s. Если они совпадают, сообщение считается подлинным, в противном случае - ложным:

$$F(x') = \begin{cases} 1, & s = f(x'), \\ 0, & s \neq f(x'). \end{cases}$$

Функция $f(\cdot)$ формирования контрольного кода должна удовлетворять следующим требованиям:

- она должна быть вычислительно необратимой, т.е. подобрать массив данных под заданный контрольный код можно только путем полного перебора по пространству возможных значений x;
- у злоумышленника должна отсутствовать возможность сформировать ложный массив данных (или ложное сообщение) x' и снабдить его корректно вычисленным контрольным кодом $s' = f(x')$.

Второе свойство можно обеспечить двумя способами: либо сделать функцию $f(\cdot)$ зависимой от некоторого секретного параметра (ключа), либо пересылать контрольный код отдельно от защищаемых данных.

Простейшим примером кода s является контрольная сумма блоков массива данных. Например, если $X = X_1X_2X_3...X_t$, то

$$s = f(X) = \sum_{i=1}^{t} X_i (\mathrm{mod}\, 2^n).$$

где n - разрядность.

Однако такое преобразование непригодно для имитозащиты, так как контрольная сумма не зависит от взаимного расположения блоков, а самое главное, соответствующее преобразование не является криптографическим. Процедура подбора данных под заданную контрольную комбинацию чрезвычайно проста. Пусть некий массив данных $X = X_1X_2X_3...X_t$ имеет контрольную сумму s. Тогда для внесения необнаруживаемых искажений противнику достаточно дополнить произвольный ложный массив

$$X' = X'_1X'_2X'_3...X'_t$$

еще одним блоком

$$X'_{t'+1} = s - \sum_{i=1}^{t'} X'_i (\mathrm{mod}\, 2^n).$$

Можно выделить два основных криптографических подхода к решению задачи защиты информации от несанкционированных изменений данных:

- формирование с помощью функции шифрования E_k блочного шифра **кода аутентификации сообщений** MAC (Message Authentication Code);
- формирование с помощью необратимой функции сжатия (хеш-функции) информации **кода обнаружения манипуляций с данными** MDC (Manipulation Detection Code).

В табл. 6.1 приведена сравнительная характеристика указанных двух подходов [9]. Главное различие между кодами MAC и MDC заключается в том, что в первом случае для формирования контрольного кода требуется секретная информация, а во втором - нет.

Таблица 6.1

Сравнительная характеристика MAC и MDC

Параметр	**MAC**	**MDC**
Используемое преобразование	Функция шифрования блочного шифра	Хеш-функция
Секретная информация	Секретный ключ	Нет
Возможность для противника вычислить контрольный код	Отсутствует	Присутствует

Продолжение табл. 6.1

Хранение и передача контрольного кода	Вместе с защищаемыми данными	Отдельно от защищаемых данных
Дополнительные условия	Требует предварительного распределения ключей	Необходим аутентичный канал для передачи контрольного кода
Области использования	Защита при передаче данных	Защита при разовой передаче данных, контроль целостности хранимой информации

6.2. Код аутентификации сообщений

Формирование кода MAC с использованием функции шифрования блочного шифра официально закреплено во многих государственных стандартах шифрования. Имитовставка ГОСТ 28147-89 является классическим примером кода MAC. Код аутентификации сообщений может формироваться в режимах CBC или CFB, обеспечивающих зависимость последнего блока криптограммы от всех блоков открытого текста. В случае использования преобразования E_k для выработки контрольного кода требования к нему несколько отличаются от требований при его использовании для шифрования: во-первых, не требуется свойство обратимости; во-вторых, его криптостойкость может быть снижена (например, за счет уменьшения числа раундов шифрования как в ГОСТ 28147-89). Действительно, в случае выработки кода MAC преобразование всегда выполняется в одну сторону, при этом в распоряжении злоумышленника есть только зависящий от всех блоков открытого текста контрольный код, в то время как при шифровании у него имеется набор блоков криптограммы, полученных с использованием одного секретного ключа.

Существует вариант построения кода MAC на основе использования секретного ключа и функции хеширования, при котором хешированию подвергается результат конкатенации секретного ключа и исходного сообщения, поэтому, как и в классическом случае у злоумышленника, не знающего ключа, отсутствует возможность вычислить контрольный код. Для повышения безопасности подобного алгоритма получения MAC создана схема вложенного MAC, в которой хеширование выполняется дважды. В американском стандарте FIPS 198 вложенный MAC назван HMAC (hashed MAC). В американском стандарте FIPS 113 определена схема формирования кода аутентификации сообщений, названная CMAC.

6.3. Код обнаружения манипуляций с данными

Код MDC есть результат действия хеш-функции. Иначе говоря, MDC - это хеш-образ сообщения X, к которому применили хеш-функцию, т.е. $s = h(X)$. Схема формирования кода MDC, обладающего гарантированной стойкостью, равной стойкости используемого шифра, может быть следующей:

1) массив данных X разбивается на блоки фиксированного размера, равного размеру ключа $|k|$, используемого блочного шифра, т.е.

$$X = X_1 X_2 X_3 ... X_t,$$
$$|X_1| = |X_2| = |X_3| = ... = |X_{m-1}| = |k|,\ 0 < |X_t| < |k|;$$

2) если последний блок X_t неполный, он дополняется каким-либо образом до нужного размера $|k|$;

3) вычисляется хеш-образ

$$s = h(X) = E_{X_t}(...E_{X_3}(E_{X_2}(E_{X_1}(s_0)))),$$

где s_0 - синхропосылка.

Задача подбора массива данных $X' = X'_1 X'_2 X'_3 ... X'_t$ под заданный контрольный код s эквивалентна системе уравнений, которую необходимо решить для определения ключа для заданных блоков открытого и закрытого (в режиме простой замены) сообщений. Однако в рассматриваемой ситуации нет необходимости решать всю систему

$$E_{X'_1}(s_0) = s_1, E_{X'_2}(s_1) = s_0, ..., E_{X'_t}(s_{X'-1}) = s,$$

достаточно решить одно уравнение

$$E_{X'_i}(s_{i-1}) = s_i,$$

относительно X'_i, остальные блоки массива X могут быть произвольными. Но и эта задача в случае использования надежной функции E_k вычислительно неразрешима.

К сожалению, приведенная схема формирования MDC не учитывает наличия так называемых **побочных ключей** шифра. Если для $k' \neq k$ справедливо

$$E_{k'}(X_i) = E_k(X_i),$$

где X_i - некоторый блок открытого текста, то такой код k' и является побочным ключом, т.е. ключом, дающим при шифровании блока X_i точно такой же результат, что и истинный ключ k.

Обнаружение противником побочного ключа при дешифровании сообщения не является особым успехом, так как с вероятностью близкой к 1 на этом найденном побочном ключе он не сможет правильно расшифровать другие блоки закрытого текста, учитывая, что для различных блоков побочные ключи в общем случае также различны. В случае выработки кода MDC

ситуация прямо противоположна: обнаружение побочного ключа означает, что противник нашел такой ложный блок данных, использование которого не изменяет контрольного кода [9].

Для уменьшения вероятности навязывания ложных данных в результате нахождения побочных ключей, при преобразовании применяются не сами блоки исходного сообщения, а результат их расширения по некоторому алгоритму. Под **расширением** понимается процедура получения блока данных большего размера из блока данных меньшего размера.

6.4. CRC-код

Идеальным средством защиты информации от случайных искажений являются CRC-коды (cyclic redundancy code). Достоинствами CRC-кодов являются [9]:

- высокая достоверность обнаружения искажений, доля обнаруживаемых искажений не зависит от длины массива данных и составляет $1-2^N$, где N - разрядность контрольного кода;
- зависимость контрольного кода не только от всех бит анализируемой информационной последовательности, но и от их взаимного расположения;
- высокое быстродействие, связанное с получением контрольного кода в реальном масштабе времени;
- простота аппаратной реализации и удобство интегрального исполнения;
- простота программной реализации.

К сожалению, простое условие пропуска искажений делает CRC-коды принципиально непригодными для защиты от умышленных искажений информации.

Сущность процесса контроля целостности с использованием CRC-кодов заключается в следующем. Генератор CRC-кода инициализируется фиксированным начальным значением. Чаще всего в качестве начального заполнения используется либо код «все 0», либо код «все 1». Учитывая, что от начального состояния генератора достоверность метода не зависит, все дальнейшие рассуждения выполняются в предположении, что исходное состояние устройства - нулевое. Анализируемая двоичная последовательность преобразуется в короткий (обычно шестнадцати- или тридцатидвухразрядный) двоичный код - CRC-код. Значение полученного CRC-кода сравнивается с эталонным значением, полученным заранее для последовательности без искажений. По результатам сравнения делается вывод о наличии или отсутствии искажений в анализируемой последовательности.

CRC-код часто называют сигнатурой (signature*)*. Однако этот термин следует признать неудачным, создающим ненужную путаницу, так как термин *signature* может означать также и электронную подпись.

Процесс получения CRC-кода можно рассматривать как процедуру наложения псевдослучайной последовательности, формируемой регистром

сдвига с линейной обратной связью (LFSR), на входную анализируемую последовательность.

Рассмотрим схему генератора CRC-кода, показанную на рис. 6.5, входной анализируемой двоичной последовательности

$$A = a_0 a_1 a_2 ... a_i ... a_{m-1},\ a_i \in \{0,1\},\ i = \overline{0, (n-1)}.$$

Можно поставить в соответствие многочлен $A(x)$ степени $(n-1)$. Тогда процесс получения CRC-кода эквивалентен делению многочлена $A(x)$ входной последовательности на характеристический многочлен $\varphi(x) = \Phi(x^{-1})x^N$ степени N генератора CRC-кода. В качестве характеристического многочлена чаще всего выбирается примитивный многочлен. Для случая, представленного на рис. 6.5, $\varphi(x) = x^4 + x + 1$.

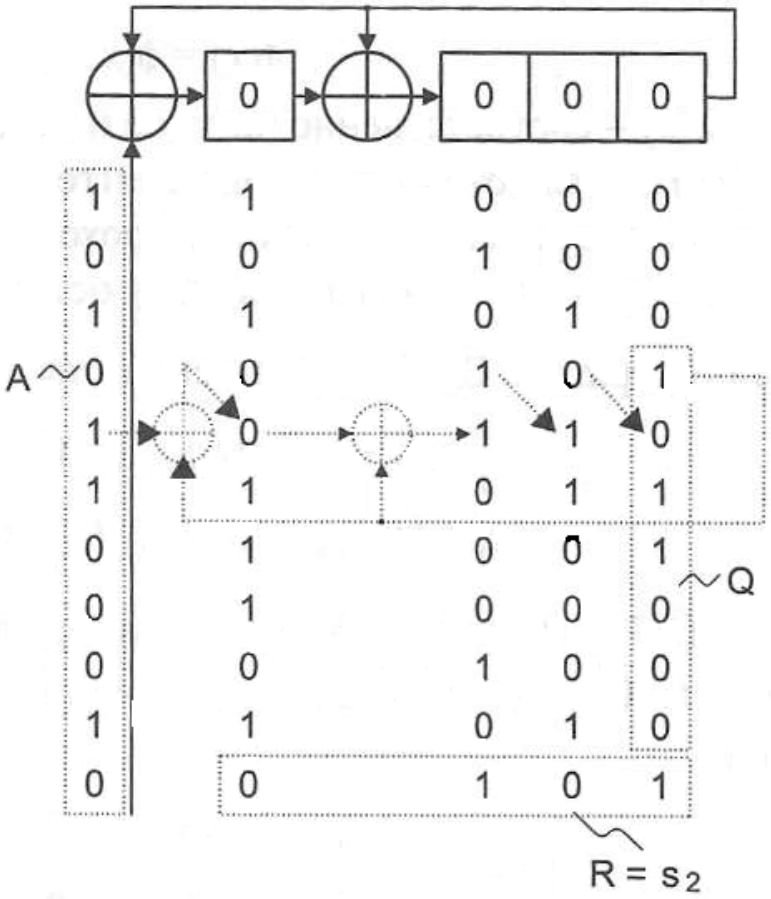

Рис. 6.5. CRC-генератор

Если

$$A(x) = \varphi(x)Q(x) + R(x),$$

где $Q(x)$ и $R(x)$ - соответственно частное и остаток от деления, то коэффициенты многочлена $Q(x)$ появляются на выходе триггера q_{N-1}, а коэффициенты многочлена $R(x)$ остаются в регистре генератора после

прохождения всей последовательности A. Иначе говоря, CRC-код s в точности равен коду остатка R.

Работа генератора CRC-кода описывается системой линейных уравнений

$$\begin{cases} q_0(t+1) = \alpha_N q_{N-1}(t) \oplus a_t, \\ q_i(t+1) = \alpha_{N-i} q_{N-1}(t) \oplus q_{i-1}(t), j = \overline{1, N-1}, \end{cases}$$

где $q_i(t)$ и $q_i(t+1)$ - состояние j-го разряда (триггера) соответственно в моменты времени t и $t+1$, $t = \overline{0, m}$, $j = \overline{0, N-1}$. В матричной форме уравнение работы CRC-генератора имеет вид

$$\begin{bmatrix} q_0(t+1) \\ q_1(t+1) \\ .... \\ q_{N-1}(t+1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & ... & \alpha_N \\ 1 & 0 & ... & \alpha_{N-1} \\ ... & ... & ... & ... \\ 0 & ... & 1 & \alpha_1 \end{bmatrix} \times \begin{bmatrix} q_0(t) \\ q_1(t) \\ ... \\ q_{N-1}(t) \end{bmatrix} + \begin{bmatrix} a_t \\ 0 \\ ... \\ 0 \end{bmatrix},$$

или

$$\mathbf{q}(t+1) = T_2 \mathbf{q}(t) \oplus \mathbf{a}(t),$$

где

$$\mathbf{q}(t) = \begin{bmatrix} q_0(t) \\ q_1(t) \\ ... \\ q_{N-1}(t) \end{bmatrix}, \ \mathbf{q}(t+1) = \begin{bmatrix} q_0(t+1) \\ q_1(t+1) \\ ... \\ q_{N-1}(t+1) \end{bmatrix}, \ \mathbf{a}(t) = \begin{bmatrix} a \\ 0 \\ ... \\ 0 \end{bmatrix}, \ s = \mathbf{q}(n) = \begin{bmatrix} q_0(n) \\ q_1(n) \\ ... \\ q_{N-1}(n) \end{bmatrix}.$$

Таким образом, для CRC-генератора как линейного устройства справедлив принцип суперпозиции, который гласит: реакция линейного устройства на сумму двух входных воздействий равна сумме реакций на каждое воздействие в отдельности.

Пусть $A = a_0 a_1 a_2 ... a_i ... a_{m-1}$ - анализируемая двоичная последовательность, $B = b_0 b_1 b_2 ... b_i ... b_{m-1}$ - правильная последовательность (без искажений), e - последовательность, полученная в результате сложения по модулю 2 соответствующих элементов последовательностей A и B, т.е. для любого элемента последовательности e справедливо

$$e_i = a_i \oplus b_i, i = \overline{0, (n-1)}.$$

Единичные биты последовательности e соответствуют искаженным битам последовательности B, поэтому последовательности e логично назвать **последовательностью** или **вектором ошибок**. При отсутствии искажений

$$\forall i = \overline{0, (n-1)}, e_i = 0.$$

Пусть $A(x)$, $B(x)$ и $e(x)$ - многочлены; S_A, S_B, S_e - CRC-коды соответственно последовательностей A, B и e. Искажения в

последовательности A будут пропущены, если $S_A = S_B$. Имеем $A = B \oplus e$, откуда, применяя принцип суперпозиции, получаем равенство

$$s_A = s_B \oplus s_e.$$

Таким образом, необходимым и достаточным условием пропуска искажений является равенство $S_e = 0$, которое имеет место, когда многочлен $e(x)$ нацело делится на многочлен $\varphi(x)$.

Пусть N-разрядность CRC-кода и $e \neq 0$, т.е. в анализируемой последовательности длиной m есть искажения. Рассмотрим достоверность метода, т.е. условия, при которых $S_e = 0$. При $m \leq N$ контрольный код не может быть нулевым, так как первая единица, попавшая в регистр генератора, не успевает выйти из него до конца формирования CRC-кода и не может быть уничтожена из-за сложения с битом обратной связи. Таким образом, при длине входной последовательности, меньшей или равной разрядности CRC-кода, для $\forall e \neq 0$ справедливо $s_e \neq 0$, т.е. число не обнаруживаемых искажений равно $N_e = 0$. При $m = N + 1$, когда степень многочлена меньше или равна N, существует только один многочлен $e(x)$, нацело делящийся на многочлен $\varphi(x)$, это $e(x) = \varphi(x)$, а значит, в этом случае $N_e = 1$. При $m = N + 2$ существуют уже три многочлена $e(x)$, степени меньшей или равной $N + 1$, нацело делящиеся на многочлен $\varphi(x)$, это $e(x) = \varphi(x)$, $e(x) = \varphi(x) \times x$, $e(x) = \varphi(x) \times (x + 1)$, а значит $N_e = 1$. В общем случае при $N > 1$ справедливо

$$N_e = 2^{m-N} - 1.$$

Учитывая, что общее число искажений в последовательности длиной m равно $2^m - 1$, для доли P_d обнаруживаемых искажений получаем соотношение

$$P_d = 1 - \frac{2^{m-N} - 1}{2^m - 1}.$$

На практике $m >> N$, а значит $2^m >> 1$, $2^{m-N} >> 1$,

$$P_d = 1 - \frac{2^{m-N}}{2^m} = 1 - 2^{-N}.$$

Таким образом, доля обнаруживаемых искажений не зависит от длины анализируемой последовательности, а определяется лишь разрядностью контрольного кода.

Контрольные вопросы

1. Дайте определение аутентификации. В чем разница между аутентификацией и идентификацией?
2. В чем разница между аутентификацией объекта и субъекта?
3. Назовите способы обеспечения целостности информации.

Литература

1. Бабаш А.В., Шанкин Г.П. Криптография /под ред. В.П. Шерстюка, Э.А. Применко. – М.: СОЛОН-ПРЕСС, 2007.
2. Баричев С.Г., Гончаров В.В., Серов Р.Е. Основы современной криптографии: учебный курс. – М.: Горячая линия-Телеком, 2002.
3. Болелов Э.А. Криптографические методы защиты информации. – М.: МГТУ ГА, 2011.- Ч. 1(Симметричные криптосистемы).
4. Нечаев В.И. Элементы криптографии (Основы теории защиты информации): учебное пособие /под ред. В.А. Садовничего. – М.: Высшая школа, 1999.
5. Плотников А.Д. Дискретная математика: учебное пособие. – Мн: Новое знание, 2008.
6. Рябко Б.Я., Фионов А.Н. Основы современной криптографии и стеганографии: учебное пособие – М.: Горячая линия-Телеком, 2010.
7. Харин Ю.С. и др. Математические и компьютерные основы криптологии: учебное пособие. – Мн.: Новое знание, 2003.
8. Ян Сонг Й. Криптоанализ RSA. – М.-Ижевск: НИЦ «Регулярная и хаотическая динамика», Ижевский институт компьютерных исследований, 2011.
9. Иванов М.А. и др. Стохастические методы и средства защиты информации в компьютерных системах и сетях /под ред. И.Ю. Журавлева – М.: КУДИЦ-ПРЕСС, 2009.
10. Молдовян Н.А. Теоретический минимум и алгоритмы цифровой подписи. – СПб.: БХВ-Петербург, 2010.
11. ГОСТ Р 34.10-94 Процессы формирования и проверки электронной цифровой подписи. - М.: Изд-во стандартов, 2001.
12. ГОСТ Р 34.10 2012 Процессы формирования и проверки электронной цифровой подписи. – М.: Изд-во стандартов, 2012.