Лекция 7.

2.8. Построение больших простых чисел.

Алгоритмы построения больших простых чисел носят рекурсивный характер. Чтобы построить большое простое число, нужно построить возрастающую последовательность простых чисел. Число нужного размера появляется в конце этой последовательности, члены же её принадлежат последовательности уменьшающихся промежутков, с построения которых начинается алгоритм. Эта конструкция вместе со всеми пояснениями обсуждается в настоящем параграфе.

Сначала мы рассмотрим следующую задачу: Для каждого заданного натурального $N \geqslant 11$ **построить простое число Q, лежащее на промежутке** $2^{N-1} \leqslant Q < 2^N$.

1. Для каждого числа $N \leqslant 18$ требуемое простое число можно найти непосредственно. Например, на промежутке от 2^{17} до 2^{18} их имеется 10749 штук, и каждое составное число этого промежутка делится на какое нибудь простое $p < 2^9 = 512$. Для нахождения простого числа в нужном маленьком промежутке (правый конец не превосходит 2^{18}) можно воспользоваться алгоритмом пробных делений, см. подраздел 2.10.1.

Далее будет найдена последовательность целых чисел a_k, b_k, q_k таких что

$$q_k \text{ - простые}, \quad 2^{a_k} \leqslant q_k < 2^{b_k}, \quad a_k = b_k - 1, \quad 1 \leqslant k \leqslant m, \quad b_m = N.$$

Тогда $Q = q_m$ - искомое простое число.

2. Пусть N - натуральное число, превосходящее 18. Построим последовательность целых чисел, начинающуюся с N, в которой за числом ℓ следует число $\lceil \frac{\ell}{2} \rceil + 1$. Легко видеть, что эта последовательность убывает при $\ell \geqslant 4$. Более того, в ней обязательно встретится целое число из промежутка $11 \leqslant \ell \leqslant 18$. Кроме того, не трудно проверить, что при любом $\ell \geqslant 19$ выполняется неравенство $\lceil \frac{\ell}{2} \rceil + 1 \geqslant 11$.

Например, для $N = 1024$ эта последовательность имеет вид

$$N = 1024, \quad 513, \quad 258, \quad 130, \quad 66, \quad 34, \quad 18.$$

Итак,

с какого бы числа N ни начиналась указанная последовательность, она обязательно содержит единственное число из промежутка $11 \leqslant$

$x \leqslant 18$. *Начиная с этого числа, перенумеруем в обратном порядке все члены последовательности буквами* $b_1, b_2, \ldots, b_m$. *Тогда* $11 \leqslant b_1 \leqslant 18$ *и* $b_m = N$. *Для каждого индекса* k, $1 \leqslant k \leqslant m$, *положим* $a_k = b_k - 1$.

3. Простое число q_1 выбирается так, как это указано в пункте 1, ведь $11 \leqslant b_1 \leqslant 18$ и $2^{a_1} < q_1 < 2^{b_1}$. Например, в случае $b_1 = 11$ на роль q_1 может быть взято любое из 137 простых чисел, лежащих на отрезке от 1024 до 2048, скажем 1933.

Предположим теперь, что $k \geqslant 2$ и простое число q_{k-1} уже построено. Для краткости обозначим

$$\ell = b_k, \quad q = q_{k-1}. \quad \text{Тогда} \quad a_k = \ell - 1, \quad b_{k-1} = \lceil \frac{\ell}{2} \rceil + 1, \quad a_{k-1} = \lceil \frac{\ell}{2} \rceil.$$

Число q_k будет строиться с помощью Следствия (2) из теоремы (7), см. также алгоритм (5). При этом $F = q$, $R = 2t$, где t - некоторое натуральное, оно будет выбираться с помощью испытаний, и

$$c = 2tq + 1 = FR + 1. \tag{2.25}$$

Для того, чтобы число c попало в нужный интервал, т.е. $2^{\ell-1} < c < 2^{\ell}$, на параметр t накладываются условия

$$\frac{2^{\ell-1}}{2q} \leqslant t \leqslant \frac{2^{\ell} - 1}{2q}. \tag{2.26}$$

Действительно, в этих ограничениях имеем

$$c = 2tq + 1 \geqslant 2^{\ell-1} + 1 > 2^{\ell-1} = 2^{a_k}$$

и

$$c = 2tq + 1 < 2^{\ell} = 2^{b_k}.$$

Строгое неравенство выполняется в силу того, что c нечётно, а 2^{ℓ} чётное число. Не трудно проверить, что для любых положительных чисел $u < v$ на интервале $u < t < v$ лежит не менее $v - u - 1$ целых чисел t. Согласно неравенству (2.26) можно утверждать, что количество целых t, для которых число c принадлежит интервалу $2^{\ell-1} < t < 2^{\ell}$ не меньше, чем $\frac{2^{\ell-}}{2q} - 1$. Например, при $b_1 = 11, \ell = b_2 = 19$ и $q = 1933$ на промежутке $2^{18} < c < 2^{19}$ имеется 10 простых чисел вида $2tq + 1$. Самые маленькие из них соответствуют числам $t = 65, 72, 75, 77, 81$. При $t = 77$ имеем $c = 297683$.

4. Как правило, строящиеся большие простые числа должны иметь случайный характер. Поэтому на интервале от $2^{\ell-1}$ до 2^{ℓ} выбирается

случайное число x, а затем от него начанается отсчёт чисел t, соответствующие им числа c проверяются на простоту. Как только попадётся простое c, оно выбирается в качестве следующего простого q_k. Если при очередном t соответствующее число c выйдет за верхний предел заданного промежутка, следующее значение t принимается равным наименьшему возможному значению и вычисления продолжаются.

Выберем псевдослучайное целое число x на промежутке $2^{\ell-1} \leqslant x < 2^{\ell}$ и положим $t = \lceil \frac{x}{2q} \rceil$. Определим также число c равенством (2.25).

Из неравенства $x \geqslant 2^{\ell-1}$ следует (2.25).

5. *Если t оказалось настолько большим, что*

$$2tq + 1 \geqslant 2^{\ell},$$

то полагаем $t = \lceil \frac{2^{\ell-1}}{2q} \rceil$. Если с помощью следствия 2 при $F = q$ и $R = 2t$ выбирая псевдослучайные числа b на промежутке от 2 до $c-2$, удаётся доказать, что число c простое, то выбираем $q_k = c$. Для того, чтобы с помощью следствия 2 можно было доказать простоту числа c, параметры R и F должны быть связаны неравенством $R \leqslant 4F + 2$, которое может быть записано в виде $t \leqslant 2q + 1$. Используя границы для t и q, находим

$$q^2 > 2^{2\lceil \frac{\ell}{2} \rceil} \geqslant 2^{\ell} > 2qt \quad и \quad t < \frac{1}{2}q < 2q + 1,$$

так что следствие 2 применимо.

Если $k = m$, алгоритм завершает свою работу. В противном случае увеличиваем k на единицу и переходим в пункт 3 к следующему промежутку $[a_k, b_k)$.

6. *Если же c оказывается составным, то параметр t увеличивается на 1 и алгоритм переходит в пункт 5.*

Следующий далее алгоритм, по заданным числам L и N строит пары простых чисел P, Q с условиями

$$Q|(P-1), \qquad 2^{L-1} \leqslant P < 2^{L}, \qquad 2^{N-1} \leqslant Q < 2^{N}.$$

Мы ограничимся здесь только формальным описанием его основных моментов.

Алгоритм 7. Даны *натуральные числа L и $N, L > N$.*

Построить *"случайные"простые числа P, Q битовой длины L и N с условием $Q|(P-1)$.*

1. С помощью описанного выше алгоритма построить "случайное" простое число Q с длиной записи N битов.

2. Построить последовательность простых чисел p_k, связанных соотношениями

$$p_k = 2tQp_{k-1} + 1, \quad 1, \ldots, m, \qquad 2^{L-1} \leqslant p_m < 2^L.$$

Для проверки на простоту чисел $c = 2tQp_{k-1} + 1$ использовать следствие 2 с параметрами $F = p_{k-1}$, $R = 2tQ$.

3. Положить $P = p_m$.

2.9. Первообразные корни и дискретное логарифмирование.

Пусть p – простое нечётное число. Множество классов вычетов целых чисел по модулю p составляет поле $\mathbb{F}_p$, а ненулевые элементы этого поля образуют циклическую мультипликативную группу порядка $p-1$. Согласно малой теореме Ферма выполняется сравнение $a^{p-1} \equiv 1 \pmod{p}$. Наименьшее натуральное число d с условием $a^d \equiv 1 \pmod{m}$ называется *показателем* числа a по модулю p. Например, показатель числа 1 по модулю p равен 1, а показатель -1 по нечётному модулю p равен 2. *Целое число a, не делящееся на p, называется первообразным корнем по модулю p, если его показатель равен $p-1$.* Например, показатель числа 5 по модулю 23 равен 22. Мы проверим это позже. Число 5 есть первообразный корень по модулю 23. Его класс вычетов по модулю 23 порождает мультипликативную группу всех ненулевых классов вычетов по модулю 23.

В 1801 году К.Ф. Гаусс опубликовал два доказательства следующей теоремы.

Теорема 8. *Для каждого простого нечетного числа p существуют первообразные корни по модулю p. Количество не сравнимых друг с другом по модулю p первообразных корней равно $\varphi(p-1)$, где $\varphi(n)$ – функция Эйлера.*

Для практического нахождения первообразных корней удобно пользоваться следующим утверждением.

Теорема 9. *Целое число g, не делящееся на простое нечетное p, будет первообразным корнем по модулю p в том и только том случае, если для любого простого числа q, делящего $p-1$, выполняется*

$$g^{\frac{p-1}{q}} \not\equiv 1 \pmod{p}.$$

Доказательство. Обозначим буквой d показатель числа g по модулю p. Разделив $p-1$ на d с остатком, получим

$$p-1 = d\cdot u + v, \qquad 0 \leqslant v < d.$$

Неравенство v>0 в силу малой теоремы Ферма и сравнений

$$1 \equiv g^{p-1} = (g^d)^u \cdot g^v \equiv g^v \pmod{p}$$

противоречит определению d. Значит $v=0$ и $d|(p-1)$. Если $d<p-1$, то найдется такое простое число q, что $d|\frac{p-1}{q}$. Но тогда $g^{\frac{p-1}{q}} = (g^d)^{\frac{p-1}{qd}} \equiv 1 \pmod{p}$, вопреки условию. Значит, $d=p-1$, и это завершает доказательство теоремы. □

Если известны все простые делители числа $p-1$, то проверка условий теоремы 9 выполняется достаточно быстро с помощью алгоритма, изложенного в параграфе 2.3.

Справедливы сравнения по модулю 23

$$5^{11} = 5\cdot 25^5 \equiv 5\cdot 2^5 \equiv -1 \pmod{23}$$

и $5^2 \equiv 2 \pmod{23}$. Применяя теорему 9 к простому числу $p=23$ и $c=5$, заключаем, что 5 есть первообразный корень по модулю 23.

Множество первообразных корней при заданном p достаточно велико, поэтому выбирая числа g случайным образом на промежутке $0<g<p$, можно с большой вероятностью попасть на первообразный корень и доказать это с помощью теоремы 9.

Если g – первообразный корень по модулю p, то числа

$$1, g, g^2, \ldots, g^{p-2} \tag{2.27}$$

имеют разные наименьшие неотрицательные остатки. Действительно, если бы нашлись целые числа $0 \leqslant u < v \leqslant p-2$, для которых g^u и g^v имеют одинаковые остатки, то $g^u \equiv g^v \pmod{p}$. В силу взаимной простоты чисел g и p мы получили бы

$$1 \equiv g^{v-u} \pmod{p}.$$

Но это невозможно, так как $0<v-u<p-1$, а показатель g равен $p-1$.

Множество (2.27) состоит из $p-1$ чисел. Поэтому для каждого целого числа a, не делящегося на p, найдется единственное целое k, удовлетворяющее условиям

$$a \equiv g^k \pmod{p}, \qquad 0 \leqslant k < p-1. \tag{2.28}$$

Число k называют *индексом* a по модулю p при основании g. Это определение и формулируемые ниже свойства индексов напоминают свойства обычных логарифмов действительных чисел. Поэтому иногда в современной литературе индексы называют дискретными логарифмами, а процесс их нахождения – дискретным логарифмированием. Используются также обозначения $\operatorname{ind}_g a$, $Log_g a$. Указание на первообразный корень иногда опускается. Итак, имеем

$$a \equiv g^{\operatorname{ind}_g a} \pmod{p}, \qquad 0 \leqslant \operatorname{ind}_g a < p-1. \tag{2.29}$$

Свойства индексов описывает следующие простые утверждения.

1. Если целые числа a, b не делятся на p, то

$$\operatorname{ind}_g ab \equiv \operatorname{ind}_g a + \operatorname{ind}_g b \pmod{p-1}.$$

2. Если a, b – первообразные корни по модулю p, то для любого числа c, не делящегося на p, выполняется сравнение

$$\operatorname{ind}_b c \equiv \operatorname{ind}_b a \cdot \operatorname{ind}_a c \pmod{p-1}.$$

Сформулируем теперь задачу дискретного логарифмирования по простому нечетному модулю.

Дано простое число p. Для заданных чисел $a, b \in \mathbb{Z}$, не делящихся на p, требуется решить сравнение

$$b^x \equiv a \pmod{p}. \tag{2.30}$$

Решение этой задачи очень трудоемко в вычислительном отношении. Не случайно в конце практически всех учебников по элементарной теории чисел приводятся таблицы индексов (дискретных логарифмов).

Лучшие из известных алгоритмов дискретного логарифмирования по простому модулю p, использующие вычисления в полях алгебраических чисел, требуют $O(e^{2(\ln p)^{1/3}(\ln\ln p)^{2/3}})$ арифметических операций. Впрочем, эта оценка условна, ибо опирается на ряд недоказанных, но весьма правдоподобных гипотез теории чисел.

Рекордный по величине простого числа p результат в задаче дискретного логарифмирования был установлен в 2016 году. Группе европейских математиков и программистов удалось сосчитать дискретный логарифм по модулю простого числа, записываемого 232 десятичными цифрами (768 битами). Применённый ими алгоритм требует достаточно глубоких познаний в теории алгебраических чисел, и мы его здесь не объясняем.

Излагаемый здесь метод решения сравнения (2.30) требует $O(p^{1/2}\ln p)$ арифметических операций. Он был придуман в 1962г. А.О.Гельфондом и

в русскоязычной литературе назывется "метод согласования". Аналогичный метод был предложен в 1971г. Д.Шенксом. Его английское название может быть переведено как "метод больших и малых шагов".

Лемма 1. *Пусть p — простое нечетное число и пусть $H = [\sqrt{p}] + 1$. Тогда для каждого целого числа d, $1 \leqslant d < p$, найдутся целые числа u, v, удовлетворяющие условиям*

$$1 \leqslant u, v \leqslant H, \qquad Hu - v = d.$$

Доказательство. Положим

$$u = \left[\frac{d}{H}\right] + 1, \qquad v = Hu - d.$$

Тогда

$$\frac{d}{H} < u \leqslant \frac{d}{H} + 1,$$

откуда видим, что, во–первых,

$$0 < u < \frac{p}{\sqrt{p}} + 1 = \sqrt{p} + 1,$$

а во–вторых, $0 < Hu - d \leqslant H$. Остается воспользоваться тем, что числа u и v целые. □

Алгоритм 8 (Алгоритм Гельфонда)**.** Данные: *Простое число $p \geqslant 3$, первообразный корень g по модулю p, число $b \in \mathbb{Z}$, $p \nmid b$.*
Найти: *Решение сравнения $g^x \equiv b$.*
1. Вычислить $H = [\sqrt{p}] + 1$.
2. Положить $c \equiv g^H \bmod p$, $1 \leqslant c < p$.
3. Составить два набора чисел

$$S_1 = \{c^u \bmod p : 1 \leqslant u \leqslant H\}, \quad S_2 = \{bg^v \bmod p : 1 \leqslant v \leqslant H\}.$$

4. Упорядочить по возрастанию оба набора S_1 и S_2. Найти совпавшие элементы этих наборов, то есть такие числа u, v, для которых

$$c^u \equiv bg^v \pmod{p}. \tag{2.31}$$

5. Положить

$$\operatorname{ind}_g b = Hu - v. \tag{2.32}$$

Решение заданного сравнения имеет вид $x \equiv \operatorname{ind}_g b \pmod{p-1}$.

Пересечение множеств S_1 и S_2 не пусто, так как сравнение (2.31) равносильно представлению (2.32), а последнее, согласно лемме 1, существует всегда.

Вычисления в пунктах 1 и 2 требуют $O(\ln p)$ арифметических операций. Вычисления в пункте 3 требуют $O(H \ln p) = O(\sqrt{p} \ln p)$ арифметических операций. Для упорядочения каждого из множеств S_1, S_2 нужно $O(H \ln H) = O(\sqrt{p} \ln p)$ арифметических операций. Для нахождения одинаковых элементов в упорядоченных множествах S_1, S_2 нужно $O(H) = O(\sqrt{p})$ арифметических операций. Общее же количество операций в алгоритме Гельфонда равно $O(\sqrt{p} \ln p)$.

Пример. Решить сравнение

$$2^x \equiv 23 \pmod{37}. \tag{2.33}$$

Решение. Так как $36 = 2^2 \cdot 3^2$ и

$$2^{18} \equiv -1 \not\equiv 1 \pmod{37}, \qquad 2^{12} \equiv 26 \not\equiv 1 \pmod{37},$$

то 2 - первообразный корень по модулю 37 и, значит данное сравнение разрешимо. В соответствии с алгоритмом вычисляем $H = [\sqrt{37}]+1 = 7$ и $c = 2^7 \equiv 17 \pmod{37}$. Множества S_1 и S_2 состоят из чисел $c^n \pmod{37}$, и $b \cdot a^n \pmod{37}$ при $n = 1, 2, \ldots, 7$, т.е.

$$S_1 = \{17, 30, 29, 12, 19, 27, 15\}, \qquad S_2 = \{9, 18, 36, 35, 33, 29, 21\}.$$

Эти два множества имеют общий элемент 29, который стоит на $u = 3$ месте в первом множестве и на $v = 6$ месте во втором множестве. Таким образом, $\mathrm{ind}_2 23 = 7 \cdot 3 - 6 = 15$, а решение сравнения (2.33) имеет вид $x \equiv 15 \pmod{37}$.

Если в сравнении (2.30) число b не является первообразным корнем по модулю p, то это сравнение можно переписать в равносильном виде

$$x \cdot \mathrm{ind}_g b \equiv \mathrm{ind}_g a \pmod{p-1}. \tag{2.34}$$

Индексы чисел a, b можно вычислить с помощью алгоритма Гельфонда. После этого можно решить линейное сравнение (2.34), см. параграф **??**. Все найденные числа x составят решения (2.30). Если сравнение (2.34) не имеет решений, то не будет их иметь и сравнение (2.30).

Алгоритм Гельфонда, конечно, быстрее полного перебора, но он работает недопустимо медленно, чтобы решать задачу дискретного логарифмирования для простых чисел размером в 250 десятичных цифр.

Конец седьмой лекции.

Лекция 8.

2.10. Задача разложения целых чисел на множители.

В этой главе будут рассматриваться простейшие методы разложения целых чисел на простые сомножители, т.е. методы поиска для заданного целого $N > 1$ простых чисел $p_1, \ldots, p_r$ таких, что

$$N = p_1^{k_1} \cdots p_r^{k_r}, \qquad k_j \geqslant 1,\ k_j \in \mathbb{Z}.$$

При этом будет предполагаться, что разлагаемое число N составное, в чем можно убедиться с помощью тестов из параграфа 2.5.

Достаточно уметь решать более простую задачу о разложении целого числа на два меньших множителя, т.е. задачу о решении в целых числах $a > 1, b > 1$ уравнения $N = ab$. Действительно, в этом случае выполняются неравенства $a < N, b < N$, можно разложить на два меньших множителя каждое из чисел a, b, и продолжать далее эту процедуру, пока такое разложение будет возможным, т.е. до тех пор, пока все сомножители не станут простыми.

Существующие алгоритмы разложения чисел на множители могут быть распределены на группы в зависимости от количества арифметических операций, которые алгоритм требует для своей работы.

1) *Экспоненциальные* алгоритмы используют $O(N^c)$ арифметических операций. Здесь c — положительная постоянная.

2) *Субэкспоненциальные* алгоритмы требуют $O(e^{c(\ln N)^\alpha (\ln \ln N)^\beta})$ арифметических операций. Здесь α, β, c — положительные постоянные, $\alpha + \beta = 1$. Заметим, что при $\alpha = 1$, т.е. $\beta = 0$, оценка совпадает с оценкой сложности экспоненциальных алгоритмов.

Для наиболее быстрого из субэкспоненциальных алгоритмов, так называемого *метода решета числового поля*, или метода просеивания в числовых полях имеем $\alpha = \frac{1}{3}, \beta = \frac{2}{3}$.

Алгоритмы полиномиальной сложности, $\alpha = 0, \beta = 1$, для задачи факторизации не известны, и весьма вероятно, что их не существует.

Деятельность по разложению чисел на множители сочетает в себе черты инженерной науки, поскольку во многом опирается на допущения, основанные на опыте и не имеющие теоретических обоснований, а с другой стороны она сродни искусству, так как зачастую продолжительность работы алгоритма и результат зависят от удачного выбора параметров.

2.10.1. Алгоритм пробных делений.

Пусть $d_1 < d_2 < \ldots$ — последовательность целых чисел, содержащая все простые числа. Алгоритм пробных делений состоит в последовательном делении N на числа $d_1, d_2, \ldots$, не превосходящие $\sqrt{N}$. Если N составное число, то оно имеет простой делитель $p \leqslant \sqrt{N}$ и потому будет разложено на множители.

Этот алгоритм часто используется для нахождения всех простых делителей числа N, не превосходящих некоторой заданной границы B.

Последовательность d_i может совпадать с множеством простых чисел. Но в этом случае необходим алгоритм, строящий все простые числа до заданной границы. Иногда бывает проще использовать последовательности, содержащие и составные числа, но менее сложные в реализации.

Пример 1. Каждое простое число $p > 6$ удовлетворяет одному из двух сравнений $p \equiv 1 \pmod 6$ или $p \equiv 5 \pmod 6$. Поэтому последовательность

$$2, 3, 5, 7, 11, 13, 17, 19, 23, 25, \ldots$$

содержащая все числа вида $6n \pm 1, n \in \mathbb{N}$, включает в себя и все простые числа. Но не только их. Например, она содержит 25 и 91, и многие другие составные числа. Но простое правило порождения этой последовательности облегчает поиск делителей N:

$$d_1 = 2,\ d_2 = 3,\ d_3 = 5, \quad d_{2k} = d_{2k-1} + 2, \quad d_{2k+1} = d_{2k} + 4, \quad k \geqslant 2.$$

При таком выборе d_i алгоритм вообще говоря требует $O(N^{1/2})$ арифметических операций и $O(1)$ памяти. Конечно, он очень медленный, но позволяет быстро отсеивать составные числа, имеющие малые простые делители.

Пример 2. В качестве модуля в сравнениях вместо 6 можно взять число $m = 2 \cdot 3 \cdot 5 = 30$. Все простые числа $p > 5$ будут содержаться в восьми прогрессиях с разностью 30, начинающихся с чисел $1, 7, 11, 13, 17, 19, 23, 29$, т.е.

$$30k + 7, 30k + 11, 30k + 13, 30k + 17, 30k + 19, 30k + 23, 30k + 29, 30k + 31.$$

Соответствующая последовательность d_i порождается формулами

$$\begin{aligned}
d_1 = 2, d_2 = 3, d_3 = 5, &\qquad d_{8k+4} = d_{8k+3} + 2,\\
d_{8k+5} = d_{8k+4} + 4, &\qquad d_{8k+6} = d_{8k+5} + 2,\\
d_{8k+7} = d_{8k+6} + 4, &\qquad d_{8k+8} = d_{8k+7} + 2,\\
d_{8k+9} = d_{8k+8} + 4, &\qquad d_{8k+10} = d_{8k+9} + 6,\\
d_{8k+11} = d_{8k+10} + 2, &\qquad d_{8k+12} = d_{8k+11} + 6, \quad k \geqslant 0.
\end{aligned}$$

Соответствующая последовательность $\{d_i\}$ будет содержать меньшую долю составных чисел. Если $m = \prod_{p \leqslant r} p$, то количество арифметических прогрессий, составляющих эту последовательность равно $\varphi(m)$, а доля чисел из $\{d_i\}$ в натуральном ряде есть $\dfrac{\varphi(m)}{m} = \prod_{p \leqslant r} \left(1 - \dfrac{1}{p}\right)$. При $m = 6$ она равна $\dfrac{1}{3}$, а при $m = 30$ имеем $\frac{1}{4}$.

2.10.2. ρ — метод Полларда.

Пусть $f(x)$ — "достаточно случайный"многочлен. Выберем "случайно" $x_1 \in \mathbb{Z}, 1 < x_1 < N$, и рассмотрим последовательность

$$x_{k+1} \equiv f(x_k) \pmod{N}, \qquad 0 \leqslant x_{k+1} < N, \quad k \geqslant 1, \tag{2.35}$$

Так как количество классов вычетов по модулю N не превосходит N, то существуют такие индексы $i, j, 0 \leqslant i, j < N$, что $x_i \equiv x_j \pmod{N}$ и значит, последовательность $x_k \pmod{N}$ зацикливается. Период этой последовательности не всегда начинается с самого начала, она может иметь предпериодическую часть. Символически такую последовательность можно изобразить в виде греческой буквы ρ. Подходящая снизу ножка этой буквы соответствует предпериодической части последовательности, она переходит в замкнутую петлю, соответствующую периодической части. Эта аналогия и дала название алгоритму.

Полларда обнаружил, что для простых p, как правило, длина периода и предпериодическая часть последовательности

$$x_{k+1} \equiv f(x_k) \pmod{p}, \qquad k \geqslant 1,$$

ограничены сверху величиной $c\sqrt{p}$, где c — некоторая константа.

Идея алгоритма состоит в том, чтобы последовательно вычислять наибольшие общие делители $(x_{2i} - x_i, N)$, $i = 1, 2, 3, \ldots$. Если p — простой делитель N, ℓ — длина периода и a — длина предпериодической части последовательности $x_{k+1} \equiv f(x_k) \pmod{p}$, то для номера i, удовлетворяющего условиям

$$i > a, \qquad \ell | i, \tag{2.36}$$

числа x_i и $x_{i+\ell}$ сравнимы друг с другом по модулю p. Мы не знаем чисел ℓ, i, но знаем, что разность $2i - i = i$ делится на ℓ, поэтому $x_{2i} \equiv x_i \pmod{p}$, так что $(x_{2i} - x_i, N) > 1$. Ясно, что существует число i, удовлетворяющее условиям (2.36) и неравенству $i \leqslant a + \ell = O(\sqrt{p})$. Конечно, может случиться, что $N | x_{2i} - x_i$. Тогда нужно выбирать иное начальное значение x_1.

Если p — наименьший простой делитель N, то $p \leqslant N^{1/2}$, так что $i = O(N^{1/4})$.

В следующем ниже алгоритме вычисляются пары $\{x_i, x_{2i}\}, i \geqslant 1$. Для нахождения следующей пары $\{x_{i+1}, x_{2i+2}\}$ нужно вычислить

$$x_{i+1} \equiv f(x_i) (\mathrm{mod}\, N), \quad x_{2i+1} \equiv f(x_{2i}) (\mathrm{mod}\, N),$$
$$x_{2i+2} \equiv f(x_{2i+1}) (\mathrm{mod}\, N),$$

т.е. выполнить $O(1)$ арифметических операций.

Алгоритм 9. Дано: *составное число* N. Найти: *нетривиальный делитель* N .

1. Выбрать случайно $x_1 \in \mathbb{Z}, 1 < x_1 < N$, *и положить*

$$x = f(x_1) \pmod N, \qquad y = f(x) \pmod N.$$

2. Вычислить $d = (y - x, N)$. *Если* $1 < d < N$ *алгоритм останавливается, нетривиальный делитель* d *числа* N *найден.*

3. Если $d = N$, *перейти в п. 1.*

4. Положить

$$x = f(x) \pmod N, \quad z = f(y) \pmod N, \quad y = f(z) \pmod N$$

и перейти в пункт 2 алгоритма.

Если все удачно сложится, то нетривиальный делитель числа N будет найден за $O(p^{1/2})$ арифметических операций, где p — наименьший простой делитель N, т.е. за $O(N^{1/4})$ арифметических операций.

В силу оценки сложности $O(p^{1/2})$ алгоритм удобен для нахождения не очень больших делителей p числа N, если они есть.

В качестве $f(x)$ обычно выбираются многочлены вида $x^2 + a$. Например, можно взять $f(x) = x^2 + 1$ или $f(x) = x^2 - 1$. В то же время выбор $f(x) = x^2 - 2$ и $x_1 = 2$ не очень удачен, так как в этом случае последовательность x_k имеет период 1.

Приведем теперь некоторые правдоподобные соображения в пользу того, что длина периода и предпериодическая часть последовательности $x_{k+1} \equiv f(x_k) \pmod p$ оцениваются сверху величиной $O(p^{1/2})$.

2.10.3. Парадокс дней рождения.

Пусть $\mathcal{R}$ - множество, состоящее из r элементов. Из принципа ящиков Дирихле следует, что в любой последовательности $z_1, z_2, \ldots z_{r+1}$, $z_i \in \mathcal{R}$, обязательно найдутся два одинаковых элемента. Но можно проверить,

что выбирая случайным образом множество $z_1, z_2, \ldots, z_m$ с теми же условиями на z_i при достаточно большом, но всё же существенно меньшем r значении m, можно с вероятностью, большей $\frac{1}{2}$ попасть на набор, имеющий два одинаковых элемента. Этот кажущийся парадокс носит название "парадокс дней рождения" по причине, о которой мы расскажем немного позже, а сейчас докажем, при некоторых естественных предположениях, указанную оценку на вероятность "хорошей" выборки.

Будем выбирать случайным образом наборы

$$\overline{z} = \{z_1, \ldots, z_m\}, \qquad z_i \in \mathcal{R}, \tag{2.37}$$

предполагая, что все они равновероятны. Какова вероятность того, что выбранная таким образом совокупность $z_1, z_2, \ldots, z_m$, содержит по крайней мере два одинаковых элемента?

Каждое значение z_i может равняться одному из r элементов множества $\mathcal{R}$. Значит количество последовательностей вида (2.40) равно r^m. Сколько же среди них последовательностей с различными элементами. Попробуем построить такие последовательности. На первом месте может стоять любое из возможных r значений для z_1. На втором месте может стоять лишь $r-1$ значений, отличных от стоящего на первом месте. На третьем месте может стоять лишь $r-2$ значения, отличных от первых двух. Действуя так далее мы сможем построить $r(r-1)(r-2)\cdots(r-m+1)$ последовательностей, состоящих из m различных значений. Легко видеть, что каждая такая последовательность будет учтена в этом процессе. Следовательно, доля последовательностей с различными значениями z_i будет равна

$$\mu = \frac{r(r-1)...(r-m+1)}{r^m}.$$

Для оценки μ сверху перепишем

$$\mu = \prod_{k=1}^{m-1}\left(1-\frac{k}{r}\right) \quad \text{и} \quad \ln\mu = \sum_{k=1}^{m-1}\ln(1-\frac{k}{r}). \tag{2.38}$$

При любом действительном x, $0 \leqslant x < 1$ справедливо неравенство

$$\ln(1-x) \leqslant -x. \tag{2.39}$$

Для его доказательства рассмотрим непрерывную на промежутке $[0;1)$ функцию $g(x) = x + \ln(1-x)$. Так как производная $g'(x) = \frac{-x}{1-x}$ отрицательна на интервале $(0;1)$, то $g(x)$ убывает на множестве $0 \leqslant x < 1$, и, значит, на этом множестве выполняется неравенство (2.39).

Взяв $x = \frac{k}{r}$, находим $\ln(1 - \frac{k}{r}) < -\frac{k}{r}$, $1 \leqslant k < m$ и из (2.38)

$$\ln \mu < -\sum_{k=1}^{m-1} \frac{k}{r} = -\frac{m^2 - m}{2r} < -\frac{(m-1)^2}{2r}.$$

Выберем теперь $m = \lceil \sqrt{2r \ln 2} \rceil + 1$. Тогда $(m-1)^2 = \lceil \sqrt{2r \ln 2} \rceil^2 \geqslant 2r \ln 2$ и $\ln \mu < -\ln 2$. Так доказано неравенство $\mu < \frac{1}{2}$. Заметим, что при $r \geqslant 5$ выполняется $m < r$.

Если увеличить константу $\sqrt{2 \ln 2}$ в определении m, оценка μ уменьшится и, значит, вероятность попасть на набор, имеющий по крайней мере два равных элемента, станет больше $\frac{1}{2}$. Например, в случае $m = \lceil 4\sqrt{r} \rceil$ вероятность попасть на выборку, имеющую 2 или более одинаковых элементов, превосходит $0,9996\ldots$.

Вернёмся теперь к последовательности (2.35) и возьмём постоянную в определении m большую, чем $\sqrt{2 \ln 2}$. Из сказанного выше следует, что для "хорошего" многочлена $f(x)$ и быть может после нескольких случайных выборов числа x_1 мы за $O(\sqrt{p} \ln p)$ арифметических операций сможем найти последовательность $x_1, x_2, \ldots, x_m$, имеющую два равных числа. Длины предпериода a и периода ℓ этой последовательности удовлетворяют неравенству $a + \ell \leqslant M$. Действительно, a есть номер первого из двух равных чисел, и $a + \ell$ - номер второго из них чисел. Это объясняет наблюдение Полларда о длине предпериода и периода последовательности в ρ - методе Полларда.

Конечно, это рассуждение не является доказательством, но оно дает правдоподобное объяснение, почему ρ — метод Полларда достаточно быстро на практике находит небольшие простые делители составных чисел.

Заметим также, что скорость работы этого алгоритма может быть увеличена, если наибольший общий делитель $(x_{2i} - x_i, N)$ вычислять не на каждом шаге. Например, можно для последовательности $i = 0, d, 2d, \ldots$ вычислять $(\prod_{k=i+1}^{i+d}(x_{2k} - x_k), N)$, выбрав d не очень большим.

Теперь мы обсудим алгоритм, также основанный на парадоксе дней рождения и позволяющий найти коллизию у любой хеш-функции $H(x)$. Он носит вероятностный характер. Обозначим буквой r количество всех различных значений функции $H(\overline{x})$, положим также $m = [\sqrt{2r}] + 1$. Не трудно проверить, что при $r \geqslant 6$ выполняется неравенство $m < r$. Выберем случайно различные аргументы $x_1, \ldots, x_m$ функции $H(\overline{x})$ и вычислим хеш-значения

$$H(x_1), \quad \ldots, \quad H(x_m). \tag{2.40}$$

Если среди этих значений есть одинаковые, то коллизия построена. При выбранном m справедливы неравенства $(m-1)^2 > \lceil\sqrt{2r}\rceil^2 > 2r$ и $\ln\mu < -1$, так что в противном случае доля последовательностей без одинаковых элементов не превосходит e^{-1}. Поэтому $\mu < e^{-1} = 0,367...$, и количество последовательностей с различными хеш-значениями существенно меньше половины всех последовательностей. Это доказывает, что если все значения хеш-функции равновероятны, то при случайном выборе набора аргументов $x_1, x_2, \ldots, x_M$ вероятность получить последовательность значений, имеющую равные элементы, превосходит $1 - e^{-1} > \frac{1}{2}$. Если все элементы выбранной последовательности хеш-значений оказались различными, можно выбрать вторую последовательность. Если опять не повезло, можно выбрать третью последовательность. Вероятность выбрать последовательность с равными элементами после k шагов не меньше $1 - e^{-k}$. С ростом k она очень быстро приближается к 1. К сожалению, этот гарантированно работающий алгоритм требует слишком много времени. Например, в случае $r = 2^{256}$ имеем $m = \lceil\sqrt{2r}\rceil + 1 > 2^128$. Последовательность из 2^{128} хеш-значений имеет огромную длину.

Рассмотрим ещё одну ситуацию, котороая собственно и дала название корневому понижению оценки числа арифметических опереций при нахождении в последовательности равных элементов. Представим себе, что студенческая группа или просто группа знакомых собралась на вечеринку по какому-то поводу. В группе 23 человека и требуется определить, какова вероятность, что среди приглашенных есть двое, дни рождения которых приходятся на одну дату. На первый взгляд кажется, что эта вероятность очень маленькая. Но рассмотрим ситуацию чуть подробнее. Если год, в котором происходит это событие не високосный, т.е. состоит из $p = 365$ дней, то вероятность выбрать 23 различные даты из 365 равна

$$\mu = \prod_{k=1}^{22}\left(1 - \frac{k}{365}\right) = 0,492703\ldots = \\ = \frac{36997978566217959340182499134166757044383351847256064}{75091883268515350125426207425223147563269805908203125}.$$

Последнее значение точное. Таким образом, любой случайный набор из 23 дат невисокосного года с вероятностью большей $\frac{1}{2}$ содержит две одинаковых даты.

Конец восьмой лекции.

Лекция 9.

2.11. Эллиптические кривые.

Многие математические открытия, как и открытия в физике, химии, биологии, других науках, основаны на эксперименте. Один из величайших математиков Л. Эйлер, живший в XVIII веке, нашел и доказал множество фундаментальных математических фактов. Он обнаружил их, выполняя вручную колоссальные по объёму вычисления. Сейчас для этого часто используется компьютер.

Уравнение

$$y^2 = x^3 + ax + b, \tag{2.41}$$

с коэффициентами a, b, удовлетворяющими условию $4a^3+27b^2 \neq 0$, определяет на плоскости кривую, называемую *эллиптической*. Неравенство означает, что многочлен $x^3 + ax + b$ не имеет кратных корней. Докажем этот факт.

Лемма 2. *Многочлен $f(x) = x^3 + ax + b$ имеет кратные корни в том и только том случае, когда $4a^3 + 27b^2 = 0$.*

Доказательство. Обозначим корни производной $f'(x) = 3x^2 + a$ буквами x_1 и $x_2 = -x_1$ Для того, чтобы эти числа были действительными необходимо и достаточно выполнение неравенства $a \leqslant 0$. Подставим эти корни в многочлен $f(x)$ и перемножим получившиеся числа. В результате, учитывая, что $x_1^2 = -\frac{a}{3}$, получим

$$f(x_1)f(-x_1) = (x_1^3 + ax_1 + b)(-x_1^3 - ax_1 + b) = b^2 - (x_1^3 + ax_1)^2 = \\ = b^2 - x_1^2(x_1^2 + a)^2 = b^2 + \frac{4a^3}{27}.$$

Многочлен $f(x)$ имеет кратный корень в том и только том случае, когда он и его производная имеют общий корень, т.е. $f(x_1) = 0$ или $f(-x_1) = 0$, что по доказанному равносильно равенству $b^2 + \frac{4a^3}{27} = 0$ или равенству $4a^3 + 27b^2 = 0$. □

Если многочлен $f(x)$ имеет кратные корни, то кривая может быть параметризована рациональными функциями и её исследование сильно упрощается.

Например, для многочлена $x^3 - 3x + 2 = (x - 1)^2(x + 2)$ выполняется тождество

$$(t^3 - 3t)^2 = (t^2 - 2)^3 - 3(t^2 - 2) + 2,$$

означающее, что при любом t формулы $x = t^2 - 2, y = t^3 - 3t$ задают некоторую точку на кривой $y^2 = x^3 - 3x + 2$. Легко увидеть, что так может быть получена любая точка (x, y) этой кривой. Соответствующее значение переменной t определяется формулой $t = \frac{y}{x-1}$. Теперь достаточно проверить равенства

$$\left(\frac{y}{x-1}\right)^2 - 2 = x + \frac{y^2 - (x-1)^2(x+2)}{(x-1)^2} = x,$$
$$\left(\frac{y}{x-1}\right)^3 - 3\frac{y}{x-1} = \frac{y}{(x-1)^3}(y^2 - (x-1)^2(x+2) + (x-1)^3) = y.$$

Точка кривой $(1, 0)$, для которой выражение $\frac{y}{x-1}$ не определено, получается по указанным формулам при $t = \sqrt{3}$.

Кривая предполагается размещённой на действительной плоскости. В дальнейшем будут рассматриваться такие кривые и над конечными полями. Именно этот случай используется в криптографии.

Эллиптическими называются и другие кривые, уравнения которых с помощью взаимно-обратных преобразований, задаваемых рациональными функциями от переменных, могут быть преобразованы к виду (2.41).

Название эллиптические кривые происходит от названия эллиптические функции. Эти функции не выражаются через элементарные, но хорошо изучены, благодаря важной роли, которую они играют в приложениях. Через них, в частности, можно выразить длину дуги эллипса, с чем и связано их название. Эллиптические кривые параметризуются эллиптическими функциями так же как окружность параметризуется синусом и косинусом угла. В дальнейшем будет предполагаться, что коэффициенты a, b уравнения (2.41) суть рациональные числа. Это условие нужно потому, что дальше будут рассматриваться решения этого уравнения в целых или рациональных числах.

Теория, описывающая свойства решений в рациональных числах или, как мы будем писать в дальнейшем *рациональных решений*, имеет большую историю и богата глубокими результатами. Тем не менее в ней есть естественные открытые вопросы, т.е. вопросы, ответы на которые не известны. Вот некоторые из них.

1. **Есть ли решение?** До сих пор не известен алгоритм, позволяющий для любого данного уравнения вида (2.41) с рациональными коэффициентами a, b определить, есть у него решение в рациональных числах или нет.

2. **Как найти решение, если известно, что оно существует?** То, что ответ на этот вопрос достаточно сложен, показывает следующий

пример. Известно, что уравнение $y^2 = x^3 + 877x$ имеет бесконечно много рациональных решений. Решение этого уравнения с наименьшими знаменателями имеет вид

$$x = \frac{375494528127162193105504069942092792346201}{6215987776871505425463220780697238044100},$$

$$y = \frac{256256267988926809388776834045513089648669153204356603464786949}{490078023219787588959802933995928925096061616470779979261000}.$$

3. **Конечно или бесконечно множество решений?**
4. **Какова структура множества решений?**

2.11.1. Метод касательных

Ещё древние греки знали, что по одному рациональному решению уравнения (2.41) иногда можно построить ещё одно решение, затем ещё и т.д. Метод, которым они пользовались сейчас называется метод касательных. Древнегреческий учёный Диофант Александрийский, живший в III веке, сочинил книгу "Арифметика", посвящённую решению в целых или рациональных числах алгебраических уравнений и систем уравнений. По его имени алгебраические уравнения, которые требуется разрешить в целых или рациональных числах, называются диофантовыми.

В книге кн. VI, под номером 18 рассматривается следующая задача:

Найти такой прямоугольный треугольник, чтобы его площадь, увеличенная на гипотенузу, образовала куб, а периметр был квадратом.

Здесь под кубом имеется в виду куб некоторого рационального числа, а под квадратом - квадрат другого рационального числа.

Решая эту задачу, Диофант выбирает один из катетов треугольника равным 2. Так делать нельзя, если мы хотим найти все решения задачи, но, если нужно найти какое-нибудь решение, такой выбор допустим, и он существенно упрощает задачу. Если обозначить сторону квадрата буквой y, а длину ребра куба буквой x, то данные в условии соотношения приводят к уравнению

$$y^2 = x^3 + 2, \tag{2.42}$$

причём x, y рациональны. Уравнение (2.42) имеет решения $x = -1, y = \pm 1$, но по смыслу задачи x и y должны быть положительными рациональными числами. Поэтому очевидные решения не подходят. Диофант предлагает следующий замечательный приём. Добавим к уравнению (2.42) ещё одно уравнение $2y = 3x + 5$, т.е. рассмотрим систему

уравнений

$$\begin{cases} y^2 = x^3 + 2 \\ 2y = 3x + 5. \end{cases} \tag{2.43}$$

Легко видеть, что второе уравнение системы задаёт прямую - касательную к кривой (2.42),см. левый рис.1, проведённую в известной нам рациональной точке $(-1, 1)$.

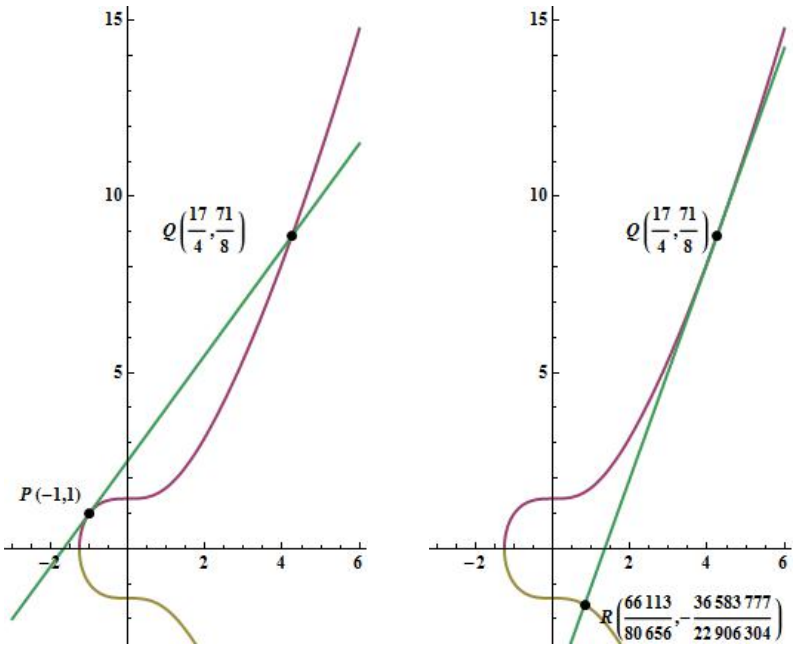

Рис. 2.1. Метод касательных

Решения системы (2.43) соответствуют точкам пересечения эллиптической кривой (2.42) с касательной. Решая систему (2.43), находим

$$(3x + 5)^2 = 4(x^3 + 2)$$

Два корня этого уравнения нам известны до его решения. Это $x_1 = x_2 = -1$, ведь по построению прямая, добавленная к уравнению эллиптической кривой (2.42), касается этой кривой в точке $(-1, 1)$. Пользуясь, например, теоремой Виета, находим $x_1 + x_2 + x_3 = \frac{9}{4}$, откуда следует $x_3 = \frac{17}{4}$ и $y_3 = \frac{3x_3+5}{2} = \frac{71}{8}$. Второй катет z искомого треугольника удовлетворяет равенству

$$2 + z + \sqrt{z^2 + 4} = \left(\frac{71}{8}\right)^2.$$

Отсюда получаем $z^2 + 4 = \left(\frac{4913}{64} - z\right)^2$ и находим второй катет искомого прямоугольного треугольника $z = \frac{24121185}{628864}$.

Заметим, что Диофант не пользовался геометрическими соображениями и работал только с уравнениями.

Процесс построения рациональных точек можно продолжить и далее. Проведём касательную через найденную точку $\left(\frac{17}{4}, \frac{71}{8}\right)$. Эта прямая пересечёт кривую (2.42) ещё в одной рациональной точке $\left(\frac{66113}{80656}, -\frac{36583777}{22906304}\right)$. Так можно построить бесконечную последовательность рациональных точек на кривой (2.42).

2.11.2. Метод секущих.

Следующий шаг в этой истории был сделан спустя много сотен лет двумя великими учёными И. Ньютоном (1643-1727), и Л. Эйлером (1707-1783).

Ньютон не опубликовал своё наблюдение, оно было найдено в его документах после смерти. Эйлер же в 1768 году опубликовал на русском языке учебник "Универсальная арифметика", сыгравший важную роль в преподавании алгебры и арифметики.

Обратимся к левому рисунку

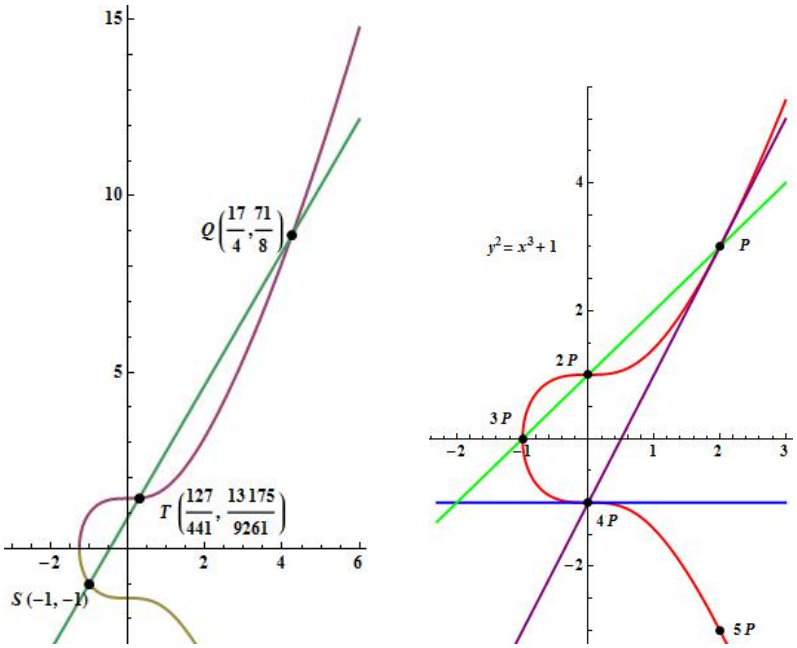

Рис. 2.2. Метод секущих

На нём нарисована кривая (4.9) и через рациональные точки $(-1,-1)$ и $\left(\frac{17}{4}, \frac{71}{8}\right)$ проведена прямая линия

$$y + 1 = \frac{79}{42}(x + 1) \tag{2.44}$$

или $y = \dfrac{79x + 37}{42}$. Решая уравнение $\left(\dfrac{79x + 37}{42}\right)^2 = x^3 + 2$, находим корни $x_1 = -1, x_2 = \frac{17}{4}$ - абсциссы точек, через которые мы проводили

прямую, и $x_3 = \frac{127}{441}$ - абсциссу третьей точки пересечения кривой (2.42) и построенной прямой. Подставляя $\frac{127}{441}$ вместо x в правую часть равенства (2.44) находим ординату третьей точки пересечения $y_3 = \frac{13175}{9261}$. Проводя секущие и касательные через уже построенные точки, мы можем получать всё новые и новые рациональные точки на кривой (2.42).

Конечно, метод секущих и касательных может быть применён к любой кривой вида (2.41). Нужно лишь знать хотя бы одну рациональную точку на ней. Но как уже было написано, именно этот вопрос не решен до сих пор. Не известен алгоритм, который говорил бы для любой кривой вида (2.41), есть на ней рациональная точка или нет. Нет и алгоритма, который находил бы рациональную точку, если известно, что она есть.

Рассмотрим ещё один любопытный пример. На правом рисунке 2 красным цветом изображена кривая, задаваемая уравнением

$$y^2 = x^3 + 1. \tag{2.45}$$

Легко проверить, что это уравнение имеет следующие 5 решений в рациональных числах

$$(-1, 0), \quad (0, -1), \quad (0, 1), \quad (2, -3), \quad (2, 3).$$

На правом рисунке эти пять точек обозначены $P, 2P, 3P, 4P, 5P$. Причина таких обозначений выяснится чуть позже. Здесь же давайте обратим внимание на то, что касательная к любой из этих точек пересекает кривую в одной из точек этого же множества. Например, касательная в точке $4P$ имеет уравнение $y = -1$. Подставляя $y = -1$ в уравнение (2.45), приходим к уравнению $1 = x^3 + 1$, имеющему единственный корень $x = 0$ кратности 3. Новая рациональная точка таким способом не получается. Вертикальная прямая, проходящая через точки P и $5P$, имеет уравнение $x = 2$. Она пересекает кривую (2.45) только в двух точках $(2, \pm 3)$. Третья точка пересечения отсутствует. Формально считают, что любая вертикальная прямая пересекает кривую (2.41) на бесконечности. Так что в рассматриваемом случае третья точка пересечения кривой (2.45) и прямой $y = 2$ находится в бесконечности, т.е. бесконечно удалена от начала координат. Можно ввести так называемые проективные координаты, в них все точки кривой, в том числе и бесконечно удалённая, станут равноправными. Бесконечно удалённую точку часто обозначают буквой O, похожей на ноль.

Итак, касательные и секущие не позволяют построить новую рациональную точку на кривой (2.45). Причина в том, что никаких других рациональных точек кроме пяти указанных $(-1, 0), (0, \pm 1), (2, \pm 3)$ на

этой кривой не существует. Этот непростой факт доказал в 1738 году Л.Эйлер.

2.11.3. Сложение точек на эллиптической кривой

Посмотрим на описанный выше метод проведения секущих и касательных не как на метод размножения рациональных точек, а как на некоторую операцию, выполняемую на множестве точек эллиптической кривой.

Пусть P и Q две точки кривой (2.41). Проведём через P и Q прямую линию и третью точку её пересечения с кривой отразим симметрично относительно оси OX. Получившуюся точку назовём суммой P и Q. Будем обозначать эту точку $P+Q$, см. рисунок 3.

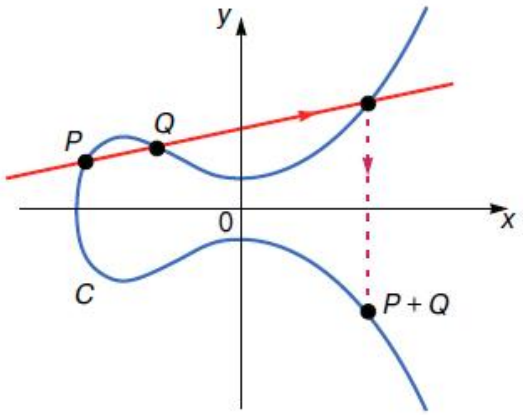

Рис. 2.3. Операция сложения точек на эллиптической кривой.

Если точки P и Q одинаковы, проведём касательную в точке P и точку пересечения с кривой отразим симметрично относительно оси OX. Получившуюся точку будем обозначать $2P$, см. правый рис. 4. Легко проверить с помощью таких же соображений, что $P+O=P$, а также $P+R=O$, см. левый рисунок 4.

Чтобы поупражняться в сложении точек, можно проверить на правом рисунке 2, что

$$P{+}P=2P, \quad P{+}2P=3P, \quad P{+}4P=5P, \quad P{+}5P=O, \quad 4P{+}4P=2P.$$

Операция сложения в координатах задается следующим образом.

Теорема 10. *Если (x_1,y_1) - координаты точки P и (x_2,y_2) - координаты точки Q, причём обе точки отличны от O, а также $Q\neq -P$, то*

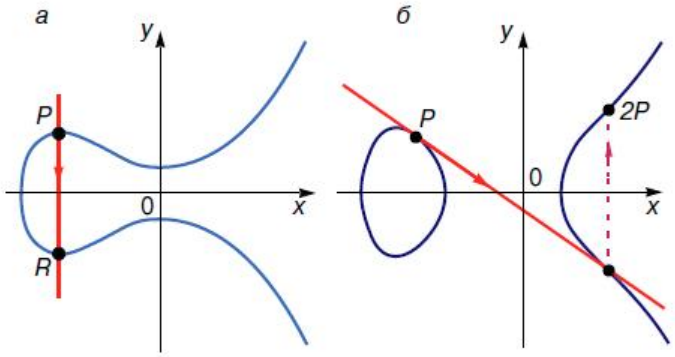

Рис. 2.4. Операция сложения точек на эллиптической кривой.

координаты (x_3, y_3) *точки* $P+Q$ *можно вычислить по формулам*

$$\begin{cases} x_3 = \lambda^2 - x_1 - x_2, \\ y_3 = \lambda(x_1 - x_3) - y_1, \end{cases} \quad \text{где} \quad \lambda = \begin{cases} \frac{y_2-y_1}{x_2-x_1}, & \text{если } x_2 \neq x_1, \\ \frac{3x_1^2+a}{2y_1}, & \text{если } x_2 = x_1 \text{ и } y_2 = y_1. \end{cases} \tag{2.46}$$

Кроме того, по определению для любой точки P *выполняются равенства* $P+O = O+P = O$ *и* $P+(-P) = O$, *где* $-P = (x_1, -y_1)$.

Доказательство. В случае $x_2 \neq x_1$ тангенс угла наклона прямой, проходящей через точки P, Q равен $\frac{y_2-y_1}{x_2-x_1} = \lambda$ и уравнение этой прямой имеет вид

$$y - y_1 = \lambda(x - x_1). \tag{2.47}$$

В этом легко убедиться, подставив в уравнение (2.47) координаты точек P и Q. Из определения точки $P+Q$ следует, что числа $(x_3, -y_3)$ удовлетворяют уравнениям (2.41) и (2.47). Выразив переменную y через x из уравнения (2.47) и подставив получившееся выражение в уравнение (2.41), найдём кубическое уравнение

$$(y_1 + \lambda(x - x_1))^2 = x^3 + ax + b,$$

корнями которого являются абсциссы трёх точек пересечения кривой (2.41) и прямой (2.47), т.е. числа x_1, x_2, x_3. Полученное уравнение может быть переписано также в виде $x^3 - \lambda^2 x^2 + \ldots = 0$, где опущены слагаемые имеющие степень по переменной x меньшую чем 2. Согласно теореме Виета сумма $x_1 + x_2 + x_3$ должна быть равна λ^2, откуда следует выражение для x_3 из равенств (2.46). Точка $(x_3, -y_3)$ лежит на прямой PQ. Из (2.47) теперь следует равенство $-y_3 - y_1 = \lambda(x_3 - x_1)$ и далее выражение для y_3 из (2.47). В первом случае терема доказана.

Рассмотрим теперь второй случай, когда $Q = P$. В этом случае вместо секущей прямой PQ рассматривается касательная к кривой (2.41),

проведённая в точке P. В случае $y_1 = 0$ имеем равенство $P = -P$ и находим $P + P = P + (-P) = O$. Теорема доказана и в этом случае.

Далее считаем $Q = P$ и $y_1 \neq 0$. В этом случае можно найти на плоскости маленькую окрестность точки $P = (x_1, y_1)$, в которой уравнение (2.41) определяет $y = y(x)$ как однозначную функцию от переменной x, а именно $y(x) = \sqrt{x^3 + ax + b}$, где ветвь квадратного корня выбрана так, чтобы значение $\sqrt{x_1^3 + ax_1 + b}$ равнялось y_1. Эта функция удовлетворяет в малой окрестности точки x_1 удовлетворяет тождеству $y(x)^2 = ax^3 + ax + b$. Дифференцируя это тождество по переменной x, находим $2y(x)y'(x) = 3x^2 + a$. Подставляя в полученное тождество $x = x_1$ и пользуясь равенством $y(x_1) = y_1$, находим

$$y'(x_1) = \frac{3x_1^2 + a}{2y_1} = \lambda.$$

Так что в этом случае число λ равно тангенсу угла наклона касательной к кривой, определённой равенством (2.41), проведённой в точке P. Далее все рассуждения проводятся в точности так же, как и в первом случае, с учётом равенства $Q = P$. □

Введённое выше определение суммы точек на первый взгляд кажется странным, однако оно обладает рядом удобных и полезных свойств: точка O играет такую же роль, как и ноль в множестве целых чисел, для каждой точки P есть единственная обратная, обозначаемая $-P$, т.е. точка с условием $P + (-P) = O$, операция сложения коммутативна, т.е. $P + Q = Q + P$ для любых двух точек P, Q и ассоциативна, т.е. для любых трёх точек P, Q, R эллиптической кривой выполняется равенство $(P+Q)+R = P+(Q+R)$ (доказательство последнего факта не просто). Множества с такой операцией называют *группами*.

В 1901г. выдающийся французский учёный А. Пуанкаре (1854-1912) высказал предположение, что *все рациональные точки эллиптической кривой могут быть получены из конечного числа проведением хорд и касательных*. По другому: *группа рациональных точек имеет конечное число образующих*.

Выше мы показали, что для кривой $y^2 = x^3 + 1$, см. правый рис. 2, все рациональные точки могут быть получены с помощью сложения из одной точки $P = (2, 3)$. Здесь, конечно, используется результат Эйлера, что других рациональных точек, кроме пяти указаных и точки O на этой кривой не лежит.

Гипотезу Пуанкаре доказал в 1922 году английский математик Л. Морделл (1888-1972).

Приведём ещё один пример.

Кривая $y^2 = x^3 + 877x$ *имеет бесконечно много рациональных точек. Точка*

$$x = \frac{375494528127162193105504069942092792346201}{62159877768715054254632207806972380441 00},$$

$$y = \frac{256256267988926809388776834045513089648669153204356603464786949}{490078023219787588959802933995928925096061616470779979261000}.$$

есть её единственная образующая.

Известен пример эллиптической кривой с числом образующих, не меньшим 28, но точное число образующих в этом случае не известно. Кривые с числом образующих большим 28 в настоящее время не найдены. Тем не менее предполагается, что

Существуют эллиптические кривые со сколь угодно большим числом независимых образующих.

Но эта гипотеза до сих пор не доказана. Далее мы не будем встречаться с эллиптическими кривыми, определёнными над полем рациональных или действительных чисел.

2.11.4. Эллиптические кривые над конечными полями.

Пусть $p > 3$ - простое число. Рассмотрим теперь уравнение (2.41), предполагая, что его коэффициенты принадлежат $\mathbb{F}_p$. Множество пар чисел x, y из $\mathbb{F}_p$, удовлетворяющих уравнению (2.41), также будем называть эллиптической кривой, добавляя при этом "над полем $\mathbb{F}_p$". Эта "кривая"состоит из конечного множества точек, ведь каждая переменная x и y может принимать не более p значений.

Например, кривая, определённая уравнением $y^2 = x^3 + 2$ над множеством $\mathbb{F}_{11}$ состоит из точек

$$(1,5)), (7,2), (6,3), (9,7), (10,1), (4,0), (10,10), (9,4), (6,8), (7,9), (1,6) \tag{2.48}$$

и ещё точки O, которую по аналогии тоже можно называть бесконечно удалённой точкой. Указанная эллиптическая кривая, определённая над полем $\mathbb{F}_{11}$, состоит из 12 точек.

То же уравнение над полем $\mathbb{F}_7$ определяет кривую

$$(0,3), (0,4), (3,1), (3,6), (5,1), (5,6), (6,1), (6,6), \tag{2.49}$$

состоящую из 9 точек: 8 указанных выше точек и ещё бесконечно удалённая точка O.

Точки кривой, определённой над конечным полем, тоже можно складывать. В этом случае, конечно, нельзя проводить касательные, но можно воспользоваться формулами (2.46), выполняя все вычисления в поле $\mathbb{F}$ по указанным в (2.46) формулам. И в этом случае точка O будет играть роль нуля, а операция сложения будет коммутативной и ассоциативной.

Например, если в случае кривой $y^2 = x^3 + 2$ обозначить $P = (1, 5)$, то все точки в (2.48) будут иметь вид

$$P,\ 2P,\ 3P,\ \ldots\ 11P,$$

а $12P = O$.

А для каждой точки Q кривой (2.49) выполняется равенство $3Q = O$.

Конец девятой лекции и части 1 .

Глава 3.

Криптографические примитивы.

Лекция 10.

Методы шифрования можно разделить на два типа: на *симметричные* и *асимметричные*.

При симметричном шифровании и для зашифровки, и для расшифровки сообщения применяется один и тот же ключ. Поэтому для безопасности передачи зашифрованного сообщения по открытым каналам необходимо предварительно обеим сторонам в тайне выбрать единый ключ, и лишь после этого начинать обмен информацией. При таком способе шифрования возникают очевидные трудности, связанные с необходимостью либо предварительной встречи, либо поиска надежного курьера, которому можно доверить секретный ключ. В частности, организовать надежный конфиденциальный канал связи для большого числа абонентов при таком способе шифрования довольно затруднительно.

Общая идея асимметричных методов шифрования заключается в том, что для зашифровки сообщения применяется один ключ, а для расшифровки — другой. Ключ для зашифровки открыто передается по любому каналу связи, в то время как ключ для расшифровки держится в секрете. Эти ключи, как правило, являются элементами некоторых алгебраических структур и связаны друг с другом некоторым соотношением. Но параметры этих структур и соотношение подбираются такими, чтобы время, требуемое для построения секретного ключа по открытому, было несравнимо больше того времени, которое нужно для построения открытого ключа по секретному.

Асимметричные методы шифрования обычно медленнее симметричных. Поэтому, если требуется передать большой объем информации, очень часто тело сообщения шифруют при помощи какого-нибудь симметричного алгоритма, а использующийся при этом ключ — при помощи асимметричного. Подобного рода схемы называются гибридными.

3.1. Алгоритм Диффи–Хеллмана обмена ключами.

Первой работой, в которой была реализована идея асимметричного криптографического преобразования, является работа [10]. Авторы этой работы использовали тот факт, что задача дискретного логарифмирования в мультипликативной группе F_p^* поля F_p, где p — простое число, существенно сложнее задачи возведения в степень.

Отметим, что это обстоятельство можно очень легко использовать для защиты произвольного ресурса, будь то компьютер, или канал связи, от несанкционированного доступа. Для этого нужно выбрать достаточно большое простое число p и какую-нибудь образующую g группы F_p^*. Далее, каждый пользователь "придумывает" себе секретный пароль $t \in \mathbb{Z}$ и вычисляет элемент g^t поля F_p, который система и будет воспринимать как идентификатор данного пользователя. И каждый раз, когда пользователь решает войти в систему, он вводит свой секретный пароль t, система вычисляет g^t и сравнивает результат с идентификатором пользователя. Возможность взлома такой системы зависит от способности злоумышленника вычислять дискретный логарифм по основанию g от идентификатора пользователя, то есть выбор достаточно больших p может обеспечить необходимую безопасность. Отметим однако, что образующую g нужно выбирать не слишком маленькой, чтобы исключить возможность найти g перебором.

Алгоритм же, о котором сейчас пойдет речь, позволяет двум лицам, общающимся по незащищенному каналу связи, без предварительного обмена секретной информацией выработать общий ключ, который будет известен только им двоим. Предположим, абонент A и абонент B выбрали простое число p и образующую g группы F_p^*. Случайным образом абонент A выбирает секретный ключ $a \in \mathbb{Z}$, $2 \leqslant a \leqslant p-2$, а абонент B — секретный ключ $b \in \mathbb{Z}$, $2 \leqslant b \leqslant p-2$.

Алгоритм 1. *Данные: Простое число p, первообразный корень g по модулю p, секретный ключ a абонента A и секретный ключ b абонента B.*
Найти: Общий ключ $k \in F_p^*$, известный только абонентам A и B.
1. Абоненту A вычислить $x = g^a$ и передать результат абоненту B.
2. Абоненту B вычислить $y = g^b$ и передать результат абоненту A.
3. Абоненту A положить $k = y^a$,
4. Абоненту B положить $k = x^b$.

Корректность данного алгоритма следует из равенства $k_B = (g^a)^b = (g^b)^a = k_A$. Открытыми ключами здесь являются элементы g^a и g^b. Они

передаются по незащищенному каналу связи, поэтому могут стать известными третьим лицам. Но если число p было выбрано достаточно большим, то для того, чтобы по известным g^a и g^b при помощи существующих на данный момент алгоритмов найти секретные числа a и b (и, соответственно, g^{ab}), потребуется очень много времени, что и обеспечивает требуемую безопасность. Аналогично, абоненты A и B не смогут за приемлемое время вычислить секретные ключи друг друга, то есть каждый из них при смене собеседника может не менять свой секретный ключ.

В описанном виде схема Диффи-Хеллмана уязвима относительно так называемой "атаки посередине" , в которой абонент - нарушитель C выступает в качестве посредника между A и B.

$$A \xleftarrow{z=g^c} \xrightarrow{x=g^a} C \xleftarrow{y=g^b} \xrightarrow{z=g^c} B$$

Если злоумышленнику удалось убедить абонента A воспринять его как абонента B, а абонента B — соответственно, как A, он, наладив с каждым из них связь, при помощи алгоритма Диффи–Хеллмана, создаёт секретный ключ g^{ac} для обмена информацией с A и секретный ключ g^{bc} для обмена информацией с B. В результате он получает полный контроль за перепиской A и B, включая возможность искажения и передачи ложной информации. Чтобы предотвратить подобную возможность, используют так называемые алгоритмы аутентификации.

Отметим также, что алгоритм 1 по своей структуре несколько отличается от изложенных ранее. Обычно предполагается, что у алгоритма ровно один исполнитель, и по этой причине команды имеют безадресный характер. Что же касается алгоритмов типа алгоритма 1, то они предполагают как минимум двоих исполнителей. Поэтому такие алгоритмы носят название *протоколов*.

3.2. Хеш-функции.

3.2.1. Конструкция Дамгорда-Меркля.

Напомним, что *отображение*

$$h : \{0,1\}^* \mapsto \{0,1\}^n, n \in \mathbb{N}$$

называется хеш-функцией*, если оно однонаправленно, не имеет коллизий и найти для него второй прообраз очень сложно.* Рассмотрим отображение

$$F : \{0,1\}^n \times \{0,1\}^k \mapsto \{0,1\}^n, k \in \mathbb{N}$$

и для конечного сообщения M построим хеш-значение $h(M)$:

1) Дополнить сообщение M до длины, кратной k бит, и разбить полученный набор бит на блоки длины k: $M \to M^{(0)}||\dots||M^{(N-1)}$, $M^{(i)} \in \{0,1\}^k, 0 \le i < N, i \in \mathbb{N}$.

2) Пусть $H^{(0)}$ – заранее фиксированная константа

3) Для всех $0 < i \le N$ вычислить $H^{(i)} := F(H^{(i-1)}, M^{(i-1)})$.

4) Положить значение $h(M) := H^{(N)}$.

Функция F называется *сжимающей функцией*.

Долее мы будем подробно рассматривать функцию хеширования SHA-256. Алгоритм вычисления её значений может использоваться для хеширования любого сообщения M, имеющего длину меньшую, чем 2^{64} бита. В основе его лежит конструкция Дамгорда-Меркля с $n = 256, k = 512$. Алгоритм использует

1) режимы обработки сообщения состоящие из шестидесяти четырех 32-битных слов $W_0, W_1, \dots, W_{63}$,

2) восемь рабочих переменных a, b, c, d, e, f, g и h из 32 бит каждая и

3) хеш-значения из восьми 32-битных слов

$$H^{(i)} = (H_0^{(i)}, H_1^{(i)}, \dots, H_7^{(i)}). \tag{3.1}$$

Конечный результат работы SHA-256 - "отпечаток"сообщения M, записывается 256 битами. Начальное хеш-значение состоит из восьми слов $H_0^{(0)}, H_1^{(0)}, H_2^{(0)}, H_3^{(0)}, H_4^{(0)}, H_5^{(0)}, H_6^{(0)}, H_7^{(0)}$, полученных взятием первых тридцати двух битов дробных частей квадратных корней из первых восьми простых чисел.

Алгоритм хеширования SHA-256 начинается с предварительной подготовки сообщения.

1. **Дополнение сообщения,** Цель этого этапа - добиться, чтобы длина дополненного сообщение была кратна 512 битам. Предположим, что длина сообщения M равна ℓ битов. Добавим в конце сообщения бит 1 и за ним k нулей, где k - наименьшее неотрицательное целое, удовлетворяющее сравнению $\ell + k + 1 \equiv 448 \pmod{512}$. Затем добавим ещё блок из 64 битов, равный числу ℓ, записанному в двоичной системе исчисления. После этого длина дополненного сообщения станет кратной 512. Обозначим её $512 \cdot N$.

2. **Разбиение сообщения.** Сообщение и его дополнение разбиваются на битовые блоки $M^{(1)}, M^{(2)}, \dots, M^{(N)}$, каждый длиной в 512 битов. Так

как эти 512 битов могут быть представлены в виде шестнадцати слов длиной в 32 бита, то первые 32 бита i-го блока сообщения обозначаются символом $M_0^{(i)}$, следующие 32 бита - символом $M_1^{(i)}$ и так далее до последнего слова $M_{15}^{(i)}$.

3.2.2. Логические операции и функции.

Напомним некоторые понятия и определения операций, используемых в дальнейшем.

Бит — это минимальная единица измерения объёма информации, так как она хранит одно из двух значений — 0 или 1. То есть одна битовая ячейка может находиться одновременно лишь в одном состоянии из возможных двух - 0 или 1. Есть определённые операции, для манипуляций с битами. Эти операции называются логическими или булевыми операциями, названные в честь математика Джорджа Буля (1815-1864). Все эти операции могут быть применены к любому биту, независимо от того, какое он имеет значение — 0 или 1. Ниже приведены основные логические операции.

$\wedge$ - Логическая операция И (AND) выполняется с двумя битами, назовем их a и b. Результат выполнения логической операции И будет равен 1, если a и b равны 1, а во всех остальных (других) случаях, результат будет равен 0.

$\vee$ - Логическая операция ИЛИ (OR) выполняется с двумя битами (a и b). Результат выполнения логической операции ИЛИ будет равен 0, если a и b равны 0, а во всех остальных случаях, результат равен 1.

$\oplus$ - Логическая операция исключающее ИЛИ (XOR) выполняется с двумя битами (a и b). Результат выполнения логической операции XOR будет равен 1, если один из битов a или b равен 1, во всех остальных случаях, результат равен 0 (нулю). Эту операцию иногда называют сложением по модулю 2.

$\neg$ - Логическая операция НЕ (NOT, отрицание) выполняется с одним битом. Результат выполнения этой логической операции напрямую зависит от состояния бита. Если бит находился в нулевом состоянии, то результат выполнения NOT будет равен единице и наоборот.

$+$ - Сложение по модулю 2^{32}.

$<<$ - Операция сдвига влево, где $x << n$ получается путем отбрасывания крайних левых n битов слова x, а затем дополнения результата n

нулями справа.

$>>$ - Операция сдвига вправо, где $x >> n$ получается путем отбрасывания крайних правых n битов слова x и последующего дополнения результата n нулями слева.

$ROTL^n(x)$ - Операция поворота влево (круговое смещение влево), где x - это 32-битное слово, а n - целое число с $0 \leqslant n < 32$, определяется как $ROTL^n(x) = (x << n) \vee (x >> 32 - n)$.

$ROTR^n(x)$ - Операция поворота вправо (круговое смещение вправо), где x - это 32 битное слово, а n - целое число с $0 \leqslant n < 32$, определяется как $ROTR^n(x) = (x >> n) \vee (x << 32 - n)$.

$SHR^n(x)$ - Операция сдвига вправо, где x - это 32 - битное слово, а n - целое число с $0 \leqslant n < 32$, определяется как $SHR^n(x) = x >> n$.

Алгоритм SHA-256 использует шесть логических функций, где каждая функция работает с 32-битными словами, которые представлены как x, y и z. Результатом действия каждой функции является новое 32-битное слово.

$$
\begin{aligned}
Ch(x, y, z) &= (x \wedge y) \oplus (\neg x \wedge z), \\
Maj(x, y, z) &= (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z), \\
\sum_0(x) &= ROTR^2(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x), \\
\sum_1(x) &= ROTR^6(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x), \\
\sigma_0(x) &= ROTR^7(x) \oplus ROTR^{18}(x) \oplus SHR^3(x), \\
\sigma_1(x) &= ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x).
\end{aligned}
$$

Здесь Maj (majority) - обозначает большинство: для каждого индекса бита этот результирующий бит соответствует большинству из 3 входных битов для x, y и z в этом индексе. Ch происходит от Choice (выбор). Для каждого индекса результирующий бит соответствует биту из y (или соответственно z) в этом индексе, в зависимости от того, равен ли бит из x в этом индексе 1 (или, соответственно, 0).

SHA-256 использует последовательность из шестидесяти четырех постоянных 32-разрядных слов: $K_0, K_1, \ldots, K_{63}$. Эти слова представляют собой первые тридцать два бита дробных частей кубических корней из первых шестидесяти четырех простых чисел.

3.2.3. Предварительная подготовка.

Предварительная подготовка состоит из трех этапов:

1. **Дополнение сообщения,** Цель этого этапа - добиться, чтобы длина дополненного сообщение была кратна 512 битам. Предположим, что длина сообщения M равна ℓ битов. Добавим в конце сообщения бит 1 и за ним k нулей, где k - наименьшее неотрицательное целое, удовлетворяющее сравнению $\ell + k + 1 \equiv 448 \pmod{512}$. Затем добавим ещё блок из 64 битов, равный числу ℓ, записанному в двоичной системе исчисления. После этого длина дополненного сообщения станет кратной 512. Будем считать, что длина дополненного сообщения равна $512N$.

2. **Разбиение сообщения.** Сообщение и его дополнение разбиваются на N 512 - битовых блоков $M^{(1)}, M^{(2)}, \ldots, M^{(N)}$. Так как 512 битов входного блока могут быть представлены в виде шестнадцати слов длиной в 32 битов, то первые 32 бита i-го блока сообщения обозначаются символом $M_0^{(i)}$, следующие 32 бита - символом $M_1^{(i)}$ и так далее до последнего слова $M_{15}^{(i)}$.

3. **Установка начального хеш-значения.** Этот вектор из восьми 32-битных слов $H_0^{(0)}, H_1^{(0)}, H_2^{(0)}, H_3^{(0)}, H_4^{(0)}, H_5^{(0)}, H_6^{(0)}, H_7^{(0)}$ уже описан ранее.

3.2.4. Сжимающая функция для SHA-256.

Алгоритм SHA-256 может использоваться для хеширования любого сообщения M, имеющего длину меньшую, чем 2^{64} бита. Алгоритм использует

1) режимы обработки сообщения из шестидесяти четырех 32-битных слов,

2) восемь рабочих переменных из 32 бит каждая и

3) хеш-значение из восьми 32-битных слов.

Конечный результат работы SHA-256 - "отпечаток"сообщения, записывается 256 битами. Слова режимов обработки сообщения помечаются буквами $W_0, W_1, \ldots, W_{63}$. Восемь рабочих переменных помечены как a, b, c, d, e, f, g и h. Слова значения хеша помечены символами

$$H^{(i)} = (H_0^{(i)}, H_1^{(i)}, \ldots, H_7^{(i)}). \tag{3.2}$$

При $i = 0$ этот вектор совпадает с начальным хеш-значением, а затем он заменяется каждым последующим промежуточным хеш-значением (3.2) (после обработки каждого блока сообщения), и заканчивая конечным значением хеша, $H^{(N)}$, SHA-256 также использует два временных слова, T_1 и T_2.

При вычислении значения хеш-функции SHA-256 каждый блок сообщения, $M^{(1)}, M^{(2)}, \ldots, M^{(N)}$, обрабатывается по порядку, на следующих шагах.

От $i = 1$ до N выполнить следующие пункты 1-4:

1. Установить слова режимов обработки сообщения

$$W_t = \begin{cases} M_t^{(i)}, & 0 \leqslant t \leqslant 15; \\ \sigma_1(W_{t-2}) + W_{t-7} + \sigma_0(W_{t-15}) + W_{t-16}, & 16 \leqslant t \leqslant 63. \end{cases}$$

2. Инициировать рабочие переменные a, b, c, d, e, f, g и h, полагая их равными восьми 32-битовым словам $H_j^{(i-1)}$, $0 \leqslant j \leqslant 7$, а именно

$$\begin{aligned} a &= H_0^{(i-1)}, \quad b = H_1^{(i-1)}, & c &= H_2^{(i-1)}, \quad d = H_3^{(i-1)}, \\ e &= H_4^{(i-1)}, \quad f = H_5^{(i-1)}, & g &= H_6^{(i-1)}, \quad h = H_7^{(i-1)}. \end{aligned}$$

3. От $t = 0$ до $t = 63$ выполнить указанные в этом пункте операции

$$\begin{aligned} T_1 &= h + \Sigma_1(a) + Ch(e, f, g) + K_t + W_t, \\ T_2 &= \Sigma_0(a) + Maj(a, b, c), \\ h &= g, \quad g = f, \quad f = e, \quad e = d + T_1, \\ d &= c, \quad c = b, \quad b = a, \quad a = T_1 + T_2. \end{aligned}$$

4. Вычислить промежуточное хеш-значение с номером i

$$\begin{aligned} H_0^{(i)} &= a + H_0^{(i-1)}, \quad H_1^{(i)} = b + H_1^{(i-1)}, \quad H_2^{(i)} = c + H_2^{(i-1)}, \\ H_3^{(i)} &= d + H_3^{(i-1)}, \quad H_4^{(i)} = e + H_4^{(i-1)}, \quad H_5^{(i)} = f + H_5^{(i-1)}, \\ & \qquad\qquad H_6^{(i)} = g + H_6^{(i-1)}, \quad H_7^{(i)} = h + H_7^{(i-1)}. \end{aligned}$$

После повторения шагов с первого по четвёртый в совокупности N раз, т.е. после обработки блока $M^(N)$, получится 256-битный отпечаток сообщения M, а именно

$$h(M) = H_0^{(N)} || H_1^{(N)} || H_2^{(N)} || H_3^{(N)} || H_4^{(N)} || H_5^{(N)} || H_6^{(N)} || H_7^{(N)}.$$

Конец десятой лекции.

Лекция 11.

3.3. Основные алгоритмы шифрования с открытым ключом.

3.3.1. Криптография с открытым ключом.

В криптосистемах, о которых пойдет речь в этом разделе, используются два ключа: открытый и секретный. Открытый ключ может быть опубликован в справочнике наряду с именем пользователя. В результате любой желающий может зашифровать с его помощью свое письмо и послать закрытую информацию владельцу соответствующего секретного ключа. Расшифровать посланное сообщение сможет только тот, у кого есть секретный ключ.

Главное из преимуществ системы шифрования с открытым ключом состоит в том, что стороны, даже не подозревавшие о существовании друг друга до первого письма, могут успешно обмениваться информацией, зашифрованной с помощью криптосистемы с открытым ключом. Каждый может послать Алисе секретную информацию, воспользовавшись ее открытым ключом. Но только Алиса в состоянии расшифровать сообщение, поскольку лишь у нее есть соответствующий секретный ключ. Первая (и наиболее успешная) криптосистема с открытым ключом, а именно, RSA, была опубликована в 1977 году.

3.3.2. Шифрование RSA.

Алгоритм RSA, один из первых алгоритмов шифрования с открытым ключом, достойно выдержал испытание временем. Этот алгоритм основывается на задаче RSA, с которой мы познакомились во вводной части. Как Вы помните, она сводится к поиску простых делителей больших натуральных чисел. Так что можно утверждать, что криптостойкость алгоритма RSA базируется на сложности проблемы факторизации, хотя и не в полной мере, поскольку иногда задачу RSA можно решать не прибегая к разложению на множители. Такие примеры мы рассмотрим ниже.

Стоит отметить, что не все наборы (p, q, d), пусть даже числа p и q велики, представляются надежными. Например, если число $n + 1$ или число $n - 1$ легко раскладывается на множители (скажем, является степенью двойки), то для числа n можно успешно применить так называемые $(N \pm 1)$–методы факторизации, то есть секретный ключ в этом

случае найдется довольно быстро. Кроме того, числа p и q не должны быть слишком близкими, так как тогда они будут близки к $\sqrt{n}$ и число n можно будет быстро разложить на множители при помощи метода факторизации Ферма. Необходимо также, чтобы ни $p-1$, ни $q-1$ не являлось произведением маленьких простых чисел, ибо в этом случае число n можно будет быстро факторизовать при помощи так называемого $(p-1)$–метода Полларда.

Напомним схему шифрования RSA. Предположим, Алиса считает нужным разрешить всем желающим отправлять ей секретные сообщения, расшифровать которые способна только она. Тогда Алиса подбирает два больших простых числа p и q. Держа их в секрете, Алиса публикует их произведение $N = p \cdot q$. Кроме того, Алиса выбирает шифрующий показатель степени E, удовлетворяющий условию НОД $(E, (p-1), (q-1)) = 1$. Как правило E берут равным 3, 17 или 65 537. Эти числа записываются в двоичной системе малым количеством ненулевых битов. Пара, доступная всем желающим, — это (N, E). Для выбора секретного ключа Алиса применяет расширенный алгоритм Евклида к паре чисел E и $(p-1)(q-1)$, получая при этом два целых числа d, f с условием $Ed + f \cdot (p-1)(q-1) = 1$. Расшифровывающий показатель степени d удовлетворяет сравнению

$$Ed \equiv 1 \pmod{(p-1)(q-1)}. \tag{3.3}$$

Секретным ключом является тройка (d, p, q).

Допустим теперь, что Боб намерен зашифровать сообщение, адресованное Алисе. Он сверяется с открытым ключом и разбивает сообщение на блоки M, строго меньшие числа N. Шифртекст C получается из M по следующему правилу:

$$C \equiv M^E \pmod{N}.$$

Алиса, получив шифрограмму, расшифровывает ее, возводя число C в степень d:

$$M \equiv C^d (mod N).$$

Последнее сравнение выполняется в связи с тем, что

$$C^d \equiv M^{dE} \equiv M \pmod{N}$$

и в силу (3.3).

Рассмотрим теперь некоторые примеры.

Пример 1: Одинаковые сообщения и модули. Допустим, Алиса посылает одно и то же сообщение M двум клиентам криптосистемы, открытые ключи которых (N, E_1) и (N, E_2), причём открытые показатели

степени E_1, E_2 взаимно просты. Ева,не принадлежащая к пользователям криптосистемы, использующим общий модуль N, но желающая прочитать сообщение Алисы, видит только зашифрованные сообщения

$$C_1 \equiv M^{E_1} \pmod{N}, \qquad C_2 \equiv M^{E_2} \pmod{N}.$$

Для расшифровки сообщения она может вычислить целые числа T_1, T_2, для которых $E_1T_1 + E_2T_2 = 1$, и восстановить сообщение M следующим способом

$$C_1^{T_1}C_2^{T_2} \equiv M^{E_1T_1+E_2T_2} \equiv M \pmod{N}. \tag{3.4}$$

Рассмотрим случай

$$N = 18923, \qquad E_1 = 11, \qquad E_2 = 5.$$

Предположим, что перехвачены шифртексты

$$C_1 = 1514 \qquad \text{и} \qquad C_2 = 8189,$$

соответствующие одному открытому тексту M. Решая уравнение $15T_1 + 5T_2 = 1$, Ева находит $T_1 = 1$ и $T_2 = -2$, после чего раскрывает исходную информацию: $M \equiv C_1 \cdot C_2^{-2} \equiv 100 \pmod{N}$.

Пример 2: малые шифрующие показатели. Если в криптосистеме RSA используются небольшие шифрующие экспоненты E, то могут возникнуть проблемы. Предположим, что у нас есть три пользователя с различными взаимно простыми модулями шифрования N_1, N_2, N_3 и одним шифрующим показателем $E = 3$. Пусть некто посылает им одно сообщение Часто приходится рассылать нескольким людям одинаковые сообщения, например, уведомления о встрече. В нашем случае нападающий видит три криптограммы:

$$C_1 \equiv m^3 \pmod{N_1}, \qquad C_2 \equiv m^3 \pmod{N_2}, \qquad C_3 \equiv m^3 \pmod{N_3}.$$

Решая, например, с помощью китайской теоремы об остатках, систему сравнений

$$X \equiv C_1 \pmod{N_1}, \quad X \equiv C_2 \pmod{N_2}, \quad X \equiv C_3 \pmod{N_3}, \tag{3.5}$$

он находит целое число X_0, удовлетворяющее сравнениям

$$X_0 \equiv m^3 \pmod{N_1}, \quad X_0 \equiv m^3 \pmod{N_2}, \quad X_0 \equiv m^3 \pmod{N_3},$$

и, значит, сравнению

$$X_0 \equiv m^3 \pmod{N_1N_2N_3}.$$

Так как $m < N_i, i = 1, 2, 3$, заключаем $m^3 < N_1 N_2 N_3$. Поэтому целые числа X_0 и m^3 должны совпадать. Вычисляя кубический корень из X_0, мы раскрываем сообщение m.

Рассмотрим конкретный случай

$$N_1 = 323, \qquad N_2 = 299, \qquad N_3 = 341$$

и допустим, Ева перехватила сообщения

$$C_1 = 50, \qquad C_2 = 268, \qquad C_3 = 1.$$

Решая соответствующую систему сравнений (3.5), Ева находит

$$X_0 \equiv 300763 \pmod{N_1 N_2 N_3}.$$

Решая уравнение $m^3 = 300763$, Ева находит $m = 300763^{1/3} = 67$.

Заметим, что в обоих рассмотренных примерах мы раскрыли текст, не используя разложение целых чисел на множители. Эти примеры также говорят о необходимости дополнять оригинальные сообщения случайными цифрами перед шифрованием. Например, можно дописывать к сообщению время, начала шифрования. Кроме того нужно избегать маленьких шифрующих показателей.

3.4. Об уязвимости системы RSA.

Общий модуль. Если пары ключей выдаются пользователям одним центральным распределяющим органом, то может возникнуть желание зафиксировать единый для всех модуль $n = pq$ и генерировать лишь пары чисел e и d. Как оказывается, данный подход нарушает безопасность системы, ибо, как видно из следующей теоремы, любой пользователь, обладающий открытым и секретным ключами, может за короткое время найти разложение $n = pq$ и, таким образом, способен быстро восстанавливать секретный ключ по любому открытому.

Теорема 11. *Пусть натуральные числа n, e, d удовлетворяют условиям $ed \equiv 1 \pmod{\varphi(n)}$, $e < \varphi(n)$, $d < \varphi(n)$ и пусть $n = pq$, где p и q — нечетные простые числа. Тогда существует полиномиальный вероятностный алгоритм, при помощи которого, зная числа n, e, d, можно найти разложение на множители числа n.*

Доказательство. Положим $r = ed - 1$. Так как r делится на $\varphi(n)$, то для любого числа g, взаимно простого с n, справедливо сравнение

$$g^r = (g^{\varphi(n)})^{\frac{ed-1}{\varphi(n)}} \equiv 1 \pmod{n}.$$

Но $\varphi(n)$ делится на 4, стало быть, $r = 2^s t$, где $s \geqslant 2$, а t нечетно. Согласно теореме Эйлера имеем

$$n | g^r - 1 = g^{2^s t} - 1 = (g^t - 1)(g^t + 1)(g^{2t} + 1) \cdots (g^{2^{s-1}t} + 1). \qquad (3.6)$$

Оценим количество чисел g, взаимно простых с n и удовлетворяющих хотя бы одному из сравнений

$$g^t \equiv 1 \pmod{n}, \qquad g^{2^k t} \equiv -1 \pmod{n}, \qquad 0 \leqslant k < s. \qquad (3.7)$$

Обозначим $c = \min(\nu_2(p-1), \nu_2(q-1)) \geqslant 1$, т.е. - наибольшее неотрицательное целое число с условием $2^c | (p-1), 2^c | (q-1)$. Не уменьшая общности можно считать, что $c = \nu_2(p-1) \leqslant \nu_2(q-1)$. Сравнение $x^t \equiv 1 (\mod p)$ имеет столько же решений x, сколько имеется чисел $\operatorname{ind} x$, удовлетворяющих линейному сравнению $t \cdot \operatorname{ind} x \equiv 0 \pmod{p-1}$, т.е. $(t, p-1)$. Аналогичная оценка имеет место и для количества решений сравнения $x^t \equiv 1 \pmod{q}$. Согласно китайской теореме об остатках заключаем, что количество решений сравнения $x^t \equiv 1 \pmod{n}$ не превосходит

$$(t, p-1) \cdot (t, q-1) = (t, \frac{p-1}{2^c})(t, \frac{q-1}{2^c}) \leqslant \frac{p-1}{2^c} \cdot \frac{q-1}{2^c}.$$

Обозначим $T = \frac{(p-1)(q-1)}{4^c}$.

Если сравнение

$$x^{2^k t} \equiv -1 (\mod n) \qquad (3.8)$$

разрешимо, то разрешимо будет такое же сравнение по модулю p и, значит, сравнение

$$2^k t \cdot \operatorname{ind} x \equiv \frac{p-1}{2} \pmod{p-1}. \qquad (3.9)$$

Условием разрешимости этого сравнения является

$$(2^k t, p-1) | \frac{p-1}{2} \qquad (3.10)$$

При $k \geqslant c$ левая часть условия (3.10) делится на 2^c, а правая нет. Значит, сравнение (3.8) в этом случае неразрешимо.

Если же $k < c$, то сравнение (3.9), а вместе с ним и сравнение $x^{2^k t} \equiv -1 (\mod p)$ имеет не более $p-1$ решений. Теперь, согласно китайской теореме об остатках можно утверждать, что сравнение (3.8) имеет не более $(p-1)(q-1) = 4^k T$ решений. Таким образом, количество чисел g, удовлетворяющих хотя бы одному из сравнений (3.7), может быть оценено сверху величиной

$$T \left(1 + \sum_{k=0}^{d-1} 4^k \right) = T \frac{4^d + 2}{3} \leqslant T \frac{4^d}{2} = \frac{\varphi(n)}{2}.$$

Таким образом, множество S целых чисел g в промежутке от 1 до n, взаимно простых с n и не удовлетворяющих ни одному из сравнений (3.7), содержит не менее $\frac{\varphi(n)}{2}$ элементов.

Выбрав случайным образом на промежутке $[1; n]$ целое число g, взаимно простое с n, мы с вероятностью, не меньшей $\frac{1}{2}$ получим $g \in S$. Число n делит произведение скобок в (3.6), но не делит ни одной из этих скобок, значит, простые делители p и q должны разделиться. Каждый из них войдёт войдёт в какую-то скобку и эти скобки различны. Проведём это рассуждение чуть более формально.

Пусть m — наименьшее целое число с условием $g^{2^{m+1}t} \equiv 1 \pmod{n}$. Так как $g \in S$ имеем $0 \leqslant m < s$. Пусть $h = g^{2^m t}$. Согласно определению m имеем $h \not\equiv 1 \pmod{n}$, а, поскольку $g \in S$, заключаем, что $h \not\equiv -1 \pmod{n}$. В то же время из сравнений $h^2 \equiv 1 \pmod{p}$, $h^2 \equiv 1 \pmod{q}$ следует, что $h \equiv \pm 1 \pmod{p}$, $h \equiv \pm 1 \pmod{q}$. По доказанному знаки в правых частях этих двух сравнений различны. Но тогда наибольший общий делитель $(h-1, n)$ отличен от 1 и n. Значит, он будет нетривиальным делителем n, а потому будет равен p или q.

Полиномиальность приведенного вероятностного алгоритма очевидна. □

Малый секретный ключ. Поскольку при применении алгоритма RSA требуется возводить в степень d, может возникнуть желание взять в качестве d не очень большое число. Однако, как видно из следующей теоремы, доказанной в [**?**], при малом d систему можно взломать за весьма короткое время.

Теорема 12. *Пусть n, e, d — натуральные числа, $ed \equiv 1 \pmod{\varphi(n)}$, $e < \varphi(n)$, $d < n^{1/4}/3$ и пусть $n = pq$, где p и q — нечетные простые числа, такие что $q < p < 2q$. Тогда существует полиномиальный алгоритм, при помощи которого, зная числа n и e, можно найти число d.*

Доказательство. Поскольку $ed \equiv 1 \pmod{\varphi(n)}$, существует такое целое число k, что

$$ed - k\varphi(n) = 1. \tag{3.11}$$

Кроме того, $\varphi(n) = n - p - q + 1$, то есть

$$|n - \varphi(n)| < 3\sqrt{n} - 1. \tag{3.12}$$

Из (3.11) и (3.12) получаем, что

$$\left|\frac{e}{n}-\frac{k}{d}\right| = \left|\frac{ed-k\varphi(n)-kn+k\varphi(n)}{nd}\right| = \\ = \left|\frac{1-k(n-\varphi(n))}{nd}\right| < \left|\frac{3k\sqrt{n}}{nd}\right| = \frac{3k}{d\sqrt{n}}\,.$$

При этом из соотношения $k\varphi(n) = ed - 1 < ed$ и неравенства $e < \varphi(n)$ следует, что $k < d < n^{1/4}/3$. Стало быть,

$$\left|\frac{e}{n}-\frac{k}{d}\right| < \frac{3k}{d\sqrt{n}} < \frac{1}{dn^{1/4}} < \frac{1}{3d^2} < \frac{1}{2d^2}\,.$$

Из теории цепных дробей известно, что в случае выполнения такого неравенства число k/d является некоторой подходящей дробью числа e/n. В силу неравенства (3.11) числа k и d взаимно просты, поэтому d в точности совпадает со знаменателем одной из подходящих дробей числа e/n. Количество же этих дробей есть величина порядка $O(\ln n)$, то есть число d восстанавливается за линейное время. □

Глава 4.

Информационные технологии

Лекция 12.

4.1. Электронная подпись.

Электронная подпись - это число фиксированной длины, зависящее от подписываемого сообщения и от секретного ключа того, кто этот документ подписывает. Подпись может зависеть также от некоторого случайного параметра, созданного в процессе работы алгоритма подписания. Это значит, что алгоритм, вырабатывающий подпись под одним и тем же документом и с одним и тем же секретным ключом, может привести к различным подписям. Этим электронная подпись принципиально отличается от "ручной"подписи. Зависимость подписи от подписываемого сообщения делает невозможным популярный алгоритм подделки "ручной"подписи: скопировать подпись в компьютер и перенести её под другой документ. Иногда для электронной подписи используется название "цифровая подпись". Эти два названия обозначают одно и то же понятие.

Электронная подпись должна иметь некоторые важные свойства.

- Подпись должна легко проверяться. Для этого служит открытый ключ подписывающего лица. С помощью этого ключа и открытой информации каждый человек должен иметь возможность подтвердить целостность и подлинность электронного сообщения.

- Подтверждать подлинность источника сообщения.

- Гарантировать, что лицо, подписавшее документ не сможет отказаться от своей подписи.

- Невозможность подделки подписи посторонним лицом.

В соответствии со сказанным, алгоритм электронной подписи состоит из двух частей: алгоритм подписания и алгоритм проверки. В дальнейшем символом $A \pmod B$, где A, B - целые числа, будет обозначаться

наименьший неотрицательный остаток при делении A на B.

Ниже мы рассматриваем некоторые примеры электронной подписи.

4.1.1. Подпись Шнорра.

Этот алгоритм электронной подписи, так же как и следующий далее алгоритм DSA основаны на сложности решения задачи лискретного логарифмирования. Аналогично алгоритму Эль-Гамаля здесь также используются простые числа P и Q, причём P-1 делится на Q. Размеры чисел должны быть большими. Например, число Q записывается 256 битами, а в записи P используется от 512 до 2048 битов. Выбирается какое-нибудь число G, не делящееся на P, с условием $G^Q \equiv 1 \pmod{P}$. Чтобы построить такое число, можно, выбирая случайным образом число $b, 1 \leqslant b < P$, вычислять $G \equiv b^{\frac{P-1}{Q}}$. Делать это нужно до тех пор, пока не будет найдено число G отличное от 1. Такое G будет удовлетворять условию $G^Q \equiv 1 \pmod{P}$ и будет порождать в $\mathbb{F}_P^*$ мультипликативную группу порядка Q. Числа P, Q, G общеизвестны.

Алиса желает передать в Банк подписанное ею сообщение m. Для этого она выбирает случайное число $d, 0 < d < Q$ (её секретный ключ) и вычисляет открытый ключ $Y \equiv G^d \pmod{P}$. Этот ключ находится в открытом доступе, например в какой-нибудь библиотеке. Восстановить по нему секретный ключ d практически невозможно ввиду большой сложности задачи дискретного логарифмирования при указанных выше размерах параметров.

Чтобы подписать сообщение, Алиса должна проделать следующие операции.

Алгоритм 11 (Алгоритм подписания.)**.** *Даны числа P, Q, G с указанными выше свойствами, хеш-функция $h(\cdot)$ и сообщение m.*

1. Выбрать случайное число $k, 0 < k < Q$, и вычислить

$$R \equiv G^k \pmod{P}.$$

2. Вычислить значение хеш-функции $E = h(m \parallel R)$. Здесь $m \parallel R$ конкатенация двух строк m и R, т.е. объединение их в одну строку путём пристыковки второй строки к первой.

3. Вычислить $S \equiv k + d \cdot E \pmod{P}$.

Пара чисел (S, E) будет электронной подписью под сообщением m.

Сообщение m и подпись (S, E) Алиса отсылает в Банк.

Получив сообщение и подпись Банк должен их проверить. Они могли быть изменены в дороге. Поэтому будем считать, что информация, полученная Банком имеет вид $m_1, (S_1, E_1)$.

Алгоритм 12 (Алгоритм проверки). *1. Вычислить значения*

$$R_1 \equiv G^{S_1} \cdot Y^{-E_1} \pmod{P}) \qquad и \qquad h(m_1 \parallel R_1).$$

2. Если выполняется равенство

$$h(m_1 \parallel R_1) = E_1, \tag{4.1}$$

то подпись считается достоверной и принимается Банком.

Проверим теперь, что если все вычисления были выполнены верно, а сообщение и подпись не были изменены по пути в Банк, т.е. $m_1 = m, S_1 = S, E_1 = E$, то подпись будет принята алгоритмом проверки. Пользуясь тем, что $G^Q \equiv 1 \pmod{P}$, находим

$$G^S \cdot Y^{-E} \pmod{P} \equiv G^{S-dE} \pmod{P} \equiv G^k \pmod{P} = R.$$

Следовательно

$$h(m \parallel G^S \cdot Y^{-E} \pmod{P}) = h(m \parallel R) = E.$$

Равенство (4.1) выполняется.

Заметим, что длина электронной подписи равна 512 битов. За счёт электронной подписи (S, E) пересылаемое сообщение m удлинится на 512 бит.

Описанные идеи могут быть использовать и в других приложениях, например, для подтверждения своей подлинности при обращении к банкомату. В этом случае можно исключить сообщение из алгоритма. Ещё одна вариация может быть использована, если Алиса располагает ограниченными вычислительными возможностями. Например, у неё есть только банковская карточка с небольшим процессором. В этом случае Алиса может поручить трудоёмкую операцию по вычислению хеш-значения из пункта 2 алгоритма подписания Банку, ведь R - открытая информация. Она может быть передана банкомату после помещения банковской карточки в считывающее устройство.

4.1.2. Алгоритм DSA (Digital Signature Algorithm).

Описываемый здесь алгоритм был принят в 1991г. в качестве стандарта электронной подписи в США. Он использует большие простые числа P, Q и образующую G циклической подгруппы порядка Q в мультипликативной группе вычетов целых чисел по модулю P. Предполагается, что эти числа обладают теми же свойствами, что и в предыдущем разделе. Как и ранее, Алиса, желающая подписать сообщение Банку, использует свой секретный ключ d и открытый ключ $Y \equiv G^d \pmod{P}$.

Алгоритм 13 (Алгоритм подписания). *Даны числа P, Q, G с указанными выше свойствами, хеш-функция $h(\cdot)$ и сообщение m.*

1. Вычислить значение хеш-функции $H = h(m)$;

2. выбрать случайное целое число $k, 0 < k < Q$, и определить

$$R \equiv (G^k \pmod P) \pmod Q, \qquad 0 < R < Q;$$

3. найти число S, удовлетворяющее условиям

$$k \cdot S \equiv H + R \cdot d \pmod Q, \qquad 0 < S < Q.$$

Подписью сообщения m служит пара чисел (R, S).

Алиса пересылает эту пару вместе с сообщением m в Банк.

Для проверки подписи Банк, получивший информацию (R_1, S_1) и m_1, делает следующие операции.

Алгоритм 14 (Алгоритм проверки). *1. Вычисляет значение хеш-функции $H_1 = h(m_1)$.*

2. Решая сравнения

$$S_1 \cdot A \equiv H_1 \pmod Q, \quad S_1 \cdot B \equiv R_1 \pmod Q, \quad 0 < A < Q,\ 0 < B < Q,$$

находит числа A, B.

3. Вычисляет

$$V \equiv (G^A \cdot Y^B \pmod P) \pmod Q, \qquad 0 < V < Q.$$

Здесь Y - открытый ключ Алисы.

4. Если выполняется соотношение $V = R_1$, то банк принимает подпись.

Чтобы не загромождать описание излишними для понимания деталями, мы опустили здесь некоторые проверки, например, проверку неравенств $0 < S_1 < Q$, нужных для разрешимости сравнений из пункта 2 алгоритма проверки.

Объясним теперь, почему, если все вычисления выполнены правильно и на пути к Банку отсылаемые числа m, R, S не были изменены, т.е. $m_1 = m, R_1 = R, S_1 = S$, то должно выполняться равенство $V = R_1$.

Пользуясь определением открытого ключа Алисы Y, находим

$$G^A \cdot Y^B \pmod P = G^{A+Bd} \pmod P.$$

Из определения чисел A, B, S следует

$$S(A + Bd) \equiv H + Rd \equiv S \cdot k \pmod Q,$$

так что $A + Bd \equiv k \pmod{Q}$ и далее

$$V \equiv (G^A \cdot Y^B \pmod{P}) \pmod{Q} \equiv (G^{A+Bd} \pmod{P}) \pmod{Q} \equiv (G^k \pmod{P}) \pmod{Q} = R.$$

Заметим, что подпись Алисы зависит от подписываемого сообщения, что создаёт дополнительные сложности при подделке подписи. Кроме того, какое-нибудь изменение сообщения m при его пересылке приведёт к существенному изменению хеш-значения $H = h(m)$ и нарушению равенства $V = R$.

4.2. Электронная подпись с помощью эллиптических кривых.

Первая программируемая ламповая вычислительная машина Colossus, рис.10, была построена в Англии в 1943г. и использовалась для расшифровки сообщений между германским верховным командованием и штабами армий во время Второй мировой войны. Восстановление текстов, зашифрованных с помощью немецкой шифровальной машины Lorenz-SZ, рис.11 шло настолько успешно, что к концу войны в 1945г. этой работой занимались уже 10 машин Colossus II.

С течением времени математические принципы, лежащие в основе методов шифрования и расшифрования, существенно усложнились. В 1985г. появилось предложение использовать эллиптические кривые для сокрытия содержания электронных документов, которые свободно гуляют на просторах Интернета, для защиты их от подделки, подтверждения авторства и т.п. Для любой эллиптической кривой над конечным полем $\mathbb{F}_p$ операция вычисления кратной точки nP по заданной точке P и натуральному числу n может быть выполнена достаточно быстро, даже при больших простых p и больших натуральных n. Для этого можно использовать формулы . Важным для применений в криптографии оказалось то, что обратная задача - нахождение натурального числа n, для которого при заданных точках P, Q выполняется равенство $Q = nP$, очень сложна в вычислительном отношении, т.е. её решение требует очень много вычислительного времени.

Как работают эллиптические кривые в криптографии мы покажем на примере алгоритма, реализующего так называемую цифровую подпись под электронными документами, которая в настоящее время всё больше заменяет подпись руки под документами бумажными. Электронная подпись может понадобиться при регистрации на портале государственных услуг, при покупке товаров или услуг на электронных торговых площадках, при обмене документами с какими-либо организациями и т.д. Во всех случаях документы, подписанные электронной подписью имеют равную юридическую силу с бумажными документами имеющими обычную подпись. С января 2013 года в России введён новый государственный стандарт на электронную подпись, мы его коротко и опишем здесь.

Изобретённая когда-то азбука Морзе записывала каждую букву последовательностью из пяти точек и тире. Вместо этих знаков можно использовать нули и единицы, так что любой текст можно представить в виде последовательности нулей и единиц. Эта последовательность во-

обще говоря будет достаточно длинной. Бумажный документ обычно подписывают на последней странице. Но ведь после подписания можно изменить первые страницы документа, исказив его смысл. Поэтому в важных случаях подписывают каждую страницу документа. Если документ содержит много страниц, это - утомительная операция. Для электронных документов имеются те же проблемы. Их решают с помощью так называемых хэш-функций (перемешивающих функций). Так называют функцию, отображающую последовательности нулей и единиц (сообщения) произвольной длины в такие же последовательности ограниченной длины (например, 160 бит) - значения хэш-функции (хэш-значения). Трудно найти двух человек с одинаковыми отпечатками пальцев. Роль хэш-значения - служить трудно повторимым "отпечатком"исходного сообщения. Для хэш-функции должно быть "очень трудно"найти два различных текста, имеющих один и тот же хэш-образ, лишь в этом случае она может быть полезной в криптографических применениях. Реально используемых хэш-функций немного, они имеют собственные имена и вводятся стандартами. Есть такой государственный стандарт и в России. При желании можно легко найти описания всех таких функций и даже их программные реализации в интернете.

Электронной подписью можно пользоваться и не зная математические алгоритмы, с помощью которых она формируется. Вся премудрость спрятана в компьютерных программах. Нас же будет интересовать математическая сторона дела.

Для того, чтобы вырабатывать электронную подпись, нужно иметь эллиптическую кривую $E : y^2 = x^3 + ax + b$ над конечным полем $\mathbb{F}_p$ и простое число q - делитель N_p - количества точек на кривой E. В российском стандарте предполагается, что $2^{254} < q < 2^{256}$. Простое число p тоже получается достаточно большим. Ведь $N_p \geqslant q$, а согласно теореме Хассе, должно быть

$$(\sqrt{p}+1)^2 = p + 1 + 2\sqrt{p} > N_p \geqslant q.$$

Следовательно $p > (\sqrt{q}-1)^2$. Должна быть известна лежащая на кривой E точка $P \neq O$, удовлетворяющая условию $qP = O$. Должна быть выбрана хэш-функция $h(M)$, которая сообщению M ставит в соответствие некоторую строку $\overline{h}$ из нулей и единиц, т.е. $\overline{h} = h(M)$. В российском стандарте электронной подписи используется хэш-функция, установленная российским стандартом. Для любого сообщения M, она выдаёт последовательность $\overline{h}$ состоящую из 256 нулей и единиц. Обозначим буквой h - целое число, двоичная запись которого совпадает с последовательностью цифр h.

Каждый желающий воспользоваться электронной подписью должен иметь известный только ему ключ подписи $\mathbf{d}$, $0 < \mathbf{d} < q$, и известный всем, кто будет получать подписанные документы ключ проверки $Q = \mathbf{d}P$ - точку на кривой E. Секретный ключ d пользователя является его полной личной собственностью и не передаётся каким-либо другим лицам. Он используется при формировании электронной подписи. Кто угодно может проверить электронную подпись, используя только открытый ключ.

Рассмотрим процесс **формирования подписи.**

1. Пусть M - подписываемый текст, он может включать и дату, когда происходит подписание документа, пусть $\overline{h} = h(M)$ и h - целое число, двоичная запись которого совпадает с $\overline{h}$.
2. Подписывающий текст пользователь выбирает случайно целое число $\mathbf{k}, 0 < \mathbf{k} < q$, и вычисляет точку $C = \mathbf{k}P = (x_C, y_C)$ на кривой E. По существу здесь строится случайная точка C на кривой.
3. Определяются целые неотрицательные числа

$$r \equiv x_C \pmod{q}, \qquad s \equiv (r\mathbf{d} + \mathbf{k}h) \pmod{q} \quad r < q, \quad s < q.$$

4. Пара чисел (r, s) и является электронной подписью сообщения M.

Отметим, что числа, обозначенные в этом алгоритме жирным шрифтом, известны только тому, кто подписывает документ. Обратите, что в следующем далее алгоритме проверки подписи используются только числа, обозначенные нежирным шрифтом. Другими словами, для проверки действительности подписи используется только общеизвестная информация.

Процесс **проверки электронной подписи.**

1. Пусть M - подписанный текст и (r, s) - его электронная подпись. Проверяющий должен вычислить $\overline{h} = h(M)$ и число h, двоичная запись которого совпадает с $\overline{h}$.
2. Затем нужно вычислить такое целое число v, что

$$h \cdot v \equiv 1 \pmod{q} \quad \text{и найти} \quad z_1 \equiv s \cdot v, \quad z_2 \equiv -r \cdot v \pmod{q}.$$

3. Вычислить точку $D = z_1P + z_2Q = (x_D, y_D)$ на кривой E и определить $R = x_D \pmod{q}$.
4. Если $R = r$, подпись принимается.

Слова "подпись принимается"означают, что текст подписанного электронного документа не был изменён и, что подписал его обладатель секретного ключа d, а не кто-нибудь другой: автор подписи не может отказаться от неё.

Проверим, сначала, что если документ и его электронная подпись не были искажены, например, во время путешествия документа по интернету, то должно выполняться равенство $R = r$. Пользуясь обозначениями, введёнными в алгоритмах формирования и проверки электронной подписи, равенствами $qP = qQ = O$ и, выполняя вычисления на кривой E, находим

$$D = z_1P + z_2Q = svP - rvQ = (r\mathbf{d}v + \mathbf{k}hv - rv\mathbf{d})P = \mathbf{k}hvP = \mathbf{k}P = C.$$

Но тогда $R = x_D = x_C = r$. Значит, если числа R и r не равны, с текстом или его подписью что-то неладно.

Отметим, что, зная Q и P, злоумышленник, пользуясь равенством $\mathbf{d}P = Q$, мог бы попытаться найти неизвестное ему $\mathbf{d}$, решая это уравнение. Но размеры параметров p и q столь велики, что для нахождения $\mathbf{d}$ с помощью самых быстрых из известных алгоритмов потребуются тысячелетия непрерывных вычислений.

Точно так же равенство $r = x_C$ позволяет определить точку C с точностью до знака. Но для нахождения неизвестного числа $\mathbf{k}$ из уравнения $C = \mathbf{k}P$ также требуется неприемлемо большое время.

Какие-нибудь эффективные попытки подделать эту электронную подпись в настоящее время не известны.

Лекция 17.

4.3. Схемы обязательств

Чтобы представить себе работу такой схемы, допустим, что Алиса помещает некоторое сообщение в закрытый ящик, запирает его и пересылает в таком виде Бобу. Находящееся внутри ящика сообщение скрыто от Боба, который, получив ящик, не может самостоятельно открыть замок. Но и Алиса, отправив ящик, потеряла доступ к нему и не может изменить его содержимое. Сообщение, отправленное Алисой станет известно Бобу позже, лишь после того, как Алиса передаст или переправит ему ключ от ящика. Иногда более важным бывает не содержание сообщения, а тот факт, что оно осталось неизменным до некоторого момента времени. Чуть ниже мы опишем подобную ситуацию, сейчас же отметим, что взаимодействие в схеме обязательства происходит в два этапа:

• выбранное обязательство в закрытом виде отправляется получателю;

• получателю отправляется ключ, последующее открытие обязательства и его проверка.

4.3.1. Подбрасывание монеты по телефону.

Предположим, Алиса и Боб хотят разрешить спор путем подбрасывания монеты. Если они физически находятся в одном месте, типичная процедура может быть такой:

1. Алиса предсказывает результат бросания монеты.

2. Боб подбрасывает монету.

3. Если предсказание Алисы оказалось правильным, она выигрывает, иначе выигрывает Боб.

Если Алиса и Боб находятся в разных местах, возникает проблема. Как только Алиса предсказала результат бросания монеты, например, по телефону, Боб может назвать тот «результат» броска, который будет ему выгоден. Точно так же, если Алиса объявляет о своем «предсказании» Бобу, после того, как Боб подбросил монету и объявил результат, Алиса может заявить, что она предсказала, тот результат, который выгоден для неё. Но Алиса и Боб могут использовать в этой процедуре

обязательства, которые позволят обоим доверять результату:

1. Алиса предсказывает результат бросания монеты, фиксирует его в обязательстве, и пересылает обязательство Бобу в закрытом виде.

2. Боб подбрасывает монету и сообщает результат Алисе.

3. После этого она раскрывает Бобу своё предсказание и пересылает ключ.

4. Боб проверяет, совпадает ли предсказание Алисы из обязательства с результатом бросания монеты.

5. Если предсказание Алисы совпало с результатом бросания монеты, о котором сообщил Боб, Алиса считается победителем.

Чтобы Боб мог исказить результат в свою пользу, он должен знать предсказание Алисы, скрытое в её обязательстве. Если схема обязательства хорошая, у Боба не должно быть возможности узнать закрытое содержание обязательства. Точно так же Алиса не может повлиять на результат, если она не может изменить значение, которое она фиксирует в обязательстве.

Реальное применение такого подхода возникает, когда люди (часто в средствах массовой информации) сообщают решение или дают ответ в «запечатанном конверте», который открывается позже. Например, на игровом шоу фраза «Давайте теперь выясним, что на это ответил соперник», может возникнуть именно в такой ситуации.

4.3.2. Алгоритмическая реализация.

Как мы собираемся применять схему обязательств. Это можно сделать, используя криптографическую хеш-функцию. Рассмотрим следующую конструкцию: пусть M есть некоторое сообщение, n – случайное одноразовое число, записываемое 256 битами. Чтобы создать обязательство с сообщением M, мы генерируем случайное 256-битовое одноразовое число. Затем объединяем это число и сообщение, а затем вычисляем значение хеш-функции $H(n||M)$[1] от этого объединенного числа. Полученное хеш-значение и будет обязательством.

[1] Символ $n||M$ обозначает строку, в которой начало - n пристыковано слевак сообщению M.

Каждый, кто хочет проверить это (Боб), должен вычислить хэш-значение одноразового числа, которое ему дали (ключ), соединенного с сообщением (от Алисы). Если вычисленное значение совпадёт с закрытым обязательством, то сообщение соответствует этому обязательству.

Из свойств хеш-функции следует выполнимость для обязательств такого рода двух свойств безопасности:

• По обязательству невозможно определить соответствующее ему сообщение.

• Невозможно найти два различных сообщения M_1 и M_2, имеющие одинаковые обязательства.

Рассмотрим ещё один пример. Будем использовать в дальнейшем простые числа P, Q, $Q|(P-1)$ и образующую G циклической подгруппы (G) порядка Q группы $\mathbb{F}_P^*$, описанные ранее в связи с электронной подписью Шнорра. Пусть $B \in (G)$ такой элемент, что его дискретный логарифм по основанию G никому не известен. Определим функцию

$$D_a(x) \equiv B^x G^a \pmod{P}, \tag{4.2}$$

где x, a — целые числа из промежутка от 1 до Q, $1 \leqslant x \leqslant Q$. Число a выбирается случайным образом на своём отрезке и называется "затемняющим", т.к. его задача - скрыть передаваемое число x.

Чтобы раскрыть обязательство отправитель должен сообщить получателю пару чисел (a, x). Получатель не может узнать содержимое x обязательства до тех пор, пока оно не будет раскрыто. Точно так же отправитель обязательства не может изменить его значение. В действительности, чтобы узнать информацию или изменить её, оба они должны использовать неприемлемо большое количество вычислительных ресурсов (очень большие вычислительные мощности и время вычислений). В основе такой уверенности лежит сложность решения задачи дискретного логарифмирования.

Докажем некоторое тождество для функции $D_a(x)$, оно пригодится в дальнейшем,а именно:

$$D_{a_1}(x_1) \cdot D_{a_2}(x_2) \equiv B^{x_1} G^{a_1} B^{x_2} G^{a_2} = B^{x_1+x_2} G^{a_1+a_2} \equiv D_{a_3}(x_3) \pmod{P}, \tag{4.3}$$

где $x_1 + x_2 \equiv x_3 \pmod{Q}$, $a_1 + a_2 \equiv a_3 \pmod{Q}$, $1 \leqslant x_3 \leqslant Q$, $1 \leqslant a_3 \leqslant Q$.

4.4. Подтверждение выбора (Доказательство с нулевым разглашением).

Чтобы представить себе, с какой целью используются доказательства с нулевым разглашением, рассмотрим, например, ситуацию, когда Алиса должна выбрать одно из двух чисел, скажем 1 или -1, для дальнейшего использования, а Боб хотел бы получить уверенность в том, что Алиса его не обманула и не выбрала какое-нибудь третье число. Алиса должна подтвердить Бобу правильность своего выбора, но так, чтобы не раскрыть выбранное ею число.

Идея состоит в том, чтобы в результате обмена некоторой информацией Алиса и Боб смогли составить систему уравнений, зависящую от выбора Алисы, для которой она, если выбрала число 1 или -1, легко могла бы предъявить решение. В случае же выбора ею третьего числа, решение системы потребовало бы неприемлемо большого времени. Для описания соответствующего протокола обмена информацией воспользуемся простыми числами P, Q, $Q|(P-1)$, элементом G порядка Q в мультипликативной группе вычетов по модулю P и элементом B подгруппы (G), порядок которого никому не известен. Необходимые свойства этих чисел уже указывались нами в предыдущем разделе, а также в разделе, посвященном электронной подписи Шнорра. Будем пользоваться определённой в том же разделе функцией $D_a(x) \equiv B^x G^a \pmod P$ и в дальнейшем обозначать буквой x выбранное Алисой число.

1. Алиса выбирает случайное секретное число $a, 1 < a < Q$, вычисляет и публикует число $D_a(x)$. После этого изменить число x она уже не может. Затем она выбирает случайным способом ещё три числа d, r, w, все из промежутка от 1 до Q, вычисляет

$$\begin{cases} A_1 = G^r (D_a(x) B)^{-d}, & A_2 = G^w, \text{ если } x = 1, \\ A_2 = G^r (D_a(x) B^{-1})^{-d}, & A_1 = G^w, \text{ если } x = -1. \end{cases} \tag{4.4}$$

и публикует числа A_1 и A_2, соответствующие выбранному ей числу x.

Теперь каждый может написать систему уравнений в поле $\mathbb{F}_P$

$$G^{R_1} = A_1 (D_a(x) B)^{D_1},$$
$$G^{R_2} = A_2 (D_a(x) B^{-1})^{D_2}$$

относительно неизвестных (D_1, D_2, R_1, R_2).

2. Боб выбирает случайное число C или, например, вычисляет его как значение хеш-функции на сообщении $D_a(x)||A_1||A_2$ и высылает его Алисе.

3. Алиса должна найти решение системы уравнений

$$\left\{\begin{array}{rcl} G^{R_1} & = & A_1(D_a(x)B)^{D_1}, \\ G^{R_2} & = & A_2(D_a(x)B^{-1})^{D_2}, \\ C & = & D_1 + D_2. \end{array}\right. \tag{4.5}$$

При $x = 1$ решение составляют числа

$$D_1 = d, \quad D_2 = C - d, \quad R_1 = r, \quad R_2 = w + a(C - d). \tag{4.6}$$

Если же $x = -1$ то решением будет набор

$$D_1 = C - d, \quad D_2 = d, \quad R_1 = w + a(C - d), \quad R_2 = r. \tag{4.7}$$

Найденное решение Алиса отправляет Бобу.

4. Боб, получив от Алисы 4 числа, подставляет их в уравнения системы и убеждается, что полученные числа действительно составляют её решение. Найдя решение системы, Алиса подтвердила, что выбранное ею число x есть 1 или -1.

Проверим теперь, что числа (4.6), (4.7) будут решениями системы уравнений (4.5) при указанных значениях x.

При любом $x = \pm 1$ согласно (4.6) и (4.7) имеем $D_1 + D_2 = C$, так что в обоих случаях последнее уравнение системы (4.5) удовлетворяется.

Если $x = 1$, то, пользуясь равенствами (4.4) и (4.6), находим

$$A_1(D_a(x)B)^{D_1} = G^r(D_a(x)B)^{-d}(D_a(x)B)^d = G^r = G^{R_1}.$$

и

$$A_2(D_a(x)B^{-1})^{D_2} = G^w(D_a(x)B^{-1})^{C-d} = G^{w+a(C-d)} = G^{R_2}.$$

Итак, при $x = 1$ числа (4.6) действительно составляют решение системы (4.5). При вычислениях использовалось равенство $D_a(1) = B \cdot G^a$, см. (4.2).

Если $x = -1$, то, пользуясь равенствами (4.4) и (4.7), находим

$$A_1(D_a(x)B)^{D_1} = G^w(D_a(-1)B)^{C-d} = G^{w+a(C-d)} = G^{R_1},$$

и

$$A_2(D_a(x)B^{-1})^{D_2} = G^r(D_a(x)B^{-1})^{-d}(D_a(x)B^{-1})^d = G^r = G^{R_2}.$$

Итак, при $x = -1$ числа (4.7) действительно составляют решение системы (4.5). При вычислениях использовалось также равенство $D_a(-1) = B^{-1} \cdot G^a$, см. (4.2).

Число $D_a(x)$ опубликовано Алисой, числа G и B также известны и ей, и Бобу. Числа A_1, A_2 Алиса определяет до того, как Боб сообщает

ей число C. В этих условиях, как бы Алиса ни выбирала параметры и переменные, в конечном счёте выбор сводится к решению трудоёмкой задачи дискретного логарифмирования. Если простые числа P, Q достаточно велики, эта задача решается за неприемлемо большое время. Итак, если Алиса выбрала число, отличное от ± 1, ей для решения системы понадобится недопустимо большое время.

Кроме того, описанный протокол не даёт Бобу никакой информации относительно выбранного Алисой значения x.

Конец двенадцатой лекции.