

# Лекции по математической криптографии

Н.К. Верещагин

22 апреля 2014 г.



# Оглавление

<b>1</b>	<b>Введение</b>	<b>7</b>
<b>2</b>	<b>Необратимые и односторонние функции</b>	<b>9</b>
2.1	Определения . . . . .	9
2.2	Другой вариант определения . . . . .	11
2.3	Задачи . . . . .	13
2.4	Примеры . . . . .	15
2.5	Усиление . . . . .	16
2.6	Необратимые частичные функции . . . . .	21
2.7	Дополнительные требования к частичным односторонним функциям . . . . .	25
<b>3</b>	<b>Генераторы псевдослучайных чисел</b>	<b>31</b>
3.1	Вычислительно неотличимые случайные величины . . . . .	31
3.2	Определение генератора ПСЧ . . . . .	34
3.3	Генераторы ПСЧ и односторонние функции . . . . .	36
3.4	Трудный бит . . . . .	36
3.5	Отступление: надежность генераторов по Яо . . . . .	43
3.6	Построение функции, являющейся трудным битом . . . . .	45
3.6.1	Код Адамара . . . . .	45
3.6.2	Конструкция . . . . .	49
3.6.3	Трудный бит для односторонней перестановки с секретом . . . . .	52
<b>4</b>	<b>Псевдослучайные функции</b>	<b>53</b>
<b>5</b>	<b>Шифрование с закрытым ключом</b>	<b>61</b>
5.1	Одноразовые схемы . . . . .	61

5.2	Многоразовые схемы . . . . .	64
5.3	Более экономная многоразовая СШЗК . . . . .	66
<b>6</b>	<b>Шифрование с открытым ключом</b>	<b>67</b>
6.1	Определение схемы шифрования с открытым ключом . . . .	67
6.2	Построение схемы шифрования с открытым ключом . . . .	69
<b>7</b>	<b>Протоколы привязки к биту (bit commitment)</b>	<b>71</b>
<b>8</b>	<b>Интерактивные протоколы привязки</b>	<b>75</b>
8.1	Протоколы привязки к биту . . . . .	75
8.2	Протоколы привязки к строке . . . . .	79
<b>9</b>	<b>Протоколы бросания монетки</b>	<b>81</b>
<b>10</b>	<b>Неразглашение информации</b>	<b>87</b>
10.1	Неразглашение информации при последовательном выпол- нении алгоритмов . . . . .	92
10.2	Защищенность схем шифрования относительно атаки с вы- бором сообщений . . . . .	97
<b>11</b>	<b>Протоколы идентификации</b>	<b>99</b>
11.1	Протоколы идентификации с закрытым ключом . . . . .	99
11.2	Протоколы идентификации с открытым ключом . . . . .	101
11.3	Конструкция на основе любой односторонней функции . . .	107
<b>12</b>	<b>Доказательства знания с нулевым разглашением</b>	<b>109</b>
12.1	Доказательство теоремы 25 . . . . .	111
<b>13</b>	<b>Семейства хэш-функций</b>	<b>119</b>
13.1	Семейства хэш-функций с трудно обнаружимыми колли- зиями . . . . .	119
13.2	Универсальные семейства односторонних хэш-функций . .	122
<b>14</b>	<b>Протоколы электронной подписи с закрытым ключом</b>	<b>127</b>
<b>15</b>	<b>Протоколы электронной подписи с открытым ключом</b>	<b>131</b>
15.1	Одноразовая схема подписи одного бита . . . . .	133
15.2	Подпись одного сообщения фиксированной длины . . . . .	135

15.3	Протокол подписи одного сообщения произвольной длины .	136
15.4	Построение схемы подписи одного сообщения произвольной длины, исходя из семейств хэш-функций с трудно обнаружимыми коллизиями . . . . .	137
15.5	Построение схемы подписи одного сообщения произвольной длины, исходя из универсальных семейств односторонних хэш-функций . . . . .	139
15.6	Общий случай: подпись произвольного количества сообщений произвольной длины . . . . .	142
<b>16</b>	<b>Конфиденциальные вычисления</b>	<b>149</b>
16.1	Забывающая передача (oblivious transfer) . . . . .	150
<b>17</b>	<b>Интерактивные алгоритмы</b>	<b>153</b>
17.1	Определение . . . . .	153
17.2	Повторение интерактивного протокола . . . . .	155



# Глава 1

## Введение

В лекциях постоянно используется понятие детерминированных полиномиальных алгоритмов (машин Тьюринга, время работы которых ограничено некоторым многочленом от длины исходного данного), вероятностных полиномиальных алгоритмов (вероятностных машин Тьюринга, время работы которых ограничено некоторым многочленом от длины исходного данного, независимо от результатов сделанных ими случайных выборов) и схем из функциональных элементов. О том, что это такое, можно прочитать в книге [1].





## Глава 2

# Необратимые и односторонние функции

### 2.1 Определения

Пусть имеется семейство функций  $f_n: \mathbb{B}^{k(n)} \rightarrow \mathbb{B}^{l(n)}$ , где  $k, l$  некоторые полиномы.

Мы хотим определить понятие “плохо обратимой” функции. Обратить  $f(x)$  означает найти какое-то  $x'$  такое, что  $f(x') = f(x)$ . Возможны разные варианты определения: слабый и сильный. Неформально говоря, обращение слабо необратимой функции оказывается неверным с заметной (не пренебрежимо малой) вероятностью; для сильно необратимой функции оно неверно почти всегда (вероятность удачного обращения пренебрежимо мала). В качестве процедур обращения мы будем рассматривать полиномиальные вероятностные алгоритмы, получающие на вход  $n$  единиц и  $f_n(x)$ .

Дадим точные определения.

Семейство функций  $f_n$  описанного вида называется *слабо необратимым*, если существует некоторый многочлен  $p$  с таким свойством: для любого полиномиального вероятностного алгоритма  $R$  вероятность события

$$f_n(R(1^n, f_n(x))) = f_n(x),$$

(где все слова  $x$  длины  $k(n)$  считаются равновероятными) не превосходит

$$1 - \frac{1}{p(n)}$$

для всех достаточно больших  $n$ . Если  $f_n(R(1^n, f_n(x))) = f_n(x)$ , то мы говорим, что схема  $R$  успешно обращает  $f_n(x)$ .

В этом определении на вход алгоритма  $R$  (гипотетического алгоритма обращения) дается слово длины  $l(n)$ , и он должна найти слово длины  $k(n)$ , попадающее в  $f_n$ -прообраз входного слова. Такой прообраз существует, так как обращению подлежит слово вида  $f_n(x)$  для некоторого (неизвестного алгоритму) слова  $x$ .

Важно, что мы не требуем точного обращения (равенства  $R(1^n, f_n(x)) = x$ ). Если бы требовали, то любая функция, склеивающая все слова в одно, оказалась бы необратимой.) Ещё отметим, что равномерное распределение берётся на словах  $x$ , а не на их образах (иначе трудность обращения могла бы быть вызвана тем, что многие слова не имеют прообраза).

Наконец, заметим, что многочлен  $p$ , задающий долю неудачных попыток обращения, не должен зависеть от последовательности схем  $C_n$ . Если оказывается, что с ростом сложности схем (в пределах полиномиальной) обращение становится возможным всё лучше и лучше, и ошибка может быть сделана меньше любого наперёд заданного полинома, то это не годится.

Семейство функций  $f_n$  называется *сильно необратимым*, если для любого полиномиального вероятностного алгоритма  $R$  вероятность события

$$f_n(R(1^n, f_n(x))) = f_n(x),$$

(где все слова  $x$  длины  $k(n)$  считаются равновероятными) стремится к нулю при  $n \rightarrow \infty$  быстрее любого обратного полинома. Это означает, что для любого многочлена  $q$  она не превосходит

$$\frac{1}{q(n)}$$

для всех достаточно больших  $n$ . (Разумеется, та граница, с которой начинаются “достаточно большие”  $n$ , может зависеть от  $q$ .)

Семейство  $f_n$  называется полиномиально вычислимым, если имеется алгоритм, получающий на вход  $n$  и слово  $x$  длины  $k(n)$  и вычисляющий за полиномиальное от  $n$  время слово  $f_n(x)$ . Сильно необратимые полиномиально вычисляемые семейства функций называются *сильно односторонними*. Аналогично, слабо необратимые семейства, вычисляемые за полиномиальное время называются *слабо односторонними*.

В дальнейшем мы будем позволять себе следующую терминологическую вольность: мы будем говорить “сильно необратимая функция”

вместо “сильно необратимое семейство функций” и “слабо односторонняя функция” вместо “слабо одностороннее семейство функций”. Когда говорят просто “односторонняя функция”, имеют в виду сильно одностороннюю функцию.

## 2.2 Другой вариант определения: односторонние функции для неравномерного противника

Как известно, любой полиномиальный алгоритм можно моделировать последовательностями схем из функциональных элементов полиномиального размера. Точнее, пусть  $A$  — произвольный полиномиальный алгоритм, получающий на вход и выдающих на выход двоичные слова, для которого длина выходного слова определяется только длиной входного слова. Тогда для каждого натурального  $n$  существует схема  $C_n$  из функциональных элементов, размер которой ограничен некоторым многочленом от  $n$ , такая, что  $C_n(x) = A(x)$  для всех двоичных слов  $x$  длины  $n$ .

Вероятностные полиномиальные алгоритмы можно моделировать вероятностными схемами из функциональных элементов полиномиального размера. Точнее, назовем вероятностной схемой любую схему  $C$  из функциональных элементов, входы которой разделены на две группы,  $x_1, \dots, x_n$  и  $r_1, \dots, r_s$ , называемых основными и случайными входами. Для такой схемы мы будем обозначать через  $C(x_1, \dots, x_n)$  случайную величину, для которой при каждой последовательности битов  $x_1, \dots, x_n$  на основном входе вероятность события  $C(x_1, \dots, x_n) = y$  равна доле слов  $r_1, \dots, r_s$ , для которых  $C(x_1, \dots, x_n, r_1, \dots, r_s) = y$ .

Для любого вероятностного полиномиального алгоритма  $A$  существует последовательность вероятностных схем  $C_n$  полиномиального размера, для которой для всех  $n$  и всех двоичных слов  $x$  длины  $n$  случайные величины  $A(x)$  и  $C_n(x)$  имеют одно и то же распределение. В самом деле, пусть алгоритм  $A(x)$  на входах длины  $n$  делает  $s(n)$  бросаний монетки, где  $s$  — некоторый многочлен. Тогда  $A$  можно воспринимать, как детерминированный полиномиальный алгоритм с дополнительным входом  $r$  длины  $s(n)$ , где  $n$  обозначает длину основного входа. Построим последовательность схем  $C_n$  полиномиального размера, моделирующую работу

этого детерминированного алгоритма, то есть  $C_n(x, r) = A(x, r)$  для всех  $x$  длины  $n$  и всех  $r$  длины  $s(n)$ . Схема  $C_n$  и является искомой.

Теперь разрешим в определении сильно и слабо необратимой функции использовать последовательности вероятностных схемы полиномиального размера. Этим мы расширим класс алгоритмов обращения и тем самым получим более сильные определения (сильного и слабого) одно-стороннего семейств функций. Более точно, мы говорим, что семейство функций  $f_n$  называется *слабо необратимым*, если существует некоторый многочлен  $p$  с таким свойством: для любой последовательности вероятностных схем  $C_n$  из функциональных элементов, размер которых ограничен некоторым полиномом от  $n$  вероятность события

$$f_n(C_n(f_n(x))) = f_n(x),$$

(где все слова  $x$  длины  $k(n)$  считаются равновероятными) не превосходит

$$1 - \frac{1}{p(n)}$$

для всех достаточно больших  $n$ .

Семейство функций  $f_n$  называется *сильно необратимым*, если для любой последовательности вероятностных схем  $C_n$  из функциональных элементов, размер которых ограничен некоторым полиномом от  $n$ , вероятность события

$$f_n(C_n(f_n(x))) = f_n(x),$$

(где все слова  $x$  длины  $k(n)$  считаются равновероятными) стремится к нулю при  $n \rightarrow \infty$  быстрее любого обратного полинома. Это означает, что для любого многочлена  $q$  она не превосходит

$$\frac{1}{q(n)}$$

для всех достаточно больших  $n$ .

Нетрудно заметить, что в обоих определениях без ограничения общности можно ограничиться детерминированными схемами из функциональных элементов. В самом деле, пусть  $C_n$  вероятностная схема из функциональных элементов полиномиального размера. Вероятность события  $f_n(C_n(f_n(x))) = f_n(x)$  (обозначим её через  $p_n$ ) есть среднее по всем случайным входам  $r$  схемы  $C$  вероятности события  $f_n(C_n(f_n(x), r)) =$

$f_n(x)$ . Поэтому для каждого  $n$  существует некоторое  $r = r_n$ , для которого последняя вероятность не меньше  $p_n$ . Задав это значение  $r$  в вероятностную схему, мы получим детерминированную схему, для которой вероятность успешного обращения будет не меньше  $p_n$ . Полученная детерминированная схема имеет не больший размер, чем исходная, и вероятность успеха у нее не меньше.

Чтобы отличать два определения односторонней функции (определение из предыдущего и этого разделов), мы будем в дальнейшем говорить об односторонних функциях *для равномерного* или *неравномерного* противника. Равномерный противник обозначает вероятностные полиномиальные алгоритмы, а неравномерный — последовательности схем (вероятностных или детерминированных) полиномиального размера.

## 2.3 Задачи

*Задача 1.* Докажите, что любая сильно необратимая функция является слабо необратимой. Это верно как для равномерного, так и для неравномерного противника.

*Задача 2.* Пусть  $f_n$  есть возведение в квадрат по модулю  $2^n$  на  $n$ -битовых натуральных числах. Докажите, что  $f_n$  не является слабо необратимым семейством функций даже для равномерного противника.

*Задача 3.* Докажите, что для неравномерного противника сильно необратимые функции существуют (не обязательно полиномиально вычисляемые).

*Задача 4.* Докажите, что и для неравномерного противника сильно необратимые функции существуют.

*Задача 5.* Пусть  $f_n : \{0, 1\}^n \rightarrow \{0, 1\}^n$  выбирается случайно среди всех функций (не обязательно биективных). Все функции считаются равновероятными и  $f_n$  независимы при разных  $n$ . Докажите, что с вероятностью 1 семейство  $f_n$  сильно необратимо (для неравномерного противника).

*Задача 6.* Пусть даны две биективные полиномиально вычисляемые функции  $f_n, g_n : \{0, 1\}^n \rightarrow \{0, 1\}^n$ . Докажите, что если хотя бы одна из них сильно [слабо] необратима, то и их композиция сильно [слабо] необратима. Это верно как для равномерного, так и для неравномерного противника.

*Задача 7.* Докажите, что существуют сильно необратимые функции  $f_n, g_n : \{0, 1\}^n \rightarrow \{0, 1\}^n$ , композиция которых не является даже слабо необратимой (полиномиальная вычислимость функций не требуется). Это верно как для равномерного, так и для неравномерного противника.

*Задача 8.* Пусть множество значений функции  $f_n$  имеет не более  $\text{poly}(n)$  элементов. Докажите, что тогда  $f_n$  не является слабо необратимым семейством функций. Это верно как для равномерного, так и для неравномерного противника, но второе значительно проще первого.

*Задача 9.* Докажите, что существуют слабо необратимые функции, не являющиеся сильно необратимыми. Это верно как для равномерного, так и для неравномерного противника.

*Задача 10.* Известно, что функции  $f_n$  и  $g_n$  связаны следующим соотношением:  $g_n(x) = (f_n(x) \text{ без последнего бита})$ . (а) Докажите, что если функция  $g_n$  необратима, то и функция  $f_n$  необратима. (б) Докажите, что обратное может быть неверно (для некоторых  $f_n, g_n$ ). Это верно как для равномерного, так и для неравномерного противника.

*Задача 11.* Докажите, что существуют функции  $f_n, g_n$ , не являющиеся слабо необратимыми и такие, что функция  $x \mapsto f_n(x)g_n(x)$  сильно необратима. Это верно как для равномерного, так и для неравномерного противника.

*Задача 12.* Пусть даны сильно [слабо] необратимая функция  $f_n : \{0, 1\}^n \rightarrow \{0, 1\}^n$ . Докажите, что тогда функция  $xu \mapsto f_n(x)u$  определенная на словах длины  $2n$  сильно [слабо] необратима. Это верно как для равномерного, так и для неравномерного противника.

*Задача 13.* Докажите, что существуют сильно необратимые функции  $f_n, g_n$  такие, что функция  $x \mapsto f_n(x)g_n(x)$  не является даже слабо необратимой. Это верно как для равномерного, так и для неравномерного противника.

*Задача 14.* Докажите, что если функция  $f_n$  сильно [слабо] необратима, то и функция  $x \mapsto f_n(x)f_n(x)$  сильно [слабо] необратима. Это верно как для равномерного, так и для неравномерного противника.

*Задача 15.* Докажите, что определение сильно необратимой функции для неравномерного противника можно переформулировать следующим образом: для любого полинома  $q$  существует последовательность чисел  $\varepsilon_n$ , стремящаяся к нулю быстрее любого обратного полинома от  $n$  и такая,

что лишь для конечного числа  $n$  существует схема размера не более  $q(n)$ , успешно обращающая  $f_n(x)$  с вероятностью более  $\varepsilon_n$ . Переформулируйте в аналогичном виде определение слабо необратимой функции.

## 2.4 Примеры

Неизвестно, существуют ли односторонние или хотя бы слабо односторонние функции (даже для неравномерного противника). Доказать их существование сложно, поскольку из него следует, что  $P=NP$ . (О том, что такое  $P$  и  $NP$  можно прочитать в книге [1].)

**Теорема 1.** *Если  $P=NP$ , то любое полиномиально вычисляемое семейство функций  $f_n$  не является слабо обратимым даже для равномерного противника. Более того, существует детерминированный алгоритм, который для всех  $x$  по  $n$  и  $f_n(x)$  за полиномиальное от  $n$  время находит некоторый прообраз  $f(x)$  длины  $k(n)$ .*

*Доказательство.* Рассмотрим следующий язык  $L$  из  $NP$ . Он состоит из всех троек слов  $(1^n, y, z)$ , таких, что длина  $z$  равна  $n$  и найдется слово  $x$  длины  $k(n)$ , начинающееся на  $z$ , для которого  $f_n(x) = y$ . Поскольку мы предположили, что  $P=NP$ , этот язык можно разрешить за полиномиальное от  $n$  количество шагов. Пусть нам даны  $n$  и слово  $y$  длины  $n$ . Сначала выясняем, принадлежит ли  $L$  тройка  $(1^n, y, \Lambda)$ , где  $\Lambda$  обозначает пустое слово. Если нет, то  $y$  не имеет прообраза. Иначе запускаем бинарный поиск прообраза  $y$ . А именно, сначала выясняем принадлежит ли  $L$  тройка  $(1^n, y, 0)$ . Если да, то у  $y$  есть прообраз, начинающийся с нуля, а иначе — с единицы. И так далее. После  $i$  шагов у нас имеется строка  $z$  длины  $i$ , для которой мы знаем, что  $y$  имеет прообраз, начинающийся на  $z$ . После  $k(n)$  шагов мы найдем некоторый прообраз целиком.  $\square$

Приведём два семейства функций, которые возможно являются сильно односторонними.

*Произведение натуральных чисел:*  $f_n(x)$  равно произведению первой и второй половинок  $x$ , рассматриваемых как двоичные записи натуральных чисел. Здесь  $k(n) = l(n) = 2n$  (произведение двух  $n$ -битовых чисел содержит не более  $2n$  битов).

*Функция SUBSET-SUM.* Определение этой функции навеяно  $NP$ -полной задачей о сумме подмножеств (SUBSET-SUM). Здесь  $k(n) = n^2 + n$ ,

$l(n) = n^2 + 2n + \lceil \log n \rceil$ . Значение  $f_n$  на словах  $x$  длины  $n^2 + n$  определяется так. Разрежем  $x$  на  $n + 1$  блоков длины  $n$  и будем понимать первые  $n$  блоков как  $n$  натуральных чисел  $x_1, \dots, x_n$  в двоичной записи. Последний блок будем понимать как подмножество  $I$  множества  $\{1, 2, \dots, n\}$ . По определению  $f_n(x)$  равно конкатенации  $x_1, \dots, x_n$  и  $\sum_{i \in I} x_i$ .

Гипотеза о их существовании сильно односторонних семейств лежит в основе почти всех криптографических протоколов. Причём достаточно предположить существование лишь слабо одностороннего семейства — из любого такого семейства можно получить сильно одностороннее семейство.

## 2.5 Преобразование слабо одностороннего семейства в сильно одностороннее

**Теорема 2.** *Если существуют слабо односторонние функции, то существуют и сильно односторонние функции. Это верно как для равномерного, так и для неравномерного противника.*

(Заметим, что в этом случае существуют и слабо односторонние функции, не являющиеся сильно односторонними, см. задачу 9.)

**Доказательство.** Проведем доказательство для равномерного противника (для неравномерного противника оно совершенно аналогично). Пусть  $f_n$  — слабо одностороннее семейство и  $p(n)$  такой полином, что вероятность успешного обращения  $f_n(x)$  вероятностными полиномиальными алгоритмами при достаточно больших  $n$  не превосходит  $1 - 1/p(n)$ . Для краткости мы будем опускать индекс  $n$  и говорить об одной функции. (Имеется в виду, что это построение и все рассуждения выполняются параллельно для всех  $n$ .) Рассмотрим новую функцию  $f^N$ , для которой

$$f^N(x_1 x_2 \dots x_N) = f(x_1) f(x_2) \dots f(x_N).$$

Она определена на словах длины  $Nk(n)$ , которые отождествляются с кортежами из  $N$  слов длины  $k(n)$ , и принимает значения длины  $Nl(n)$ . В качестве  $N$  мы возьмем некоторый достаточно большой многочлен от  $n$  (какой — скажем дальше).

Начнём с простого (но, увы, неправильного) объяснения, почему при достаточно большом  $N$  функция  $f^N$  является сильно односторонней.



В самом деле, обращение  $f^N(x_1 \dots x_N)$  для случайного  $x_1 \dots x_N$  требует одновременного обращения  $f(x_1), \dots, f(x_N)$  при независимых случайных  $x_1, \dots, x_N$ . В каждом из  $N$  случаев вероятность ошибки не меньше  $1/p(n)$ . Поэтому общая вероятность успеха не больше

$$\left(1 - \frac{1}{p(n)}\right)^N.$$

Если положить  $N = np(n)$ , то выражение это равно  $(1 - 1/p(n))^{p(n)}$  в степени  $n$ , что экспоненциально убывает (примерно равно  $e^{-n}$ ). Поэтому функция  $f^N$  сильно односторонняя. Забегая вперед, отметим, что хотя это рассуждение и ошибочное, выбранное значение  $N$  окажется правильным.

## Исправление

Что неверно в этом рассуждении? Дело в том, что мы рассматриваем вероятность успешного обращения  $f^N$  с помощью алгоритма специального вида, состоящего в том, что некий алгоритм обращения функции  $f$  применяется параллельно для всех  $n$ . А что делать, если алгоритм обращения такого вида не имеет? Кажется странным, что какие-то алгоритмы обращения функции  $f^N$  могут делать что-то более сложное, чем обращать в отдельности каждое  $f(x_i)$ , но это не доказательство (и непонятно, как это можно было бы формализовать буквально).

Правильное доказательство должно идти в другом направлении: вместо того, чтобы начинать с алгоритма обращения для  $f$  и затем смотреть, что получится при его раздельном применении к  $f(x_i)$ , мы должны рассмотреть произвольный алгоритм  $R_{f^N}$ , претендующий на обращение функции  $f^N$ , и показать, что если он имеет непренебрежимые шансы на успех, то для функции  $f$  есть алгоритм  $R_f$ , у которого вероятность ошибки обращения меньше  $1/p(n)$ .

Попробуем следующий (ещё не окончательный) вероятностный алгоритм обращения  $f(x)$ . Пусть нам дано некоторое слово  $y$ , прообраз которого мы ищем. Мысленно представим себе, что  $y$  есть  $f(x_1)$  для неизвестного  $x_1$  и дополним слово  $y$  словами  $f(x_2), \dots, f(x_N)$  для случайных слов  $x_2, \dots, x_N$ . Применим к полученному слову

$$yf(x_2) \dots f(x_N)$$

алгоритм  $R_{f^N}$  и посмотрим, что он даст на первом месте (каковы первые  $k(n)$  битов). Это слово и будет результатом работы алгоритма.

Нетрудно понять, что вероятность успеха алгоритма  $R_f$  (для  $f(x_1)$  при случайном  $x_1$ ) не меньше вероятности успешного обращения  $f(x_1) \dots f(x_N)$  алгоритмом  $R_{f^N}$  для случайных  $x_1 \dots x_N$ . Этого нам мало, поскольку надо перейти от не очень малой вероятности к вероятности, близкой к единице.

Как можно усовершенствовать алгоритм  $R_f$ ? Сразу возникают две идеи. Во-первых, можно производить несколько попыток обращения подряд (и выбирать из них удачную, если таковая случилась). Ведь сама функция  $f$  полиномиально вычислима, поэтому мы можем за полиномиальное время проверить, удалась ли попытка обращения. Другая идея: мы можем ставить обрабатываемое слово  $y$  не только на первое место, но и на любое другое место от 1 до  $N$ .

### Алгоритм обращения

Этих двух идей окажется достаточно для доказательства теоремы. А именно, для данного алгоритма  $R_{f^N}$ , претендующего на обращение  $f^N$ , мы рассмотрим вероятностный алгоритм для обращения  $f(x)$ , действующий следующим образом.

Для каждого  $i$  от 1 до  $N$  поставим обрабатываемое слово  $y$  на  $i$ -е место и окружим его  $N - 1$  словами, получив набор

$$f(x_1), \dots, f(x_{i-1}), y, f(x_{i+1}), \dots, f(x_N).$$

Слова  $x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_N$  выбираются независимо и равномерно распределены в  $\mathbb{B}^{k(n)}$ . К полученному набору применяется  $R_{f^N}$ . Если  $i$ -я компонента результата лежит в прообразе  $y$ , констатируем удачу (и сворачиваем всё дальнейшее).

Эти действия ( $N$  попыток, по одной для каждого  $i$ ) составляют один этап алгоритма  $R_f$ . Полностью  $R_f$  состоит в повторении этого действия несколько раз. Обозначим число повторений через  $M$  (и выберем его дальше).

[Заметим сразу, что ставить  $y$  на разные позиции необходимо. В самом деле, если  $R_{f^N}$  почему-то не работает на образах слов, в которых первый бит (первого слова) равен нулю, то он вполне может иметь вероятность успеха  $1/2$ . Но тогда и алгоритм  $R_f$  (если  $y$  ставится лишь на первое

место) не будет работать на образах таких слов, и вероятность успеха не подымется выше  $1/2$ , сколько повторений ни делай.]

Теперь нам нужно подобрать значение  $M$  таким образом, чтобы из того, что вероятность успеха  $R_f$  меньше  $1 - 1/p(n)$ , следовало бы, что вероятность успеха  $R_{f^N}$  меньше  $1/q(n)$ . (Число  $M$  будет некоторым полиномом от  $n$ , причём тем бóльшим, чем больше  $q(n)$ .) Для этого необходимо разобраться, как связаны вероятности успеха для алгоритмов  $R_{f^N}$  и  $R_f$ . Обозначим через  $s_1(x_1)$  вероятность того, алгоритм  $R_{f^N}$  правильно обращает слово

$$f(x_1)f(x_2)\dots f(x_N)$$

для случайно выбранных  $x_2, \dots, x_N$ . Вероятность успешного обращения для первой попытки (когда мы ставим  $y$  на первое место) для слова  $f(x_1)$  не меньше  $s_1(x_1)$ . Аналогичные вероятности можно рассмотреть и для других попыток (всего в каждом этапе  $N$  попыток). По аналогии обозначим их  $s_2(x_2), \dots, s_N(x_N)$ .

Какова вероятность успеха на одном этапе? Для входа  $f(x)$  она заведомо не меньше

$$\max(s_1(x), \dots, s_N(x)),$$

поскольку успех одной попытки означает успех всего этапа. Обозначим этот максимум через  $\hat{s}(x)$ .

Дальнейший ход доказательства можно описать так. Посмотрим, сколько имеется слов  $x$ , для которых величина  $\hat{s}(x)$  очень мала (меньше некоторой границы  $\varepsilon$ , которую мы выберем впоследствии). Такие слова, неформально говоря, трудны для обращения. Нетрудные слова хорошо обращаются алгоритмом  $R_f$ , поскольку на каждом этапе вероятность обращения не меньше  $\varepsilon$ , этапы независимы и их много. Пользуясь тем, что вероятность неудачи  $R_f$  больше  $1/p(n)$  мы докажем, что трудных слов много. Отсюда будет следовать, что  $R_{f^N}$  обращает  $f^N$  с вероятностью меньше  $1/q(n)$ . Эти неформальные разговоры следует, конечно, уточнить.

## Оценки

Вероятность успеха алгоритмов  $R_f$  и  $R_{f^N}$  можно связать следующим образом. Пусть  $\varepsilon$  — некоторое положительное число, порядка  $1/\text{poly}(n)$ . Будем называть слово  $x$  “трудным”, если  $\hat{s}(x) < \varepsilon$ . Обозначим через  $\delta$  долю трудных слов среди всех слов длины  $n$ .

**Лемма.** (1) Вероятность ошибки  $R_f$  при обращении слова  $f(x)$  (для случайного  $x$ ) меньше

$$\delta + (1 - \varepsilon)^M$$

(2) Вероятность успеха  $R_{f^N}$  при обращении слова  $f^N(x_1 \dots x_N)$  (для случайных независимых  $x_1, \dots, x_N$ ) меньше

$$N\delta\varepsilon + (1 - \delta)^N.$$

При подходящем выборе параметров  $\varepsilon, M$  эта лемма позволит нам ограничить сверху вероятность успеха  $R_{f^N}$ , зная, что вероятность ошибки  $R_f$  не меньше  $1/p(n)$ .

*Доказательство.* (1) Вероятность ошибки  $R_f$  при обращении слова  $f(x)$  (для случайного  $x$ ) меньше доли трудных слов плюс верхняя оценка вероятности обращения с  $M$  попыток каждого из простых слов.

(2) Оценивая вероятность успешного обращения  $f^N(x_1, \dots, x_N)$  алгоритмом  $R_{f^N}$  на случайном входе, разделим пространство на две части: когда одно из слов  $x_i$  является трудным и когда все они легкие. Для второй части нам не важны значения  $\hat{s}(x)$ , мы пользуемся тем, что эта часть сама по себе имеет вероятность не больше  $(1 - \delta)^N$ .

Первая часть: мы оценим вероятность в случае, когда трудным оказалось слово  $x_i$ , а затем умножим оценку на  $N$  (вероятность объединения событий не больше суммы вероятностей). Вероятность того, что  $x_i$  оказалось трудным, равна  $\delta$ , и для каждого из трудных слов  $x_i$  условная вероятность успешной работы алгоритма  $R_{f^N}$  (при данном значении  $i$ -й координаты) меньше  $\varepsilon$ . В итоге получаем для первой части оценку  $N\delta\varepsilon$ .  $\square$

Положим  $M = n/\varepsilon$ . Тогда второе слагаемое в верхней оценке вероятности ошибки при обращении  $f$  (первое утверждение леммы) примерно равно  $e^{-n}$ . Как мы говорили, параметр  $\varepsilon$  будет выбран равным некоторому обратному полиному от  $n$ . Поэтому и число  $M$  будет некоторым полиномом от  $n$ . Следовательно, алгоритм  $R_f$  является полиномиальным. По условию вероятность ошибки  $R_f$  не меньше  $1/p(n)$  (при всех достаточно больших  $n$ ). Поэтому при всех достаточно больших  $n$  выполнено  $\delta \geq 1/2p(n)$ .

Итак, доля трудных  $\delta$  не меньше  $1/2p(n)$ . Теперь воспользуемся вторым утверждением леммы для оценки сверху вероятности успеха  $R_{f^N}$ . Нам надо выбрать  $\varepsilon, N$  так, чтобы эта оценка была строго меньшей

$1/q(n)$ . Как было уже обещано, положим  $N = np(n)$ . Тогда второе слагаемое в верхней оценке равно

$$\left(1 - \frac{1}{2p(n)}\right)^{np(n)} = \left[\left(1 - \frac{1}{2p(n)}\right)^{2p(n)}\right]^{n/2} \approx e^{-n/2}$$

а значит при больших  $n$  меньше  $1/2q(n)$ . Поэтому  $\varepsilon$  можно определить так, чтобы первое слагаемое было не больше  $1/2q(n)$ . Например, положим

$$\varepsilon = \frac{1}{2Nq(n)} = \frac{1}{2np(n)q(n)}.$$

Количество этапов  $M$  в алгоритме  $R_f$  будет равно  $n/\varepsilon = 2n^2p(n)q(n)$ . Теорема доказана.

*Задача 16.* Покажите, что в построении алгоритма  $R_f$  можно обойтись меньшим количеством этапов  $M = n^2q(n)$ .

## 2.6 Необратимые частичные функции

### Определение.

Мы рассматривали односторонние функции  $f_n$ , определенные на множестве слов данной длины  $k(n)$ . В этом разделе мы рассмотрим функции определенные на некотором подмножестве  $D_n$  множества  $\{0, 1\}^{k(n)}$ . Такие функции будем называть *частичными*. Необратимость частичных функций определяется аналогично необратимости всюду определенных функций, только теперь обращающей схеме на вход подается  $f_n(x)$  для случайным образом выбранного  $x \in D_n$  и все слова из  $D_n$  считаются равновероятными. Односторонней частичной функцией будем называть полиномиально вычислимую необратимую функцию, для которой можно за полиномиальное от  $n$  время порождать элементы из  $D_n$  так, чтобы все элементы порождались с примерно одинаковой вероятностью. Уточним последнее.

Последовательность распределений вероятностей  $\mu_n$  на множестве двоичных слов называется *полиномиально моделируемой*, если существует полиномиальный вероятностный алгоритм  $K$  такой, что для всех  $x \in \{0, 1\}^*$  выполнено

$$\Pr[K \text{ выдает } x \text{ на входе } n] = \mu_n(x).$$

Алгоритм  $K$  получает на вход число  $n$  в унарной записи и при любых исходах своих бросаний должен остановиться за полиномиальное от  $n$  время и выдать некоторую двоичную строку.

*Статистическим расстоянием* между распределениями вероятностей  $\mu$  и  $\nu$  на множестве двоичных слов называется максимум  $|\mu(A) - \nu(A)|$  по всем множествам слов  $A$ . Нетрудно понять, что этот максимум достигается на множестве слов  $A = \{x \mid \mu(x) > \nu(x)\}$  (а также на его дополнении) и равен  $\sum_{x \in \{0,1\}^*} |\mu(x) - \nu(x)|/2$ . Последовательности распределений вероятностей  $\mu_n$  и  $\nu_n$ , называются *статистически неотличимыми*, если статистическое расстояние между  $\mu_n$  и  $\nu_n$  стремится к нулю быстрее любого обратного многочлена от  $n$ . Аналогично определяется статистическая неотличимость случайных величин: случайные величины  $\alpha_n$  и  $\beta_n$ , зависящие от натурального параметра  $n$ , называются *статистически неотличимыми*, если их распределения  $\mu_n(x) = \text{Pr}[\alpha_n = x]$  и  $\nu_n(x) = \text{Pr}[\beta_n = x]$  статистически неотличимы.

Будем называть распределение  $\mu_n$  *доступным*, если  $\mu_n$  статистически неотлично от некоторого полиномиально моделируемого распределения  $\nu_n$ .

Примеры.

1. Пусть  $\mu_n$  не зависит от  $n$  и является равномерным распределением на множестве из трёх последовательностей  $\{00, 01, 10\}$  длины 2. Это семейство распределений не является полиномиально моделируемым. В самом деле, для любого полиномиально моделируемого распределения, вероятность каждой строки есть двоично-рациональное число, то есть, рациональное число со знаменателем  $2^{-k}$ . (Вероятность любого исхода равна доле всех бросаний, для которых моделирующий алгоритм выдаёт этот исход. А общее число исходов бросаний есть степень двойки.) Поскольку  $1/3$  не является двоично-рациональным числом, смоделировать это распределение за полиномиальное время невозможно<sup>1</sup>. Но за полиномиальное время можно смоделировать очень близкое распределение: делаем два бросания, если выпало 00, 01 или 10, то выдаём полученную последовательность на выход. Иначе повторяем попытку еще раз. Так делаем  $n$  раз, и если в каждой из  $n$  попыток получалось 11, то выдаём любую последовательность, например, 11. Чему равно статистическое расстояние между равномерным распределением на  $\{00, 01, 10\}$  и смодели-

<sup>1</sup>Это можно было бы сделать за полиномиальное в среднем время, но такие алгоритмы мы не рассматриваем.

лированным распределением? По смоделированному распределению исходы 00, 01 и 10 имеют меньшую вероятность, чем нужно, а исход 11 — бóльшую, чем нужно. Причём превышение равно  $4^{-n}$ , что и является статистическим расстоянием.

2. Пусть  $\mu_n$  есть равномерное распределение на множестве всех последовательностей длины  $n$ , кроме последовательности из одних единиц. Опять  $\mu_n$  не является полиномиально моделируемым, но является доступным. В самом деле, статистическое расстояние между равномерным распределением на всех словах длины  $n$  и  $\mu_n$  равно  $2^{-n}$ , а равномерное распределение очевидно полиномиально моделируемо.

*Определение 1.* Семейство частичных функций  $f_n$  называется *сильно односторонним*, если  $f_n$  полиномиально вычислимо, сильно необратимо и равномерное распределение  $\mu_n$  на области определения  $f_n$  доступно. Аналогично определяются слабо необратимые семейства частичных функций. Семейство частичных функций  $f_n$  называется *сильно односторонней перестановкой*, если для всех  $n$  функция  $f_n$  является перестановкой своей области определения  $D_n$  (то есть, множество её значений равно  $D_n$  и функция  $f_n$  инъективна).

## Примеры предположительно сильно односторонних частичных функций

Приведем примеры (предположительно) односторонних частичных функций. Все они являются перестановками. Известны три таких примера.

*Функция Рабина.* Функция Рабина  $f_n$  определена на словах вида  $xu$  (имеется в виду конкатенация слов) длины  $4n$ , где  $|x| = |y| = 2n$ , удовлетворяющих следующим требованиям. (а) Слово  $y$  есть двоичная запись числа  $p \cdot q$ , где  $p, q$  простые  $n$ -битовые числа вида  $4k + 3$ . (б) Слово  $x$  есть двоичная запись некоторого числа меньше  $y$ , являющегося полным квадратом по модулю  $y$  и взаимно просто с  $y$ . Значение функции на аргументе  $xu$  равно конкатенации слов  $x^2 \bmod y$  и  $y$ .

Покажем, что функция Рабина является перестановкой своей области определения  $D_n$ . Ясно, что  $f_n$  отображает  $D_n$  в себя. Покажем, что  $f_n$  инъективна. Пусть  $x = z^2$  и  $z$  взаимно просто с  $p$ . Тогда по  $x^2$  можно найти  $x$  по модулю  $p$ , возведя  $x$  в степень  $(p+1)/4$ . Действительно,  $(x^2)^{(p+1)/4} = z^{p+1} = z^2 = x \pmod{p}$ . То же самое верно и по модулю  $q$ , поэтому по  $x^2$  и  $y = pq$  можно восстановить  $x$ . (Из этого, конечно, не сле-

дует обратимость функции Рабина, поскольку для её обращения нужно разложить  $y$  на простые множители.) Задача обращения этой функции есть задача вычисления квадратного корня из данного числа по данному модулю. При составном модуле полиномиальных алгоритмов для извлечения квадратных корней неизвестно.

Алгоритм генерации случайной величины, статистически неотличимой от случайной величины, равномерно распределенной в области определения  $f_n$ , работает так. Выбираем случайно число вида  $4k + 3$  из  $n$  битов и проверяем, просто ли оно, скажем, с помощью полиномиального алгоритма проверки простоты (“алгоритма трех индийцев”). Если оно просто, положим  $p$  равным этому числу. Иначе повторяем попытку и т.д. Если после  $n^2$  попыток простое число не найдется, то положим  $p = 0$ . Из закона распределения простых чисел (количество простых чисел, меньших  $m$  примерно равно  $m/\ln m$ , причём примерно половина из них имеет вид  $4k + 3$ ) следует, что вероятность этого события будет меньше  $(1 - \varepsilon/n)^{n^2} < e^{-\varepsilon n}$  для некоторого положительного  $\varepsilon$ . (Можно вместо алгоритма трех индийцев использовать и вероятностный тест на простоту с вероятностью ошибки, скажем  $2^{-n}$ . Тогда статистическое расстояние будет больше на величину, равную вероятности того, что  $p$  или  $q$  составные, то есть  $2^{-n+1}$ .) Затем так же генерируем  $q$ . Число  $x$  выбираем равномерно среди натуральных чисел, меньших  $y$ . Число чисел, не взаимно простых с  $y$ , ничтожно, поэтому мы как раз получим распределение, статистически неотличимое от равномерного распределения. В самом деле, количество чисел, не взаимно простых с  $y = pq$  меньше  $p + q$ , а их доля меньше  $1/q + 1/p$ . При случайном выборе простого  $n$ -битового числа  $p$  среднее значение  $1/p$  пренебрежимо мало. В самом деле, с приблизительно единичной вероятностью число  $p$  окажется не меньшим  $2^{-n/2}$ .

*Функция RSA.* Функция RSA есть обобщение функции Рабина. Она определена на словах вида  $xyz$ , где  $x$ ,  $y$  и  $z$  имеют длину  $2n$  и понимаются как двоичные записи натуральных чисел, причем выполнены следующие условия: (а)  $y = p \cdot q$ , где  $p, q$  — простые  $n$ -битовые числа, (б)  $x$  находится в интервале  $(0, pq)$  и взаимно просто с  $y$ , (в)  $z$  взаимно просто со значением функции Эйлера  $\phi(pq) = (p - 1)(q - 1)$  на  $pq$ .

Значение функции  $f_n$  равно конкатенации  $(x^z \bmod y)$ ,  $y$  и  $z$ . Очевидно, что функция RSA отображает свою область определения  $D_n$  в себя. Инъективность следует из того, что по модулю  $\phi(pq)$  число  $z$  имеет обратный элемент  $u$ . Поэтому по  $x^z$  можно найти  $x$ , возведя  $x^z$  в степень  $u$ . Действительно,  $\phi(pq)$  есть мощность мультипликативной группы вы-



## 2.7. ДОПОЛНИТЕЛЬНЫЕ ТРЕБОВАНИЯ К ЧАСТИЧНЫМ ОДНОСТОРОННИМ ФУНКЦИЯМ

четов по модулю  $pq$  и по теореме Лагранжа мы имеем  $x^{zu} = x^{1+k\phi(pq)} = x \pmod{pq}$ . Из этого, конечно не следует, обратимость функции RSA, поскольку для обращения нужно знать  $u$  (или числа  $p$  и  $q$ , по которым  $u$  можно вычислить).

Доступность равномерного распределения на  $D_n$  не очевидна из-за того, что надо уметь порождать числа, взаимно простые с  $(p-1)(q-1)$ . Для этого нам нужно знать разложение  $(p-1)(q-1)$  на простые множители. Оказывается, можно порождать равномерное распределение на простых  $n$ -битовых числах вместе с разложением числа  $p-1$  на простые множители (см. [2]).

*Дискретная экспонента.* Функция  $f_n$  определена на словах длины  $3n$ , удовлетворяющим следующим условиям:  $y$  есть  $n$ -битовое простое число,  $x \in [2, y-1]$  порождает всю мультипликативную группу вычетов по модулю  $y$  (то есть любой ненулевой вычет является некоторой степенью  $x$ ), а  $z \in [1, y-1]$ .

Значение  $f_n$  на слове равно конкатенации  $x, y$  и  $x^z \pmod{y}$ :

$$f_n(xyz) = xy(x^z \pmod{y}).$$

Задача обращения этой функции есть задача вычисления логарифма данного числа по данному основанию в кольце вычетов по данному простому модулю. Эту задачу принято называть “дискретным логарифмированием”. Доступность равномерного распределения на области определения  $f_n$  не очевидна, это доказано в [2].

## 2.7 Дополнительные требования к частичным односторонним функциям

### Проверяемость равенства $f_n(x) = y$

Для некоторых применений частичных односторонних функций нам нужно еще следующее условие: по  $n$ , любому слову  $x$  длины  $k(n)$  и любому слову  $y$  из множества значений  $f_n$  можно за полиномиальное от  $n$  время определить, верно ли равенство  $f_n(x) = y$  (это равенство включает в себя принадлежность  $x$  к области определения  $f_n$ ). Это условие не является тривиально выполненным, поскольку алгоритм вычисления  $f_n$  может на данном входе  $x$  вне области определения функции  $f_n$  выдать некоторый

результат  $y$ , принадлежащий множеству значений функции  $f_n$ . (В этом случае мы ошибочно подумаем, что  $f_n(x) = y$ .) Если же  $f_n$  всюду определена, что это условие тривиально выполнено. Оно выполнено также для частичных односторонних функций, область определения которых разрешима за полиномиальное время.

*Определение 2.* Частичная функция  $f^n : \{0, 1\}^{k(n)} \rightarrow \{0, 1\}^{l(n)}$  называется *проверяемой*, если по любому  $n$ , любому слову  $x$  длины  $k(n)$  и любому слову  $y$  из множества значений  $f_n$  можно за полиномиальное от  $n$  время определить, верно ли,  $f_n$  определено на  $x$  и её значение на  $x$  равно  $y$ .

Функции RSA и дискретная экспонента являются проверяемыми. Про функцию Рабина это непонятно, поскольку неясно, как за полиномиальное время можно проверить, является ли данное число квадратичным вычетом по составному модулю.

## Односторонние перестановки с секретом

На самом деле речь идет не об одной функции, а о семействе функций. Грубо говоря, необратимой перестановкой с секретом (trapdoor permutation) называется семейство перестановок  $f_n^e : D_n^e \rightarrow D_n^e$  такие, что функция  $\langle e, x \rangle \mapsto \langle e, f_n^e(x) \rangle$  односторонняя, но при этом существует полиномиальный алгоритм, который по  $f_n^e(x)$  и некоторой дополнительной информации  $d$ , зависящей только от  $e$ , находит  $x$ .

Теперь формальное определение. Пусть задана полиномиально генерируемая последовательность случайных величин  $\langle e_n, d_n \rangle$ . Будем через  $A_n$  обозначать первые компоненты всех возможных значений  $\langle e_n, d_n \rangle$ . Пусть для каждого  $e \in A_n$  задана перестановка  $f_n^e$  некоторого множества слов  $D_n^e$  длины  $\text{poly}(n)$ . Последовательность пар  $\langle \langle e_n, d_n \rangle, \{f_n^e\} \rangle$  называется односторонней перестановкой с секретом, если выполнены следующие условия.

(а) (Полиномиальная вычислимость.) Функция  $\langle 1^n, e, x \rangle \mapsto f_n^e(x)$  вычислима за полиномиальное от  $n$  время (существует полиномиальный алгоритм, который по любой тройке  $\langle 1^n, e, x \rangle$ , где  $e \in A_n$ ,  $x \in D_n^e$ , вычисляет  $f_n^e(x)$ ).

(б) (Необратимость.) Для любой последовательности схем  $C_n$  размера  $\text{poly}(n)$  вероятность того, что  $C_n$  по  $\langle e, f_e(x) \rangle$  найдет  $x$  стремится к нулю быстрее любого обратного полинома от  $n$ . Здесь  $e$  выбирается случайно, как первая компонента случайной величины  $\langle e_n, d_n \rangle$ , а  $x$  выбирается

## 2.7. ДОПОЛНИТЕЛЬНЫЕ ТРЕБОВАНИЯ К ЧАСТИЧНЫМ ОДНОСТОРОННИМ ФУНКЦИЯМ

независимо от  $e$  (все слова из  $D^e$  предполагаются равновероятными).

(в) (Доступность равномерного распределения на области определения.) Существует вероятностный полиномиальный алгоритм  $B$ , который по  $1^n$  и любому  $e \in A_n$  генерирует случайную величину, статистически неотличимую от равномерно распределенной случайной величины в  $D_n^e$  при известном  $e_n$ . (То есть, пара, состоящая из  $e$  и выхода алгоритма статистически неотличима от пары, состоящей из  $e$  и случайной строки из  $D_n^e$ .)

Из этого условия следует доступность случайной величины

$$\langle e_n, \text{случайная строка из } D_n^e \rangle$$

(сначала генерируем  $e_n$ , а затем примеряем алгоритм  $B$ ). Поэтому из условий (а), (б), (в) следует, что функция  $\langle e, x \rangle \mapsto \langle e, f_n^e(x) \rangle$  является обобщенной односторонней перестановкой.

(г) (Возможность обращения при известном закрытом ключе.) Существует полиномиальный вероятностный алгоритм, который по тройке  $\langle 1^n, d, f_e(x) \rangle$  с вероятностью приблизительно равной 1 вычисляет  $x$ . Здесь пара  $\langle e, d \rangle$  выбирается генератором случайной величины  $\langle e_n, d_n \rangle$ , а  $x$  выбирается равномерно в  $D_n^e$ .

Поскольку неизвестно, существуют ли односторонние перестановки, тем более неизвестно, существуют ли односторонние перестановки с секретом. Гипотеза об их существовании довольно правдоподобна. Действительно, если функции Рабина или функция RSA в самом деле необратима, то существуют односторонние функции с секретом. Убедимся в этом.

*Функция Рабина.* Случайная величина  $\langle e_n, d_n \rangle$  равномерно распределена в множестве всех пар  $\langle pq, \langle p, q \rangle \rangle$ , где  $p, q$  —  $n$ -битовые простые числа вида  $4k + 3$  (точнее, полиномиально моделируемая величина, статистически неотличимая от этой). Множество  $D_n^e$  есть множество всех квадратичных вычетов по модулю  $pq$ , принадлежащих интервалу  $0, \dots, pq - 1$ , при этом  $f_n^e(x) = x^2 \bmod pq$ . Условия (а), (в) и (г) очевидно выполнены. Условие (б) будет выполнено, если функция Рабина необратима.

*Функция RSA.* Случайная величина  $\langle e_n, d_n \rangle$  равномерно распределена в множестве всех пар  $\langle \langle pq, i \rangle, \langle pq, j \rangle \rangle$ , где  $p, q$   $n$ -битовые простые числа,  $i, j \in \{1, \dots, pq - 1\}$  и  $ij \equiv 1 \pmod{(p-1)(q-1)}$ . (Точнее, полиномиально моделируемая величина, статистически неотличимая от этой.) Множество  $D_n^e$  есть множество всех  $0 < x < pq$ , взаимно простых с  $pq$ , при этом

$f_n^e(x) = x^i \bmod pq$ . Условия (а), (в) и (г) очевидно выполнены. Условие (б) будет выполнено, если функции RSA в самом деле необратима.

## Улучшенные перестановки с секретом

Улучшенная перестановка с секретом (enhanced trapdoor permutation)—это односторонняя перестановка с секретом, удовлетворяющая усиленному требованию необратимости. А именно, требуется, чтобы даже зная случайные биты, использованные при выборе случайного элемента  $y$  из области определения  $D_n^e$  перестановки  $f_e$  было нельзя найти  $f_e^{-1}(y)$ . Для того, чтобы это сформулировать аккуратно, нужно вспомнить требование доступности равномерного распределения на области определения, которое тоже нужно немного усилить

(Доступность) Обозначим через  $\gamma_n^e$  случайную величину, равномерно распределенную в  $D_n^e$ . Существует вероятностный полиномиальный алгоритм  $S$ , который по  $1^n$  и любому открытому ключу  $e$  генерирует случайную величину  $\xi_n^e$  статистически неотличимую от  $\gamma_n^e$ .

Обозначим через  $S(1^n, e, r)$  последовательность, напечатанную алгоритмом  $S$  при случайных битах  $r$  (длина  $r$  предполагается равной некоторому фиксированному полиному  $p(n)$ ). Теперь требования необратимости формулируется так.

(Необратимость) Для любой последовательности схем  $C_n$  размера  $\text{poly}(n)$  вероятность того, что  $C_n$  по  $\langle e, r \rangle$  найдет  $f_e^{-1}(S(1^n, e, r))$  стремится к нулю быстрее любого обратного полинома от  $n$ . Здесь  $e$  выбирается случайно, как первая компонента случайной величины  $\langle e_n, d_n \rangle$ , а  $r$  выбирается независимо от  $e$ .

## Задачи

*Задача 17.* Докажите, что существует семейство функций  $g_n$ , не являющаяся сильно необратимым и такое, что функцию  $g_n$  можно сузить на некоторое множество, получив сильно необратимое семейство.

*Задача 18.* Предположим, существует сильно односторонняя частичная функция. Докажите, что тогда существует и сильно односторонняя всюду определенная функция.

*Задача 19.* Предположим, существует слабо односторонняя частичная функция. Докажите, что тогда существует и слабо односторонняя всюду

## 2.7. ДОПОЛНИТЕЛЬНЫЕ ТРЕБОВАНИЯ К ЧАСТИЧНЫМ ОДНОСТОРОННИМ ФУНКЦИЯМ

определенная функция.

*Задача 20.* Предположим, существует слабо односторонняя частичная функция. Докажите, что тогда существует и сильно односторонняя всюду определенная функция.

*Задача 21.* Докажите, что существует семейство всюду определенных функций  $f_n$ , не являющееся даже слабо необратимым, и при этом для некоторой последовательности множеств  $D_n$  при сужении  $f_n$  на  $D_n$  получается сильно необратимое семейство.

*Задача 22.* Докажите, что существует слабо необратимое семейство всюду определённых функций  $f_n$  такое, что для любой последовательности множеств  $D_n$  семейство, получаемое сужением  $f_n$  на  $D_n$ , не является сильно необратимым.



## Глава 3

# Генераторы псевдослучайных чисел

### 3.1 Вычислительно неотличимые случайные величины

*Для неравномерного противника:* случайные величины  $\alpha_n$  и  $\beta_n$ , зависящие от натурального параметра  $n$ , со значениями в множестве слов некоторой длины  $l(n)$  называются вычислительно неотличимыми, если для любой последовательности схем  $C_0, C_1, \dots, C_n, \dots$  размера  $\text{poly}(n)$  (с  $l(n)$  входами и одним выходом) вероятность событий  $C_n(\alpha_n) = 1$  и  $C_n(\beta_n) = 1$  отличаются на пренебрежимо малую величину. Схема  $C_n$  в этом контексте называется тестом и мы говорим, что случайная величина  $\alpha_n$  проходит тест  $C_n$ , если  $C_n(\alpha_n) = 1$ . Таким образом, мы требуем, чтобы  $\alpha_n$  и  $\beta_n$  проходили любые тесты полиномиального размера с приблизительно равной вероятностью.

*Для равномерного противника* в этом определении надо заменить последовательность схем  $C_0, C_1, \dots, C_n, \dots$  на вероятностный алгоритм, получающий на вход число  $n$  в унарной записи и значение случайной величины. Как и в случае необратимых функций, равномерное определение слабее неравномерного.

Если  $\alpha_n$  и  $\beta_n$  статистически неотличимы, то они и вычислительно неотличимы (для неравномерного противника), поскольку разность вероятностей попадания  $\alpha_n$  и  $\beta_n$  в множество  $\{x \mid C_n(x) = 1\}$ , задаваемое тестом  $C_n$ , не превосходит статистического расстояния между  $\alpha_n$  и  $\beta_n$ .

Нам понадобятся некоторые простые свойства понятия вычислительной неотличимости.

**Лемма.** 1) Отношение вычислительной неотличимости рефлексивно, симметрично и транзитивно.

2) Для неравномерного противника: вычислительно неотличимые последовательности случайных величин  $\alpha_n$  и  $\beta_n$  вычислительно неотличимы и вероятностными тестами полиномиального размера. Это означает, что для любой последовательности  $T_n$  вероятностных схем полиномиального от  $n$  размера с  $l(n)$  входами, вероятности событий  $T_n(\alpha_n) = 1$  и  $T_n(\beta_n) = 1$  приблизительно равны.

3) Если  $\alpha_n$  и  $\beta_n$  вычислительно неотличимы, а  $C_n$  последовательность вероятностных схем полиномиального от  $n$  размера с  $l(n)$  входами, то и случайные величины  $C_n(\alpha_n)$  и  $C_n(\beta_n)$  вычислительно неотличимы. (Для равномерного противника в этом свойстве последовательности схем надо заменить на полиномиальный вероятностный алгоритм, получающий на вход число  $n$  в унарной записи и значение случайной величины.)

4) Пусть случайные величины  $\alpha_n$ ,  $\beta_n$  и  $\gamma_n$  имеют совместное распределение. И пусть для любой последовательности значений  $c_n$  случайной величины  $\gamma_n$  случайные величины  $(\alpha_n | \gamma_n = c_n)$  и  $(\beta_n | \gamma_n = c_n)$  вычислительно неотличимы.<sup>1</sup> Тогда  $\alpha_n \gamma_n$  и  $\beta_n \gamma_n$  вычислительно неотличимы.

Для равномерного противника это свойство справедливо только для независимых случайных величин  $\alpha_n, \gamma_n$ , причем случайная величина  $\gamma_n$  должна быть полиномиально моделируемой.

*Доказательство.* Первое и второе утверждение очевидны.

3) Пусть дана последовательность тестов  $T_n$  полиномиального от  $n$  размера с  $s(n)$  входами (где  $s(n)$  — количество выходов  $C_n$ ) на отличимость  $C_n(\alpha_n)$  и  $C_n(\beta_n)$ . Тогда последовательность схем  $D_n(x) = T_n(C_n(x))$  можно рассматривать, как вероятностный тест на отличимость  $\alpha_n$  и  $\beta_n$ . Размер схемы  $D_n$  есть сумма размеров  $T_n$  и  $C_n$ , а значит ограничен полиномом от  $n$ . По второму свойству вероятность того, что  $\alpha_n$  проходит тест  $D_n$  пренебрежимо мало отличается от вероятности того, что  $\beta_n$  проходит тест  $D_n$ . Осталось заметить, что первая вероятность равна вероятности

<sup>1</sup> $(\alpha_n | \gamma_n = c_n)$  обозначает случайную величину с тем же множеством значений, что и  $\alpha_n$ , которая принимает значение  $a$  с вероятностью  $\Pr[\alpha_n = a | \gamma_n = c_n]$ .



### 3.1. ВЫЧИСЛИТЕЛЬНО НЕОТЛИЧИМЫЕ СЛУЧАЙНЫЕ ВЕЛИЧИНЫ 33

того, что случайная величина  $C_n(\alpha_n)$  пройдет тест  $T_n$ , и то же самое верно для  $C_n(\beta_n)$ .

4) Допустим, что существует последовательность схем-тестов  $T_n$  полиномиального от  $n$  размера такая, что вероятности событий  $T_n(\alpha_n\gamma_n) = 1$  и  $T_n(\beta_n\gamma_n) = 1$  отличаются на  $\varepsilon = 1/\text{poly}(n)$  для бесконечно многих  $n$ . Разность вероятностей событий  $T_n(\alpha_n\gamma_n) = 1$  и  $T_n(\beta_n\gamma_n) = 1$  равна среднему значению разности вероятностей

$$\Pr[T_n(\alpha_n c) = 1 | \gamma_n = c] - \Pr[T_n(\beta_n c) = 1 | \gamma_n = c]$$

по случайно выбранному  $c$  (в соответствии с распределением случайной величины  $\gamma_n$ ). Поэтому для бесконечно многих  $n$  найдется  $c = c_n$  в множестве значений  $\gamma_n$ , для которого эта разность не меньше  $\varepsilon$  по абсолютной величине. Запавив значение  $c_n$  в схему  $T_n$ , мы получим отличитель случайных величин  $(\alpha_n | \gamma_n = c_n)$  и  $(\beta_n | \gamma_n = c_n)$ , что противоречит условию.  $\square$

*Задача 23.* Пусть случайные величины  $\alpha_n$  и  $\beta_n$  вычислительно неотличимы. Докажите, что в результате отрезания от них начала полиномиальной длины получатся вычислительно неотличимые случайные величины.

*Задача 24.* Пусть случайные величины  $\alpha_n$  и  $\beta_n$  вычислительно неотличимы, а  $\gamma_n$  произвольная случайная величина, независимая от  $\alpha_n$  и  $\beta_n$ . Докажите, что  $\alpha_n\gamma_n$  и  $\beta_n\gamma_n$  вычислительно неотличимы.

В некоторых приложениях нам понадобится понятие семейства попарно неотличимых величин. Пусть для каждого натурального  $n$  задано множество слов  $D_n$  полиномиальной от  $n$  длины и для каждого  $u \in D_n$  задана случайная величина  $\alpha_n^u$  со значениями в некотором множестве слов полиномиальной от  $n$  длины (эта длина зависит только от  $n$ ). Мы говорим, что случайные величины  $\alpha_n^u$  попарно вычислительно неотличимы, если для любой последовательности пар слов  $u_n, v_n \in D_n$  случайные величины  $\alpha_n^{u_n}$  и  $\alpha_n^{v_n}$  вычислительно неотличимы.

**Лемма.** Случайные величины  $\alpha_n^u$  попарно вычислительно неотличимы тогда и только тогда, когда любой последовательности схем  $T_n$  полиномиального размера найдется пренебрежимо малая последовательность  $\varepsilon_n$  для которой

$$|\Pr[T_n(\alpha_n^u) = 1] - \Pr[T_n(\alpha_n^v) = 1]| < \varepsilon_n$$

для всех  $u, v \in D_n$ .

*Доказательство.* Утверждение “тогда” следует непосредственно из определения. Докажем утверждение “только тогда”. Допустим случайные величины  $\alpha_n^u$  попарно вычислительно неотличимы. Зафиксируем произвольную последовательности схем  $T_n$  полиномиального размера. Для каждого  $n$  рассмотрим пару слов  $u_n, v_n \in D_n$  для которых разность

$$\Pr[T_n(\alpha_n^{u_n}) = 1] - \Pr[T_n(\alpha_n^{v_n}) = 1]$$

максимальна по абсолютной величине. Обозначим через  $\varepsilon_n$  абсолютную величину этой разности для этих  $u_n, v_n$ . По условию  $\varepsilon_n$  пренебрежимо мало.  $\square$

*Задача 25.* Докажите, что определение обобщенной односторонней функции не изменится, если потребовать, чтобы трудная случайная была вычислительно (а не статистически) неотличима от некоторой случайной величины, моделируемой за полиномиальное время. Докажите, что оно также не изменится, если потребовать, чтобы трудная случайная сама генерировалась за полиномиальное время.

## 3.2 Определение генератора ПСЧ

Пусть даны многочлены  $k(n), l(n)$  такие, что  $l(n) > k(n)$  для всех  $n$ . Генератором ПСЧ типа  $k(n) \rightarrow l(n)$  будем называть семейство функций  $G_n$ , где для каждого  $n$  функция  $G_n$  отображает двоичные слова длины  $k(n)$  в слова длины  $l(n)$ , удовлетворяющее следующим двум условиям:

(а) Семейство  $G_n$  вычислимо за полиномиальное время (по данным  $n$  и  $s$  за полиномиальное от  $n$  количество шагов можно найти  $G_n(s)$ ).

(б) Случайная величина  $G_n(s)$  (где  $s$  выбирается случайно среди всех строк длины  $k(n)$ ) вычислительно неотличима от равномерно распределенной среди слов длины  $l(n)$  случайной величины (при стремлении  $n$  к бесконечности). Это свойство генератора называют надежностью.

*Задача 26.* Докажите, что второе требование нельзя усилить, потребовав статистической неотличимости  $G_n(s)$  и равномерно распределенной случайной величины.

*Задача 27.* Докажите, что композиция генераторов ПСЧ типа  $k(n) \rightarrow l(n)$  и  $l(n) \rightarrow m(n)$  является генератором ПСЧ типа  $k(n) \rightarrow m(n)$ .

Определим также генераторы ПСЧ типа  $k(n) \rightarrow \infty$ , как семейства  $G_n$ , где  $G_n$  отображает двоичные слова длины  $k(n)$  в бесконечные двоичные последовательности, удовлетворяющие следующим требованиям.

(а) Существует алгоритм, который по слову  $s$  и натуральному  $l$  за полиномиальное от  $|s| + l$  время вычисляет  $l$ -ый бит последовательности  $G_n(s)$ .

(б) Случайная величина  $G_n(s)$  вычислительно неотличима от равномерно распределенной бесконечной последовательности нулей и единиц.

В этом определении вычислительная неотличимость двух случайных величин  $\omega_n, \xi_n$  со значениями в множестве всех бесконечных двоичных последовательностей понимается в следующем смысле: для любой последовательности схем  $C_n$  размера  $\text{poly}(n)$  вероятность событий  $C_n(\omega_n) = 1$  и  $C_n(\xi_n) = 1$  отличаются на пренебрежимо малую величину. Здесь  $C_n(\omega_n)$  обозначает результат применения схемы к  $k$  первым битам  $\omega_n$ , где  $k$  — количество входов  $C_n$ . Аналогично понимается  $C_n(\xi_n)$ . Другими словами,  $\omega_n$  и  $\xi_n$  вычислительно неотличимы, если для любого полинома  $p(n)$  вычислительно неотличимы случайные величины, равные и первым  $p(n)$  битам случайных величин  $\omega_n$  и  $\xi_n$ .

Имеется следующее соотношение между генераторами типа  $k(n) \rightarrow l(n)$  и генераторами типа  $k(n) \rightarrow \infty$ . Если  $G_n$  — генератор типа  $k(n) \rightarrow \infty$ , то для любого полинома  $l(n)$  последовательность функций  $H_n(s) = (G_n(s))_{l(n)}$ , где  $s \in \{0, 1\}^{k(n)}$ , будет генератором типа  $k(n) \rightarrow l(n)$  (что очевидно). Обратно, мы докажем, что по любому генератору типа  $k(n) \rightarrow l(n)$  (какие бы ни были  $k(n) < l(n)$ ) можно построить генератор типа  $k(n) \rightarrow \infty$ .

*Задача 28.* Докажите, что существует функция  $G_n$  (не обязательно вычислимая за полиномиальное время), типа  $n \rightarrow n + 1$  такая, что случайная величина  $G_n(s)$  вычислительно неотличима от равномерно распределенной случайной величины.

*Задача 29.* Докажите, что если функция  $G_n$  является генератором типа  $k(n) \rightarrow l(n)$ , а  $x_n$  последовательность слов длины  $l(n)$ , вычислимая за время  $\text{poly}(n)$ , то функция  $s \mapsto G_n(s) \oplus x_n$  является генератором.

### 3.3 Генераторы ПСЧ и односторонние функции

Любой генератор  $G_n$  типа  $k(n) \rightarrow l(n)$  является слабо необратимой функцией. Действительно, никакая последовательность схем  $C_n$  полиномиального размера не может обратить  $G_n(s)$  в вероятность успеха, большей  $3/4$  (для бесконечно многих  $n$ ). Допустим, такая последовательность схем существует. Тогда рассмотрим следующий тест на последовательностях длины  $l(n)$ : применяем к последовательности  $y$  схему  $C_n$ , затем проверяем, с помощью алгоритма, вычисляющего  $G_n$ , правильно ли схема  $C_n$  обратила  $y$  (то есть,  $G_n(C_n(y)) = y$ ). Если обращение произошло удачно, то выдаем 1, а иначе 0. Вероятность того, что тест будет пройден случайной величиной  $G_n(s)$ , не менее  $3/4$  (для бесконечно многих  $n$ ). Вероятность того, что равномерно распределенная случайная величина пройдет тест, не больше  $1/2$ , поскольку множество значений  $G_n$  составляет не более половины всего множества последовательностей длины  $l(n)$ . Поскольку  $C_n$  и  $G_n$  можно вычислить схемой полиномиального от  $n$  размера, описанный тест имеет полиномиальный размер.

Итак, если для каких-то  $l(n) > k(n)$  существует генератор типа  $k(n) \rightarrow l(n)$ , то существуют и слабо односторонние, а значит и сильно односторонние функции. Оказывается, верно и обратное.

**Теорема 3.** [3] *Если существует сильно односторонняя функция, то существует и генератор ПСЧ типа  $n \rightarrow \infty$ .*

Мы докажем более слабое утверждение: если существует односторонняя перестановка, то существует и генератор ПСЧ типа  $n \rightarrow \infty$ .

**Теорема 4.** *Если существует односторонняя обобщенная перестановка, то существует генератор типа  $n \rightarrow \infty$ .*

### 3.4 Трудный бит

Сначала мы научимся добавлять один случайный бит к уже имеющимся. Для этого нам понадобится понятие трудного бита для данной функции. Пусть  $\beta_n, \gamma_n$  последовательность пар совместно распределенных случайных величин, причем случайная величина  $\beta_n$  имеет значения 0,1. Мы говорим, что  $\beta_n$  является трудно вычислимым по  $\gamma_n$  если для любой последовательности схем из функциональных элементов  $C_n$  размера  $\text{poly}(n)$

вероятность события  $C_n(\gamma_n) = \beta_n$  приблизительно равна  $1/2$  (то есть, стремится к  $1/2$  быстрее любого обратного полинома от  $n$ ). Это означает, что случайную величину  $\beta_n$  нельзя вычислить по  $\gamma_n$  с вероятностью существенно лучшей, чем при простом угадывании. Если  $\beta_n$  является трудно вычислимой по  $\gamma_n$ , то оба своих значения она принимает с примерно равной вероятностью (иначе в качестве  $C_n$  можно взять схему, которая выдает 0 независимо от входа).

*Определение 3.* Полиномиально вычислимая функция  $h_n : D_n \rightarrow \{0, 1\}$  является трудным битом для функции  $g_n : D_n \rightarrow D_n$ , если случайная величина  $h_n(\alpha_n)$  трудно вычислима по случайной величине  $g_n(\alpha_n)$ , где  $\alpha_n$  — случайная величина, равномерно распределенная в  $D_n$ .

Аналогично определяется понятие трудного бита для односторонней перестановки с секретом. Назовем семейство функций  $h_n^e : D_n^e \rightarrow \{0, 1\}$ , трудным битом к  $f_n^e$ , если случайная величина  $h_n^{e_n}(x)$  трудно вычислима по случайной величине  $\langle f_n^{e_n}(x), e_n \rangle$  (где  $x$  выбирается равномерно из  $D_n^{e_n}$  независимо от  $e_n$ ).

*Задача 30.* Докажите, что если имеется трудный бит для полиномиально вычислимой перестановки  $g_n : D_n \rightarrow D_n$ , то перестановка сильно необратима.

Трудным битом для функции Рабина является последний бит (четность), в предположении необратимости функции Рабина. Трудным битом для дискретной экспоненты (в предположении её необратимости) является функция  $h_n(xyz) = 0$ , если  $z < (y - 1)/2$ , и  $h_n(xyz) = 1$  иначе.

**Лемма (Яо (Yao)).** Пусть дана последовательность пар совместно распределенных случайных величин  $\beta_n, \gamma_n$ , причём  $\beta_n$  принимает значения в множестве  $0, 1$ . Пусть случайная величина  $r_n$  равномерно распределена в  $0, 1$  и независима от  $\gamma_n$ . Тогда случайная величина  $\beta_n$  трудно вычислима по  $\gamma_n$  тогда и только тогда, когда случайные величины  $\beta_n \gamma_n$  и  $r_n \gamma_n$  вычислительно неотличимы.

*Доказательство.* В одну сторону утверждение леммы очевидно: если  $\beta \gamma$  и  $r \gamma$  вычислительно неотличимы, то  $\beta$  является трудным битом для  $\gamma$ . (Мы опускаем индекс  $n$ .) Действительно, пусть имеется последовательность схем  $C_n$  полиномиального от  $n$  размера такая, что для бесконечно многих  $n$  вероятность события  $C(\gamma) = \beta$  отличается от  $1/2$  не менее, чем на  $\varepsilon = 1/\text{poly}(n)$ . Рассмотрим следующий тест  $D$ : он вычисляет результат работы схемы  $C$  на входной последовательности без первого бита и

выдает 1, если результат равен первому биту входной последовательности, и 0 иначе. Случайная величина  $r\gamma$  проходит этот тест с вероятностью ровно  $1/2$ , поскольку ее первый бит не зависит от последующих битов и принимает значения 0,1 с равными вероятностями. Вероятность того, что последовательность  $\beta\gamma$  пройдет этот тест, в точности равна вероятности того, что  $C(\gamma) = \beta$ . Ясно, что тест  $D$  задается схемой полиномиального от  $n$  размера.

Теперь докажем обратное. Допустим, что существует последовательность схем  $D_n$  размера  $\text{poly}(n)$  такая, что для бесконечно многих  $n$  выполнено

$$\Pr[D(\beta\gamma) = 1] - \Pr[D(r\gamma) = 1] = \varepsilon,$$

где  $|\varepsilon| \geq 1/\text{poly}(n)$  (мы опять опускаем индекс  $n$ ). Фиксируем любое из этих бесконечно многих  $n$ . Пусть для определенности  $\varepsilon > 0$ . Это значит, что схема  $D$  “больше любит”  $\beta$ , чем случайный бит. Рассмотрим следующий алгоритм  $C$  вычисления  $\beta$  по  $\gamma$ .

Пусть нам надо некоторое возможное значение  $x$  случайной величины  $\gamma$ . Нам нужно, зная  $x$ , выдать по возможности правильный прогноз для значения  $\gamma$ . Поступаем следующим образом. Если  $D(0x) = D(1x)$ , то выдаем 0 и 1 с равными вероятностями. Если  $D(0x) = 0$ ,  $D(1x) = 1$ , выдаем 1 (неформальное объяснение: при данном нам значении  $x$  схема любит 1 и не любит 0, при этом мы знаем, что в схема (в среднем, при случайном выборе  $x$ ) больше любит  $\beta$ , чем случайный бит, значит  $\beta$  скорее равно 1, чем 0). Если  $D(0x) = 1$ ,  $D(1x) = 0$ , выдаем 0.

Оказывается, вероятность того, что этот алгоритм выдаст  $\beta$ , равна  $1/2 + \varepsilon$ . Чтобы доказать это, достаточно доказать это при любом фиксированном значении  $x$  случайной величины  $\gamma$  выполнено равенство

$$\Pr[C(x) = \beta] = 1/2 + \Pr[D(\beta x) = 1] - \Pr[D(rx) = 1]. \quad (3.1)$$

(В этой формуле все вероятности вычисляются при условии  $\gamma = x$ .) Пусть  $x$  фиксировано. Если  $D(0x) = D(1x)$ , то схема не чувствует замены 0 на 1. Поэтому правая часть равенства (3.1) равна  $1/2$ . Левая часть также равна  $1/2$  по построению схемы  $C$ .

Если  $D(0x) = 0$ ,  $D(1x) = 1$ , то левая часть равенства (3.1) равна вероятности события  $\beta = 1$  (при условии  $\gamma = x$ ), а в правая равна  $1/2 + \Pr[\beta = 1] - 1/2$ , и равенство выполнено. Если  $D(0x) = 1$ ,  $D(1x) = 0$ , то левая часть равенства (3.1) равна (условной) вероятности события  $\beta = 0$ , а правая равна  $1/2 + \Pr[\beta = 0] - 1/2$ .

Алгоритм  $C$  задается вероятностной схемой примерно вдвое большей, чем  $D$ , поэтому имеющей также полиномиальный размер. Можно обойтись и схемой примерно того же размера, если заметить, что  $C(\gamma) = D(\gamma r) \oplus r \oplus 1$ , где  $r$  случайный бит, независимый от  $\gamma$ .  $\square$

Пусть  $h_n$  — трудный бит для односторонней перестановки  $g_n : D_n \rightarrow D_n$ , а случайная величина  $\alpha_n$  равномерно распределена в  $D_n$ . По лемме случайные величины  $h_n(\alpha_n)g_n(\alpha_n)$  и  $rg_n(\alpha_n)$  вычислительно неотличимы. Вторая имеет то же распределение, что и  $r\alpha_n$ . Таким образом, случайные величины  $h_n(\alpha_n)g_n(\alpha_n)$  и  $r\alpha_n$  вычислительно неотличимы. Говоря неформально, случайная величина  $h_n(\alpha_n)g_n(\alpha_n)$  имеет на один бит случайности больше, чем  $\alpha_n$ . Повторяя этот прием, можно добавить и второй случайный бит. А именно, случайная величина  $h_n(\alpha_n)h_n(g_n(\alpha_n))g_n^2(\alpha_n)$  вычислительно неотличима от случайной величины  $r_1r_2\alpha_n$ , где  $r_1, r_2$  — случайные биты, независимые от  $\alpha_n$ . В самом деле, обе случайные величины вычислительно неотличимы от случайной величины  $r_1h(\alpha)g(\alpha)$ . Неотличимость этой случайной величины от  $r_1r_2\alpha$  следует из неотличимости  $h(\alpha)g(\alpha)$  и  $r_2\alpha$  (обе случайные величины получаются приписыванием к этим случайным величинам одного бита). А неотличимость случайных величин  $h(\alpha)h(g(\alpha))g^2(\alpha)$  и  $r_1h(\alpha)g(\alpha)$  устанавливается немного сложнее. Рассмотрим преобразование  $T$ , которое переводит строку  $r_1\alpha$  в строку  $r_1h(\alpha)g(\alpha)$ . Это преобразование вычислимо схемой полиномиального размера. Кроме того, оно переводит случайную величину  $h(\alpha)g(\alpha)$  в случайную величину  $h(\alpha)h(g(\alpha))g^2(\alpha)$ . Поскольку случайные величины  $r_1\alpha$  и  $h(\alpha)g(\alpha)$  неотличимы, будет неотличимы и результаты применения к ним преобразования  $T$ .

Этот приём можно повторять и таким образом генерировать сколько угодно случайных битов. А именно, пусть дан произвольный полином  $p(n)$ , задающий количество случайных битов, которые нам нужно получить. Рассмотрим последовательность

$$h\alpha, hg\alpha, hgg\alpha, \dots, hg^{p(n)-1}\alpha, g^{p(n)}\alpha \quad (3.2)$$

(для наглядности мы опускаем индекс  $n$  и скобки в выражении  $g(\alpha)$ ). Утверждается, что она вычислительно неотличима от случайной величины

$$r_1, r_2, \dots, r_{p(n)}, \alpha, \quad (3.3)$$

где  $r_1r_2 \dots r_{p(n)}$  случайная равномерно распределенная строка длины  $p(n)$ , независимая от  $\alpha$ .

**Лемма.** Пусть  $h_n$  является трудным битом для полиномиально вычислимой перестановки  $g_n : D_n \rightarrow D_n$ . Тогда для любого полинома  $p(n)$  случайные величины (3.2) и (3.3) вычислительно неотличимы. Здесь  $\alpha$  равномерно распределена в  $D_n$ , а  $r_1 r_2 \dots r_{p(n)}$  случайная равномерно распределенная строка длины  $p(n)$ , независимая от  $\alpha$ .

*Доказательство.* Мы уже установили это для случаев  $p(n) \equiv 1$  и  $p(n) \equiv 2$ . Нам надо доказать это утверждение в общем случае. Будем рассуждать от противного и допустим, что некоторая последовательность схем  $C_n$  полиномиального размера отличает случайные величины

$$h\alpha, hg\alpha, \dots, hg^{p(n)-1}\alpha, g^{p(n)}\alpha \quad \text{и} \quad r_1, r_2, \dots, r_{p(n)}, \alpha.$$

То есть, для бесконечно многих  $n$  вероятность того, что первая случайная величина проходит тест  $C_n$ , отличается не менее, чем на  $\varepsilon_n = 1/\text{poly}(n)$ , от вероятности того, что вторая случайная величина проходит тест  $C_n$ . Чтобы получить противоречие, мы построим схему полиномиального размера, отличающую  $h\alpha, g\alpha$  от  $r, \alpha$  для тех самых  $n$ .

Фиксируем любое такое  $n$  и для каждого  $i = 0, \dots, p(n)$  рассмотрим гибридную случайную величину

$$H_i = r_1, r_2, \dots, r_i, h\alpha, hg\alpha, \dots, hg^{p(n)-1-i}\alpha, g^{p(n)-i}\alpha.$$

Случайные величины  $H_0$  и  $H_{p(n)}$  совпадают со случайными величинами из условия леммы. Поэтому для некоторого  $i$  будет выполнено

$$|\Pr[C_n(H_i) = 1] - \Pr[C_n(H_{i+1}) = 1]| \geq \varepsilon_n/p(n).$$

Это  $i$  зависит от  $n$  и мы будем использовать его в построении схемы отличителя  $h\alpha, g\alpha$  от  $r, \alpha$ . Случайная величина  $H_{i+1}$  получена некоторым преобразованием  $T$  строки  $r_1 r_2 \dots r_i r_{i+1} \alpha$ , вычисляемым схемой размера  $\text{poly}(n)$ . Если это преобразование применить к строке  $r_1 r_2 \dots r_i h\alpha g\alpha$ , то получится случайная величина  $H_i$ . Поэтому схема  $C_n(T(r_1 r_2 \dots r_i r_{i+1} \alpha))$  различает величины  $r_1 r_2 \dots r_i r_{i+1} \alpha$  и  $r_1 r_2 \dots r_i h(\alpha)g(\alpha)$ . Зафиксировав подходящим образом значения  $r_1 \dots r_i$ , мы получим схему, отличающую  $r_{i+1}, \alpha$  и  $h\alpha, g\alpha$ .  $\square$

Совершенно аналогично доказывается аналог этой леммы для односторонних функций с секретом.



**Лемма.** Пусть  $h_n$  является трудным битом для полиномиально вычислимой перестановки с секретом  $g_n^e : D_n^e \rightarrow D_n^e$ . Тогда для любого полинома  $p(n)$  случайные величины

$$e, h^e(\gamma^e), h^e(g^e(\gamma^e)), \dots, h^e((g^e)^{p(n)-1}(\gamma^e)), (g^e)^{p(n)}(\gamma^e)$$

и  $e, r_1, r_2, \dots, r_{p(n)}, \gamma^e$  вычислительно неотличимы.

Итак, мы почти доказали следующее утверждение.

**Теорема 5.** Если существует односторонняя перестановка и трудный бит для нее, то для некоторого полинома  $q(n)$  существует генератор ПСЧ типа  $q(n) \rightarrow \infty$ .

*Доказательство.* Сначала предположим, что  $D_n$  есть множество всех слов длины  $k(n)$ . Тогда в качестве генератора  $q(n)$  можно взять  $k(n)$  и положить

$$G_n(s) = h(s)h(g(s))h(g^2(s)) \dots$$

По Лемме 3.4 это отображение является генератором ПСЧ. Действительно, лемма утверждает, что любое начало этой последовательности полиномиальной длины вычислительно неотлично от равномерно распределенной последовательности той же длины.

В общем случае воспользуемся доступностью равномерного распределения на  $D_n$ . Зафиксируем вероятностный алгоритм  $K$ , который по  $n$  за полиномиальное от  $n$  время генерирует случайную величину, которая статистически неотличима от случайной величины  $\alpha_n$ , равномерно распределенной в  $D_n$ . Пусть  $K$  использует случайную строку  $s$  длины  $q(n)$  и  $K_n(s)$  обозначает выдаваемую им строку. Определим генератор  $G_n$ . Его значение на строке  $s$  длины  $q(n)$  равно

$$G_n(s) = h(K_n(s))h(g(K_n(s)))h(g^2(K_n(s))) \dots$$

В этой последовательности  $l$ -ый бит равен  $h(g^l(K_n(s)))$  и может быть вычислен за полиномиальное от  $l + n$  время.

Докажем, что построенный генератор  $G_n$  надежен. Пусть  $p(n)$  — произвольный полином. Случайная величина  $K_n(s)$  статистически неотличима от случайной величины  $\alpha_n$ . Поэтому и первые  $p(n)$  битов  $G_n(s)$  неотличимы от

$$h(\alpha_n)h(g(\alpha_n))h(g^2(\alpha_n)) \dots h(g^{p(n)-1}(\alpha_n))$$

тестами того же размера.<sup>2</sup> По Лемме 3.4 последняя неотличима от равномерно распределенной последовательности тестами размера  $\text{poly}(n)$ .  $\square$

Генератор, построенный в доказательстве теоремы 5 обладает двумя практически важными свойствами, не отраженным в определении. (а) Биты последовательности  $G_n(s)$  можно вычислять по очереди так, что на вычисление очередного бита уходит время, ограниченное некоторым полиномом от  $n$ , не зависящим от номера этого бита. Действительно, вычислив  $l$  первых битов  $G_n(s)$ , мы сохраняем слово  $g^l(s)$ . Для вычисления очередного бита надо применить к  $g^l(s)$  сначала функцию  $g$ , а потом к полученному слову применить функцию  $h$ . (б) Для любого полинома  $p(n)$  подряд идущих битов (а не только первые) сгенерированной последовательности вычислительно неотличимы от равномерно распределенной последовательности длины  $p(n)$ . Действительно, случайная величина  $g^l(s)$  имеет в точности то же распределение, что и  $s$ . Поэтому подслово  $G_n(s)$ , состоящее из битов с номерами  $l, l+1, \dots, l+p(n)-1$ , имеет в точности то же распределение, что и начало  $G_n(s)$  длины  $p(n)$ .

Замечание. В каком месте доказательства Леммы 3.4 мы использовали то, что  $g_n$  является перестановкой? Ровно в одном месте, а именно, когда использовали вычислительную неотличимость случайных величин  $\alpha_n$  и  $g_n(\alpha_n)$ . Поэтому в условии леммы требование инъективности  $g_n$  можно заменить на более слабое требование: случайные величины  $\alpha_n$  и  $g_n(\alpha_n)$  вычислительно неотличимы (где  $\alpha_n$  равномерно распределена в  $D_n$ ). Более того, совершенно необязательно, чтобы случайная величина  $\alpha_n$  была равномерно распределена в  $D_n$ . Нам нужно, чтобы она была доступна и трудна для  $g_n$ . Будем функции  $g_n$ , для которых существуют доступная трудная случайная величина  $\alpha_n$ , вычислительно неотличимая от  $g_n(\alpha_n)$ , называть *обобщенными перестановками*.

Итак, мы получаем следующие утверждения.

**Лемма.** Пусть функции  $g_n, h_n$  вычислимы за полиномиальное время, причем  $h_n$  принимает значения  $0, 1$ . Пусть случайная величина  $h_n(\alpha_n)$  трудно вычислима по  $g_n(\alpha_n)$ , где  $\alpha_n$  доступная случайная величина, вычислительно неотличимая от  $g_n(\alpha_n)$ . Тогда для любого полинома  $p(n)$  случайные величины

$$h(\alpha), h(g(\alpha)), \dots, h(g^{p(n)-1}(\alpha)), g^{p(n)}(\alpha) \quad \text{и} \quad r_1, r_2, \dots, r_{p(n)}, \alpha$$

<sup>2</sup>В этом доказательстве нам достаточно всего лишь вычислительной неотличимости  $K_n(s)$  и  $\alpha_n$ .

вычислительно неотличимы (для наглядности мы опускаем индекс  $n$ ). Здесь  $r_1 r_2 \dots r_{p(n)}$  случайная равномерно распределенная строка длины  $p(n)$ , независимая от  $\alpha_n$ .

**Теорема 6.** Если существуют полиномиально вычислимые функции  $g_n, h_n$  и доступная случайная величина  $\alpha_n$  такие, что  $h_n(\alpha_n)$  трудно вычислимо по  $g_n(\alpha_n)$  и  $\alpha_n$  вычислительно неотлично от  $g_n(\alpha_n)$ , то для некоторого полинома  $q(n)$  существует генератор ПСЧ типа  $q(n) \rightarrow \infty$ .

Из этой теоремы мы получаем следующее следствие:

**Теорема 7.** Если для некоторой полиномиально вычислимой функции  $k(n)$ , ограниченной сверху полиномом, существует генератор  $G_n$  типа  $k(n) \rightarrow k(n) + 1$ , то для некоторого полинома  $q(n)$  существует генератор ПСЧ типа  $q(n) \rightarrow \infty$ .

*Доказательство.* Обозначим через  $h_n(s)$  первый бит  $G_n(s)$ , а через  $g_n(s)$  остальные  $k(n)$  битов. Мы получили полиномиально вычислимую обобщенную перестановку  $g_n : \{0, 1\}^{k(n)} \rightarrow \{0, 1\}^{k(n)}$  вместе с трудным битом для нее. Осталось воспользоваться Теоремой 6.  $\square$

**Задача 31.** Докажите, что если для некоторого полинома  $q(n)$  существует генератор ПСЧ типа  $q(n) \rightarrow \infty$ , то существует и генератор ПСЧ типа  $n \rightarrow \infty$ .

## 3.5 Отступление: надежность генераторов по Яо

В некоторых приложениях генераторов псевдослучайных чисел нам всего лишь нужно, чтобы невозможно было предсказывать никакой бит сгенерированной последовательности по ее предыдущим битам с вероятностью, существенно отличающейся от  $1/2$  (например, в казино). Генераторы, удовлетворяющие этому требованию называются надежными по Яо.

**Определение.** Пусть имеется последовательность случайных величин  $\alpha^1, \alpha^2, \alpha^3, \dots$  и каждая  $\alpha^n$  принимает значения в в множестве бесконечных последовательностей нулей и единиц. Мы будем говорить, что  $\alpha^n$  неотличима по Яо от равномерно распределенной последовательности,

если для любой последовательности индексов  $i_n \leq \text{poly}(n)$  случайная величина  $\alpha_{i_n+1}^n$  ( $i_n + 1$ -ый бит  $\alpha^n$ ) является трудно вычислима по случайной величине  $(\alpha^n)_{i_n}$  (начало длины  $i_n$  последовательности  $\alpha^n$ ).<sup>3</sup> Генератор типа  $n \rightarrow \infty$  называется *надежным по Яо*, если генерируемая им случайная величина неотличима по Яо от равномерно распределенной последовательности.<sup>4</sup>

Если случайная величина вычислительно неотличима от равномерно распределенной случайной величины, то она неотличима от нее и по Яо. Действительно, по лемме 3.4 случайная величина  $(\alpha^n)_{i_n+1}$  трудно вычислима по случайной величине  $(\alpha^n)_{i_n}$  тогда и только тогда, когда случайные величины  $(\alpha^n)_{i_n+1}$  и  $(\alpha^n)_{i_n}r$  вычислительно неотличимы. А последнее верно, поскольку обе случайные величины неотличимы от случайной величины, от равномерно распределенной среди слов длины  $i_n + 1$ . Верно и обратное.

**Теорема 8.** *Если случайная величина  $\alpha^n$  в множестве бесконечных последовательностей нулей и единиц неотличима по Яо от равномерно распределенной случайной величины, то она вычислительно неотличима от нее. Следовательно, любой надежный по Яо генератор надежен в обычном смысле.*

*Доказательство.* Пусть  $C_n$  произвольная последовательность схем размера  $\text{poly}(n)$ . Нам надо доказать, что  $C_n$  не отличает  $\alpha^n$  от равномерно распределенной случайной величины. Обозначим через  $l(n)$  количество входов  $C_n$ . Фиксируем произвольное  $n$  и для каждого  $0 \leq i \leq l(n)$  рассмотрим случайную величину  $H_i$ , которая получится, если в последовательности  $\alpha^n$  заменить все биты, начиная с  $i + 1$ -ого, на случайные биты, независимые от  $\alpha^n$ . Ясно, что начало  $H_{l(n)}$  длины  $l(n)$  распределено так же, как начало  $\alpha^n$  длины  $l(n)$ , а  $H_0$  равномерно распределена. Обозначим через  $i_n$  тот индекс, для которого  $|\text{Pr}[C_n(H_{i_n}) = 1] - \text{Pr}[C_n(H_{i_n+1}) = 1]|$  максимально. Сравним случайные величины  $H_{i_n}$  и  $H_{i_n+1}$ . У них первые  $i_n$  битов равны  $i_n$  первым битам  $\alpha^n$ . Последние  $n - i_n - 1$  битов выбираются случайно. Разница только в  $i_n + 1$ -ом бите, который в  $H_{i_n}$  выбирается случайным образом, а в  $H_{i_n+1}$  берется из  $\alpha^n$ . Поскольку  $\alpha^n$  неотличима по Яо от равномерно распределенной последовательности,  $\alpha_{i_n+1}^n$  (верхний индекс

<sup>3</sup>Можно дать определение неотличимости по Яо любых двух случайных величин в множестве слов одной длины, но оно нам не понадобится.

<sup>4</sup>Аналогично дается определение надежности по Яо генераторов вида  $k(n) \rightarrow l(n)$ .

### 3.6. ПОСТРОЕНИЕ ФУНКЦИИ, ЯВЛЯЮЩЕЙСЯ ТРУДНЫМ БИТОМ 45

$n$  опускаем) невозможно вычислить по  $(\alpha)_{i_n}$  с вероятностью, существенно отличающейся от  $1/2$ . По лемме 3.4 случайные величины  $(\alpha)_{i_{n+1}}$  и  $(\alpha)_{i_n} r$  вычислительно неотличимы (здесь  $r$  случайный равномерно распределенный бит, независимый от  $\alpha$ ). Значит и случайные величины  $H_{i_n}$  и  $H_{i_{n+1}}$  вычислительно неотличимы (они получаются приписыванием к  $(\alpha)_{i_{n+1}}$  и  $(\alpha)_{i_n} r$ , соответственно, случайной независимой последовательности). Следовательно, величина  $|\Pr[C_n(H_{i_n}) = 1] - \Pr[C_n(H_{i_{n+1}}) = 1]|$  стремится к нулю быстрее любого обратного полинома от  $n$ . Поскольку величина  $|\Pr[H_0 = 1] - \Pr[C_n(H_{l(n)}) = 1]|$  не более, чем в  $l(n)$  раз превосходит эту, то и она стремится к нулю быстрее любого обратного полинома от  $n$ .  $\square$

## 3.6 Построение функции, являющейся трудным битом

### 3.6.1 Код Адамара

Конструкция основана на так называемом коде Адамара. Пусть дано двоичное слово  $x$  длины  $n$ . Будем воспринимать биты слова  $x$ , как коэффициенты линейной функции

$$y_1, \dots, y_n \mapsto x_1 y_1 + \dots + x_n y_n$$

из  $n$ -мерного линейного пространства над полем  $\mathbb{F}_2$  из 2 элементов в  $\mathbb{F}_2$ . *Кодом Адамара* слова  $x$  называется последовательность длины  $2^n$ , состоящая из значений этой линейной функции на *всех* возможных аргументах.

Мы будем использовать способность кода Адамара “исправлять ошибки”. Это означает, что если в коде Адамара  $f$  слова  $x$  изменить не более чем  $(1/2 - \varepsilon)2^n$  символов на противоположные, то по полученной строке  $g$  можно быстро восстановить  $x$ . Точнее, существует вероятностный алгоритм  $\mathcal{B}$ , который за полиномиальное от  $n/\varepsilon$  время находит некоторый список, который с вероятностью не менее  $1/2$  содержит  $x$ . Испорченный код Адамара  $g$  (имеющий длину  $2^n$ ) подаётся на вход алгоритма  $\mathcal{B}$  в виде “оракула” (алгоритм может запросить значение любого бита, указав его номер, то есть слово  $g$  рассматривается как функция  $g: \mathbb{F}_2^n \rightarrow \mathbb{F}_2$ , и алгоритм может её “вызывать”). Это важно, поскольку, если бы  $g$  подавался

на вход в виде слова, то за полиномиальное от  $n/\varepsilon$  время алгоритм смог бы прочесть только его начало небольшой длины. В качестве  $\varepsilon$  мы будем рассматривать только числа вида  $1/k$ , где  $k$  натурально.

**Теорема 9.** *Существует вероятностный алгоритм  $\mathcal{B}$ , который, для любого слова  $x$  длины  $n$  и любого  $\varepsilon$ , получив на вход  $n$  и  $\varepsilon$ , и имея в качестве оракула любую строку  $g$ , отличающуюся от кода Адамара строки  $x$  не более чем в  $(1/2 - \varepsilon)2^n$  позициях, за полиномиальное от  $n/\varepsilon$  время с вероятностью не менее  $1/2$  находит некоторый список слов, содержащий  $x$ .*

*Доказательство.* Итак, в роли алгоритма  $\mathcal{B}$  мы имеем доступ к  $g$  — “искажённому варианту” неизвестной линейной функции

$$f(y_1, \dots, y_n) = x_1 y_1 + \dots + x_n y_n,$$

и должны отгадать коэффициенты  $x_1, \dots, x_n$ . Эти коэффициенты определяются значениями  $f$  в  $n$  точках

$$(1, 0, \dots, 0), (0, 1, \dots, 0), \dots, (0, 0, \dots, 1)$$

(и наоборот), и нам будет удобнее говорить об отгадывании этих значений. (На самом деле конкретный вид точек нам не важен, мы можем отгадывать значения в произвольных  $n$  точках — и даже в произвольном полиномиальном числе точек.)

Поскольку функция  $f$  является линейной, для любых  $y$  и  $r$  из  $\mathbb{W}^n$  выполняется равенство

$$f(y) = f(y + r) - f(r).$$

Пусть  $y$  фиксировано, а  $r$  пробегает  $\mathbb{W}^n$ . Поскольку  $f$  и  $g$  совпадают по крайней мере в  $(\frac{1}{2} + \varepsilon)$ -доле позиций (что больше половины), то

$$f(y) = \text{большинство из } [g(y + r) - f(r)],$$

и это большинство можно определить методом Монте-Карло, выбрав несколько случайных  $r$  и проведя голосование среди выбранных значений. План этот на первый взгляд не имеет смысла, так как в правой части стоит  $f(r)$ , которого мы всё равно не знаем.

### 3.6. ПОСТРОЕНИЕ ФУНКЦИИ, ЯВЛЯЮЩЕЙСЯ ТРУДНЫМ БИТОМ 47

Заметим, что для нахождения большинства по методу Монте-Карло не обязательно, чтобы испытания были по-настоящему независимы. Достаточно (хотя и с худшей оценкой на вероятность ошибки), чтобы они были *попарно* независимы. Именно, имеет место

**Закон больших чисел для попарно независимых событий.** Пусть события  $A_1, \dots, A_N$  попарно независимы и каждое из них имеет вероятность меньше  $\frac{1}{2} - \varepsilon$ . Тогда вероятность того, что произойдёт половина или больше событий, не превосходит

$$\frac{1}{\varepsilon^2} \cdot \frac{1}{N}$$

(и потому мала при больших  $N$ ).

В самом деле, рассмотрим индикаторы этих событий (случайные величины  $a_i$ , равные единице, если  $A_i$  случилось, и нулю в противном случае). Математическое ожидание каждого из индикаторов не больше  $\frac{1}{2} - \varepsilon$ . Поэтому математическое ожидание суммы не больше  $N(\frac{1}{2} - \varepsilon)$ . Теперь воспользуемся попарной независимостью: она гарантирует, что дисперсия суммы  $\sum a_i$  равна сумме дисперсий; дисперсия  $a_i$  не превосходит 1 (на самом деле даже  $1/4$ ), поэтому дисперсия суммы не превосходит  $N$ . Дисперсия есть математическое ожидание квадрата отклонения от математического ожидания. Если произойдёт больше половины событий, то отклонение будет по крайней мере  $N\varepsilon$ , а его квадрат  $N^2\varepsilon^2$ . Поэтому по неравенству Чебышёва вероятность такого события не больше

$$\frac{N}{N^2\varepsilon^2} = \frac{1}{\varepsilon^2} \cdot \frac{1}{N},$$

что и требовалось доказать.

Откуда мы возьмём попарно независимые величины? Если взять  $s$  (полностью) независимых векторов в  $\mathbb{W}^n$ , то их частичные суммы (всего  $2^s - 1$  векторов) будут попарно независимы. Например, если  $u_1, u_2$  и  $u_3$  — (полностью) независимые векторы, то семь векторов

$$u_1, u_2, u_3, u_1 + u_2, u_1 + u_3, u_2 + u_3, u_1 + u_2 + u_3$$

будут попарно независимы. (Скажем, при данном значении  $u_1 + u_2$  все значения  $u_1 + u_3$  равновероятны, поскольку  $u_3$  не зависит от пары  $(u_1, u_2)$ .)

Геометрически: на  $s$  случайных независимых равномерно распределённых векторов мы натягиваем параллелепипед, и  $2^s - 1$  его вершин (не считая нулевой) будут попарно независимы.

Теперь уже можно объяснить идею доказательства: мы положим

$$f(y) = \text{большинство из } [g(y + r_i) - f(r_i)],$$

где  $r_i$  — попарно независимые векторы ( $2^s - 1$  штук), а именно, ненулевые вершины параллелепипеда, натянутого на (полностью) независимые векторы  $u_1, \dots, u_s$ . Чтобы получить малую вероятность ошибки, достаточно взять полиномиальное число попарно независимых векторов, то есть  $2^s - 1 = \text{poly}(n/\varepsilon)$ , а тогда  $s = O(\log n/\varepsilon)$ .

Теперь главное: все значения  $f(r_i)$  в силу линейности  $f$  определяются значениями  $f(u_1), \dots, f(u_s)$ , то есть  $s$  битами. Возможных вариантов этих значений  $2^s$ , то есть полиномиальное от  $n/\varepsilon$  количество. Вспомним, что нам разрешалось отгадывать с нескольких попыток: алгоритм  $\mathcal{B}$  выдаёт не один набор коэффициентов, а полиномиальный (от  $n/\varepsilon$ ) список вариантов. Так что нам позволено перепробовать эти варианты, и всё сходится.

Оценим возникающие вероятности более подробно. Нам нужно восстановить значения  $f$  в  $n$  точках. Мы начинаем с того, что выбираем независимые векторы  $u_1, \dots, u_s$  и получаем из них попарно независимые векторы  $r_1, \dots, r_{2^s-1}$ . Затем для каждой из  $n$  точек мы проводим голосование с  $2^s - 1$  участниками и получаем набор из  $n$  предполагаемых значений. Эта процедура предполагает известными значения  $f(u_1), \dots, f(u_s)$  и проводится  $2^s$  раз, для всех возможных вариантов. Подчеркнём, что выбор случайных векторов  $u_1, \dots, u_s$  производится только один раз: одни и те же векторы используются для всех  $n$  точек и во всех  $2^s$  попытках.

Для данной точки  $y$  события  $f(y + r_i) \neq g(y + r_i)$  (всего  $2^s - 1$  событий при  $i = 1, 2, \dots, 2^s - 1$ ) являются попарно независимыми. Вероятность каждого из них не больше  $\frac{1}{2} - \varepsilon$ . Поэтому вероятность того, что выборка окажется “плохой” (не меньше половины попаданий в ошибки, где  $f \neq g$ ), не превосходит

$$\frac{1}{\varepsilon^2} \cdot \frac{1}{2^s - 1}.$$

Это — для данной точки  $y$ ; вероятность того, что выборка окажется плохой хотя бы для одной из  $n$  точек, не больше

$$\frac{1}{\varepsilon^2} \cdot \frac{1}{2^s - 1} \cdot n$$

и потому не превосходит  $1/2$ , если

$$2^s \geq \frac{2n}{\varepsilon^2} + 1.$$



### 3.6. ПОСТРОЕНИЕ ФУНКЦИИ, ЯВЛЯЮЩЕЙСЯ ТРУДНЫМ БИТОМ 49

А если выборка хороша для всех  $n$  точек, то в список, выдаваемый алгоритмом  $\mathcal{B}$  (и содержащий  $2^s$  вариантов), войдет и правильный ответ (коэффициенты аффинной функции  $f$ ). Длина списка и время работы алгоритма  $\mathcal{B}$  при этом полиномиальны от  $n/\varepsilon^2 = \text{poly}(n/\varepsilon)$ .  $\square$

*Задача 32.* Постройте вероятностный алгоритм, который для любого слова  $x$  длины  $n$  и любых  $\varepsilon, \delta$ , получив на вход  $n, \varepsilon, \delta$  и имея в качестве оракула любую строку  $g$ , отличающуюся от кода Адамара строки  $x$  не более чем в  $(1/4 - \varepsilon)2^n$  позициях (мы заменили  $1/2$  на  $1/4$ ), за полиномиальное от  $n, 1/\varepsilon, 1/\delta$  время с вероятностью не менее  $1 - \delta$  находит само слово  $x$  (а не список, содержащий  $x$ , как в теореме).

*Задача 33.* Докажите, что в предыдущей задаче в оценке времени работы алгоритма можно заменить  $1/\delta$  на  $\log(1/\delta)$ , не меняя всего остального.

#### 3.6.2 Конструкция

Пусть  $f_n : D_n \rightarrow D_n$  односторонняя перестановка, и  $D_n \subset \{0, 1\}^{l(n)}$ . Рассмотрим функции  $g_n, h_n$  определенные на множестве  $D_n \times \{0, 1\}^{l(n)}$  равенствами  $g_n(xy) = f_n(x)y$  и  $h_n(xy) = x \odot y$ . Здесь выражение  $xy$  обозначает конкатенацию слов  $x, y$  длины  $l(n)$ , а  $x \odot y = \sum_{i=1}^{l(n)} x_i y_i \pmod 2$ . Ясно, что функции  $g_n$  и  $h_n$  полиномиально вычислимы и  $g_n$  является перестановкой. Теорема Гольдрайха—Левина утверждает, что  $h_n$  является трудным битом для нее.

**Теорема 10** (Гольдрайха—Левина). *Если  $f_n$  сильно необратима и проверяема, то функция  $x \odot y$  является трудным битом для функции  $g_n(x, y) = f_n(x)y$ .*

*Доказательство.* Допустим противное — для бесконечно многих  $n$  существует схема  $C_n$  размера  $\text{poly}(n)$  такая, что вероятность события  $C_n(f(x)y) = x \odot y$  не меньше  $1/2 + \varepsilon_n$ , где  $\varepsilon_n = 1/\text{poly}(n)$  (слово  $x$  выбирается равномерно из  $D_n$ , слово  $y$  выбирается равномерно из  $\{0, 1\}^{l(n)}$  независимо от  $x$ ). Зафиксируем некоторое  $n$  из этого бесконечного множества и построим вероятностную схему полиномиального от  $n$  размера, которая будет обращать  $f(x)$  с вероятностью не меньше  $\varepsilon_n/4$ . Эта схема работает так: она запускает алгоритм из теоремы 9, используя в качестве внешней процедуры  $g(y) = C_n(f(x)y)$ , и  $\varepsilon_n/2$  в качестве  $\varepsilon$ . Затем для каждой строки  $\tilde{x}$  из списка, выданного алгоритмом, схема проверяет равенство  $f(\tilde{x}) = f(x)$  (здесь используется условие проверяемости функции  $f$ ) и

выдает первое  $\tilde{x}$ , для которого  $f(\tilde{x}) = f(x)$ . Если такого  $\tilde{x}$  в списке не нашлось, то схема выдает, что угодно.

Оценим вероятность, с которой эта схема правильно обращает  $f(x)$ . Обозначим через  $p(x)$  вероятность события  $C_n(f(x)y) = x \odot y$  для случайно выбранного  $y$ . По теореме 9 схема обратит  $f(x)$  с вероятностью не меньше  $1/2$  для любого такого  $x$ , что

$$p(x) \geq 1/2 + \varepsilon_n/2.$$

Поэтому достаточно доказать, что для случайного  $x \in D_n$  это неравенство имеет место с вероятностью не меньше  $\varepsilon_n/2$ .

Это легко доказать от противного. Действительно, по условию среднее значение величины  $p(x)$  не меньше  $1/2 + \varepsilon_n$ . Разделим все строки из  $D_n$  на “хорошие”, для которых

$$p(x) \geq 1/2 + \varepsilon_n/2,$$

и “плохие”, для которых это неверно. Если бы  $x$  было хорошим с вероятностью меньше  $\varepsilon_n/2$ , то среднее значение величины  $p(x)$  было бы меньше

$$(\varepsilon_n/2) \cdot 1 + 1 \cdot (1/2 + \varepsilon_n/2) = 1/2 + \varepsilon_n$$

(первое слагаемое в этой сумме ограничивает сверху вклад в среднее хороших строк, второе слагаемое — вклад плохих строк). Теорема доказана.  $\square$

Из этой теоремы легко получается следующее следствие.

**Лемма.** *Если существует односторонняя проверяемая перестановка  $f^e$  с секретом, то существует и односторонняя проверяемая перестановка  $g^e$  с секретом и трудный бит  $h^e$  для нее.*

*Кроме того, если  $f^e$  является улучшенной односторонней проверяемой перестановкой с секретом, то  $h^e((f^e)^{-1}(y))$  трудно вычислимо по  $e$  и случайным битам, использованным при генерации  $y$ .*

*Доказательство.* Пусть  $f^e$  исходная односторонняя перестановка с секретом. В новой односторонней перестановке случайная величина  $\langle e_n, d_n \rangle$  та же самая, а функция  $g^e$  определяется как  $g^e(xy) = f^e(x)y$ , где  $y$  строка из нулей и единиц той же длины, что и  $x$  (мы опускаем параметр безопасности  $n$  в обозначениях). Таким образом область определения  $g^e$  состоит

### 3.6. ПОСТРОЕНИЕ ФУНКЦИИ, ЯВЛЯЮЩЕЙСЯ ТРУДНЫМ БИТОМ 51

из всех слов вида  $xy$ , где  $x \in D^e$ ,  $|y| = |x|$ . Трудный бит  $h^e$  определяется как скалярное произведение:  $h^e(xy) = x \odot y$ .

Условия (а)–(г) очевидно выполнены для  $g^e$ . Докажем, что  $h^e$  трудный бит к  $g^e$ . Предположим, что схема  $C_n$  полиномиального от  $n$  размера с вероятностью  $1/2 + \varepsilon_n$  по  $e$ ,  $f^e(x)$ ,  $y$  вычисляет  $x \odot y$ . Тогда ее можно переделать в схему полиномиального размера, которая по  $e$ ,  $f^e(x)$ ,  $y$ ,  $z$  (где  $z$  равномерно распределено среди слов той же длины, что  $e$ , и независимо от  $x$ ,  $y$ ,  $e$ ) вычисляет с вероятностью  $1/2 + \varepsilon_n$  скалярное произведение конкатенации  $ex$  и  $zy$ , поскольку последнее равно  $e \odot z \oplus z \odot y$ . Из условия (б) следует, что указанное распределение на тройках  $\langle e, x, y \rangle$  является трудным для функции  $(e, x, y, z) \mapsto (e, g^e(x), y, z)$ . По теореме 11  $\varepsilon_n$  пренебрежимо мало.  $\square$

*Замечание.* В доказательстве теоремы мы не использовали то, что трудное распределение для  $g_n$  является равномерным. Таким образом, верно следующее обобщение теоремы Голдрайха–Левина и ее следствие.

**Теорема 11.** Пусть  $\alpha_n$  доступная трудная случайная величина для проверяемой функции  $g_n$ . Тогда случайная величина  $x \odot y$  трудно вычислима по случайной величине  $g_n(x)y$ , где  $x$  распределено так же, как  $\alpha_n$ , а  $y$  независимо от  $x$  и распределено равномерно среди слов той же длины, что и  $\alpha_n$ .

**Теорема 12.** Если существует обобщенная односторонняя перестановка, то для некоторого полинома  $q(n)$  существует генератор ПСЧ типа  $q(n) \rightarrow \infty$ .

*Доказательство.* Пусть даны перестановка  $g_n$  множества  $D_n$  и трудная для нее случайная величина  $\alpha_n$ , вычислительно неотличимая от  $g_n(\alpha_n)$ . Рассмотрим функцию  $(x, y) \mapsto (g_n(x), y)$ , где  $x \in D_n$ , а  $y$  произвольное слово той же длины, что и  $x$ . Будем выбирать  $x$  случайно по распределению  $\alpha_n$ , а  $y$  равномерно и независимо от  $x$ . Такое распределение на парах  $(x, y)$  доступно, и по предыдущей теореме  $x \odot y$  трудно вычислимо по  $(g_n(x), y)$ . Кроме того,  $(x, y)$  вычислительно неотлично от  $(g_n(x), y)$ . Осталось применить Теорему 6.  $\square$

### 3.6.3 Трудный бит для односторонней перестановки с секретом

**Лемма.** *Если существует односторонняя проверяемая перестановка с секретом, то существует и односторонняя проверяемая перестановка и секретом и трудный бит для нее.*

*Доказательство.* Это следует из теоремы Левина–Голдрайха. Пусть  $f^e$  исходная односторонняя перестановка с секретом. В новой односторонней перестановке случайная величина  $\langle e_n, d_n \rangle$  та же самая, а функция  $g^e$  определяется как  $g^e(xy) = f^e(x)y$ , где  $y$  строка из нулей и единиц той же длины, что и  $x$  (мы опускаем параметр безопасности  $n$  в обозначениях). Таким образом область определения  $g^e$  состоит из всех слов вида  $xy$ , где  $x \in D^e$ ,  $|y| = |x|$ . Трудный бит  $h^e$  определяется как скалярное произведение:  $h^e(xy) = x \odot y$ .

Условия (а)–(г) очевидно выполнены для  $g^e$ . Докажем, что  $h^e$  трудный бит к  $g^e$ . Предположим, что схема  $C_n$  полиномиального от  $n$  размера с вероятностью  $1/2 + \varepsilon_n$  по  $e, f^e(x), y$  вычисляет  $x \odot y$ . Тогда ее можно переделать в схему полиномиального размера, которая по  $e, f^e(x), y, z$  (где  $z$  равномерно распределено среди слов той же длины, что  $e$ , и независимо от  $x, y, e$ ) вычисляет с вероятностью  $1/2 + \varepsilon_n$  скалярное произведение конкатенации  $ex$  и  $zy$ , поскольку последнее равно  $e \odot z \oplus z \odot y$ . Из условия (б) следует, что указанное распределение на тройках  $\langle e, x, y \rangle$  является трудным для функции  $(e, x, y, z) \mapsto (e, g^e(x), y, z)$ . По теореме 11  $\varepsilon_n$  пренебрежимо мало.  $\square$

## Глава 4

# Псевдослучайные функции

Пусть фиксированы многочлены  $k(n), l(n)$ . Будем рассматривать функции из  $\{0, 1\}^{k(n)}$  в  $\{0, 1\}^{l(n)}$  с равномерным распределением на них. Всего таких функций  $(2^{l(n)})^{2^{k(n)}}$  и каждая из них задается  $l(n)2^{k(n)}$  битами. Поэтому для выбора случайной функции нам нужно  $l(n)2^{k(n)}$  случайных битов. Наша цель — уменьшить количество случайных битов до некоторого многочлена от  $n$  за счет небольшого ухудшения “степени случайности”. Точнее, мы хотим построить семейство функций  $f_s^n : \{0, 1\}^{k(n)} \rightarrow \{0, 1\}^{l(n)}$ , где  $s \in \{0, 1\}^{m(n)}$  и  $m(n)$  — некоторый многочлен, обладающее следующими двумя свойствами.

(а) По  $s, x$  и  $n$  можно найти  $f_s^n(x)$  за полиномиальное от  $n$  время (по идентификатору функции и ее аргументу можно быстро вычислить её значение).

Второе свойство утверждает, что для любого множества слов полиномиальной от  $n$  мощности при случайном выборе  $s$  из  $\{0, 1\}^{m(n)}$  значения функции  $f_s^n$  на строках этого множества вычислительно неотличимы от значений случайно выбранной функции

$$g : \{0, 1\}^{k(n)} \rightarrow \{0, 1\}^{l(n)}.$$

Будем задавать множество из не более чем  $p(n)$  слов длины  $k(n)$  последовательностью  $x$  длины  $k(n)p(n)$ . Такую последовательность можно разбить на  $p(n)$  блоков  $x_1, \dots, x_{p(n)}$  длины  $k(n)$ , которые и будут задавать множество.

(б) Для любого полинома  $p(n)$  и любой последовательности строк  $\{x^n\}$ , где длина  $x^n$  равна  $k(n)p(n)$ , случайные величины

$$f_s^n(x_1^n) \dots f_s^n(x_{p(n)}^n) \text{ и } g(x_1^n) \dots g(x_{p(n)}^n) \quad (4.1)$$

вычислительно неотличимы. Здесь  $s$  выбирается случайно среди всех слов длины  $m(n)$  с равномерным распределением, а  $g$  выбирается случайно среди всех функций из  $\{0, 1\}^{k(n)}$  в  $\{0, 1\}^{l(n)}$  с равномерным распределением.

В этом требовании строки  $x_1^n, \dots, x_{p(n)}^n$  не предполагаются попарно различными. Однако без ограничения общности можно ограничиться только этим случаем. Действительно, если какие-то два слова, скажем  $x_1^n, x_2^n$ , совпадают, то и значения  $f_s^n$  и  $g$  на них совпадают. Поэтому слово  $x_1^n$  можно удалить из последовательности  $x_1^n, \dots, x_{p(n)}^n$ : неотличимость случайных величин (4.1) для полученной последовательности влечет неотличимость случайных величин (4.1) для исходной последовательности. Поэтому для доказательства свойства (б) для повторяющихся слов можно вычеркнуть все повторения и воспользоваться свойством (б) для полученной последовательности из попарно различных слов.

Заметим, что если  $x_1^n, \dots, x_{p(n)}^n$  попарно различны, то  $g(x_1^n) \dots g(x_{p(n)}^n)$  есть просто равномерно распределенная строка длины  $p(n)k(n)$ . Поэтому свойство (б) можно переформулировать так:

для любого полинома  $p(n)$  и любой последовательности последовательностей из  $p(n)$  попарно различных строк  $x_1^n, \dots, x_{p(n)}^n$  длины  $k(n)$  случайная величина

$$f_s^n(x_1^n) \dots f_s^n(x_{p(n)}^n)$$

вычислительно неотличима от равномерно распределенной строки длины  $p(n)k(n)$ .

Псевдослучайные функции можно понимать и как генераторы псевдослучайных чисел вида  $m(n) \rightarrow 2^{k(n)}l(n)$ . При этом тестирующему генератор алгоритму доступны любые символы выходной последовательности в полиномиальном количестве. Поэтому возникает естественная идея построения семейства ПСФ. Пусть нам дан генератор  $G_n : \{0, 1\}^{m(n)} \rightarrow \{0, 1\}^\infty$ , где  $m(n)$  — некоторый многочлен. Разрежем  $G_n(s)$  на блоки длины  $l(n)$ , занумеруем первые  $2^{k(n)}$  из них двоичными последовательностями длины  $k(n)$  и положим  $f_s^n(x)$  равным  $x$ -ому из полученных блоков.

Однако эта конструкция не удовлетворяет ни первому, ни второму требованию. Первое не выполнено, поскольку мы не можем по  $n, x, s$  за полиномиальное от  $n$  время найти  $x$ -ый блок  $G_n(s)$  (определение генератора гарантирует только возможность найти его за экспоненциальное от  $n$  время). Второе не выполнено, поскольку определение генератора гарантирует вычислительную неотличимость только начала полиномиаль-

ной длины выходной последовательности от равномерно распределённой последовательности.

Тем не менее, генераторы ПСЧ можно использовать для построения семейства ПСФ. Для этого надо действовать несколько другим способом.

**Теорема 13.** *Если существует генератор ПСЧ  $G_n : \{0, 1\}^n \rightarrow \{0, 1\}^{2n}$ , то для любых полиномов  $k(n), l(n)$  существует семейство ПСФ  $f_s^n : \{0, 1\}^{k(n)} \rightarrow \{0, 1\}^{l(n)}$ .*

*Доказательство.* Предположим сначала, что  $l(n) \geq n$ . Тогда отображение  $G_{l(n)} : \{0, 1\}^{l(n)} \rightarrow \{0, 1\}^{2l(n)}$  является генератором ПСЧ.

Идентификатор случайной функции  $s$  будет иметь длину  $l(n)$ . Пусть  $s$  произвольная строка длины  $l(n)$  и  $x = x_1 \dots x_k$  произвольное слово длины  $k(n)$ . Значение  $f_s(x)$  определяется так. Разрежем слово  $G(s)$  на две половины, и обозначим их через  $s_0$  и  $s_1$ . Оставим первую половину, если  $x_1 = 0$  и вторую половину, если  $x_1 = 1$ . Опять применим к полученному слову  $s_{x_1}$  отображение  $G$  и разрежем полученное слово на две части. Обозначим их через  $s_{x_1 0}, s_{x_1 1}$ . Теперь опять оставим только слово  $s_{x_1 x_2}$  и так далее. На каждом шаге мы применяем к текущему слову  $s_w$  отображение  $G$  и оставляем левую половину, если очередной бит слова  $x$  равен нулю, и вторую половину, иначе.

В результате мы получим некоторое слово  $s_x$ , которое и будет значением  $f_s$  на  $x$ . Очевидно, что  $f_s(x)$  можно найти за полиномиальное время. Проверим условие (б).

Пусть  $x_1, \dots, x_{p(n)}$  последовательность из  $p(n)$  различных слов длины  $k(n)$ . Нам нужно доказать вычислительную неотличимость  $f_s(x_1) \dots f_s(x_{p(n)})$  от равномерно распределённой строки длины  $p(n)k(n)$ .

Для этого рассмотрим новое семейство функций  $\tilde{f}_{t_0 t_1}$ , каждая функция в котором задается строкой  $t_0 t_1$  длины  $2l$  (а не длины  $l$ , как раньше). Новое семейство будем определять в точности, как и раньше, за исключением первого шага в рекуррентном определении  $s_w$  (для слов  $w$  длины не более  $k$ ). А именно, слова  $s_0, s_1$  теперь положим равными  $t_0, t_1$ , соответственно. Слово  $s_\Lambda$  (где  $\Lambda$  обозначает пустое слово) теперь не определено. Мы утверждаем, что случайные величины  $f_s(x_1) \dots f_s(x_{p(n)})$  и  $\tilde{f}_{t_0 t_1}(x_1) \dots \tilde{f}_{t_0 t_1}(x_{p(n)})$  вычислительно неотличимы. Действительно, если бы они были отличимы схемой  $C$ , то эту схему можно было бы использовать для отличения случайных величин  $G(s) = s_0 s_1$  и  $t_0 t_1$ .

Тот же трюк можно проделать еще раз: в определении  $\tilde{f}_{t_0, t_1}$  можно заменить  $s_{00}, s_{01}$  на независимые равномерно распределенные слова длины

$l$  (слово  $s_0$  после этой замены становится неопределенным). Затем можно заменить, например,  $s_{010}, s_{011}$  на независимые равномерно распределенные слова и так не более, чем полином раз. На очередном шаге мы можем сделать следующее: если в определении текущего семейства  $\tilde{f}_t$  слово  $s_w$  выбирается чисто случайным образом (как подслово строки  $t$ , задающей функцию), мы можем заменить способ определения  $s_{w0}, s_{w1}$ : вместо того, чтобы применять к  $s_w$  генератор  $G$ , мы можем взять случайные и независимые строки. При этом количество битов в  $t$  увеличится на  $l$ . При каждом изменении семейства  $\tilde{f}_t$  случайная величина  $\tilde{f}_t(x_1) \dots \tilde{f}_t(x_{p(n)})$  будет изменяться незначительно: измененная случайная величина будет вычислительно неотличима от исходной.

Если проделать это не более чем полином раз, то для результирующего семейства  $\tilde{f}_t$  случайная величина  $\tilde{f}_t(x_1) \dots \tilde{f}_t(x_{p(n)})$  будет вычислительно неотличима от исходной случайной величины  $f_s(x_1) \dots f_s(x_{p(n)})$ . Применим это преобразование для всех начал  $w$  слова  $x_1$  (в порядке возрастания длины  $w$ ), затем для всех начал слова  $x_2$ , которые не являются началами  $x_1$ , и так далее. Всего мы произведем не более  $k(n)p(n)$  изменений. Полученная случайная величина  $\tilde{f}_t(x_1) \dots \tilde{f}_t(x_{p(n)})$  вычислительно неотличима от случайной величины  $f_s(x_1) \dots f_s(x_{p(n)})$ . Осталось заметить, что  $\tilde{f}_t(x_1) \dots \tilde{f}_t(x_{p(n)})$  имеет равномерное распределение, поскольку для всех  $i$  слово  $\tilde{f}_t(x_i)$  по построению является подсловом  $t$  и при разных  $i$  эти подслова не перекрываются.

Осталось построить ПСФ для полиномов  $l(n) < n$ . Для этого построим сначала ПСФ  $f_s^n : \{0, 1\}^{k(n)} \rightarrow \{0, 1\}^n$ , а затем модифицируем его, обрезав в  $f^n(x)$  лишние  $n - l(n)$  битов.  $\square$

Можно усилить второе требование к семейству ПСФ, разрешив слову  $x_2^n$  зависеть от  $g(x_1^n)$ , слову  $x_3^n$  зависеть от  $g(x_1^n), g(x_2^n)$  и так далее. То есть, разрешим алгоритму, тестирующему псевдослучайную функцию, запрашивать ее значение на произвольных аргументах, причем очередной аргумент можно выбрать после того, как алгоритм узнал значение функции на предыдущих аргументах. Для аккуратного определения таких тестеров, нам понадобятся алгоритмы с оракулом. Так называют алгоритмы, которые в любой момент своей работы может запрашивать значение внешней функции  $g$ , аргументами и значениями которой являются бинарные слова. Пусть имеется полиномиальный алгоритм с оракулом  $A$  и некоторая последовательность слов  $\{a_n\}$ , в которой длина  $n$ -ого слова ограничена полиномом от  $n$ . Подадим на вход алгоритму  $A$  пару



$1^n, a_n$ , а в качестве внешней функции дадим произвольную функцию

$$g : \{0, 1\}^{k(n)} \rightarrow \{0, 1\}^{l(n)}.$$

Результат работы алгоритма будем обозначать через  $A^g(1^n, a_n)$ .

Пару, состоящую из  $A$  и последовательности  $\{a_n\}$ , будем называть полиномиальным алгоритмом с оракулом и советом полиномиальной длины. Усиленное требование (б) звучит так: *для любого полиномиального алгоритма с оракулом и советом полиномиальной длины  $A, \{a_n\}$  вероятность события  $A^{f_s}(1^n, a_n) = 1$  приблизительно равна вероятности события  $A^g(1^n, a_n) = 1$ .* Здесь  $s$  выбирается с равномерным распределением среди строк длины  $t(n)$ , а  $g$  выбирается с равномерным распределением среди всех функций из  $\{0, 1\}^{k(n)}$  в  $\{0, 1\}^{l(n)}$ .

*Замечание 1.* Заметим, что это требование и в самом деле сильнее: существует семейство функций  $f_s$ , удовлетворяющие требованию (б), но не удовлетворяющее усиленному требованию (б). Пусть, например, функция  $f_s : \{0, 1\}^n \rightarrow \{0, 1\}^n$  выбирается так. Выберем случайное слово  $v$  и положим  $f_s(00 \dots 0) = v$  и  $f_s(v) = 00 \dots 0$ . Значение на всех остальных словах выберем случайно и независимо. Тогда для всех  $s$  выполнено равенство  $f_s(f_s(00 \dots 0)) = 00 \dots 0$ . Это равенство можно проверить полиномиальным алгоритмом с оракулом. При этом, для случайной функции  $g$  равенство  $g(g(00 \dots 0)) = 00 \dots 0$  выполнено с пренебрежимо малой вероятностью  $2^{-n} + (2^n - 1)2^{-2n}$  (первое слагаемое есть вероятность того, что  $g(00 \dots 0) = 00 \dots 0$ , а вторая — вероятность того, что  $g(00 \dots 0) = v$  и  $g(v) = 00 \dots 0$  для некоторого  $v$ , состоящего не из одних нулей). Таким образом, усиленное требование не выполнено.

С другой стороны, для любой последовательности различных слов  $u_1, \dots, u_{\text{poly}(n)}$  длины  $n$  слово  $v$  с приблизительно единичной вероятностью отлично от всех слов  $u_1, \dots, u_{\text{poly}(n)}$ . Поэтому распределение случайной величины  $f_s(u_1), \dots, f_s(u_{\text{poly}(n)})$  статистически неотличимо от равномерного распределения.

**Теорема 14.** *Построенное в доказательстве предыдущей теоремы семейство ПСФ  $f_s^n : \{0, 1\}^{k(n)} \rightarrow \{0, 1\}^{l(n)}$  удовлетворяет этому усиленному требованию.*

*Доказательство.* Пусть дан полиномиальный алгоритм с оракулом  $A$  и последовательность слов  $a_n$  полиномиальной длины. Нам нужно доказать, что вероятность события  $A^{f_s} = 1$  приблизительно равна вероятности события  $A^g = 1$ , где  $s$  случайная строка, а  $g$  — случайная функция.

(Для простоты обозначений мы опускаем вход  $(1^n, a_n)$  алгоритма  $A$ ). Без ограничения общности мы можем считать, что очередной вопрос алгоритма к оракулу всегда отличается от предыдущих вопросов. При таком предположении вероятность второго события есть вероятность ответа 1, если на каждый вопрос к оракулу выдавать случайный ответ (с равномерным распределением), не зависящий от предыдущих ответов. Нам нужно доказать, что и первая вероятность примерно равна тому же самому.

Для этого мы будем действовать примерно так же, как в предыдущем доказательстве. А именно, найдем первое слово  $x_1$ , значение внешней функции на котором запрашивает алгоритм  $A$  (при работе на входе  $(1^n, a_n)$ ). Это слово не зависит от внешней функции. Затем применим к исходному семейству функций  $f_s$  преобразование из предыдущего доказательства для всех собственных начал  $w$  слова  $x_1$ . В результате этого преобразования возникнет новое семейство функций, которое мы будем также обозначать  $f_s$  (длина слова  $s$  увеличится на  $n^2$  по сравнению с исходной длиной). Заметим, что по построению слово  $f_s(x_1)$  есть некоторая подстрока строки  $s$ . Теперь запустим алгоритм  $A$  (на входе  $1^n, a_n$ ) и на первый запрос к внешней процедуре выдадим ответ  $f_s(x_1)$ . Обозначим через  $x_2$  второе слово, значением внешней процедуры на котором запрашивает  $A$ . В отличие от  $x_1$ , слово  $x_2$  зависит от  $s$ . И опять применим преобразование из предыдущего доказательства для всех собственных начал  $w$  слова  $x_2$ , которые не являются началом  $x_1$ . После этого мы получим новое семейство  $f_s$ . Длина слова  $s$  опять увеличится по сравнению с исходной, причем необходимое увеличение зависит от  $f_s(x_1)$ . Например, если  $x_2$  отличается от  $x_1$  только в последнем бите, то увеличивать  $s$  вообще не нужно, поскольку  $x_2$  в этом случае не имеет собственных начал, не являющихся началами  $x_1$ . А если  $x_2$  отличается от  $x_1$  уже в первом символе, то длина  $s$  увеличивается на  $n^2 - n$ . Удобно считать, что в любом случае мы увеличиваем длину  $s$  на эту величину, а лишние символы  $s$  просто не используются. Заметим, что опять  $f_s(x_2)$  равно некоторой подстроке строки  $s$ , не перекрывающейся со строкой, равной  $f_s(x_1)$ . Используя его, определим  $x_3$  и так далее. При каждом изменении семейства  $f_s$  вероятность события  $A^{f_s} = 1$  меняется незначительно. А для заключительного семейства  $f_s$  вероятность события  $A^{f_s} = 1$  совпадает с вероятностью события  $A^g = 1$  для случайно выбранной функции  $g$ . Действительно, как мы отмечали,  $f_s(x_i)$  равно некоторой подстроке строки  $s$ , не перекрывающейся с до этого использованными подстрока-

ми  $x_1, \dots, x_{i-1}$ , а значит является случайной строкой, не зависящей от значения  $f_s$  на  $x_1, \dots, x_{i-1}$ .  $\square$

[Это можно переписать, заменив полиномиальные алгоритмы с оракулом на полиномиальные схемы с элементами для тестируемой функции.]

В описанной конструкции длина идентификатора случайной функции  $s$  равна длине значений  $l(n)$  случайной функции. Немного изменив конструкцию можно добиться того, чтобы длина идентификатора не зависела от длины значений и была равна, скажем,  $n$ . Например, можно применить генератор ПСЧ к исходному идентификатору  $s$  длины  $n$ , получить псевдослучайное слово  $s'$  длины  $l(n)$  и затем применить уже построенную функцию  $f_{s'}$ . Любой тестер полиномиального размера не сможет отличить новое семейство от старого, поскольку иначе мы смогли бы отличить случайный идентификатор от псевдослучайного.

Можно действовать и по-другому: скажем, для построения семейства ПСФ  $f_s^n : \{0, 1\}^{k(n)} \rightarrow \{0, 1\}^{n^{10}}$  с идентификатором  $s$  длины  $n$ , сначала построим семейство ПСФ  $f_s^n : \{0, 1\}^{k(n)+9 \log n} \rightarrow \{0, 1\}^n$ , затем объединим  $n^9$  подряд идущих значений функции в одно значение. То есть, положим

$$g_s^n(x) = f_s^n(x \underbrace{00 \dots 0}_{9 \log n}) f_s^n(x \underbrace{00 \dots 1}_{9 \log n}) \dots f_s^n(x \underbrace{11 \dots 1}_{9 \log n}).$$

[Добавить построение псевдослучайных функций, определённых на множестве всех слов.]

**Задача 34.** Нужно зарегистрировать паспортные данные всех желающих голосовать так, чтобы каждый мог проголосовать только один раз. При этом требуется, чтобы база данных избирательной комиссии этих данных не разглашала. Точнее, надо построить вероятностный полиномиальный алгоритм  $K$  и детерминированный полиномиальный алгоритм  $B$  с такими свойствами. Алгоритм  $K$  получает на вход параметр безопасности  $n$  в унарной записи и генерирует ключ  $k$  (хранящийся в избирательной комиссии). Алгоритм  $B$  получает на вход параметр безопасности, ключ  $k$  и паспортные данные (строку)  $x$  и генерирует элемент базы данных (другую строку)  $y$ . Требование безопасности заключается в следующем: для любого полиномиального количества различных строк  $x_1, \dots, x_m$  полиномиальной длины (от параметра безопасности) строка  $B(n, k, x_1) \dots B(n, k, x_m)$  имеет распределение, неотличимое от некоторо-

го полиномиально моделируемого распределения. Требование возможности проголосовать: для любого полиномиального количества различных строк  $x_1, \dots, x_m$  полиномиальной длины (от параметра безопасности) пренебрежимо мала вероятность совпадения каких-то из двух значений  $B(n, k, x_1) \dots B(n, k, x_m)$ . Постройте такие алгоритмы. (Похожая проблема возникла при выборах в Координационный совет российской оппозиции в 2012 году.)

## Глава 5

# Шифрование с закрытым ключом

### 5.1 Одноразовые схемы

Схема шифрования с закрытым ключом (в другой терминологии “симметричная схема”) состоит из полиномиально моделируемой последовательности  $e_n$  случайных величин и вероятностных полиномиальных алгоритмов  $E, D$ , называемых алгоритмами *шифровки* и *расшифровки*. Случайная величина  $e_n$  называется *ключом*,

Алгоритмы  $E$  и  $D$  получают на вход натуральное число  $n$  в унарной записи и пару двоичных строк и выдают на выход одну двоичную строку. А именно, алгоритм  $E$  получает на вход параметр безопасности  $n$ , ключ  $e_n$  и сообщение  $x \in \{0, 1\}^*$  и выдает некоторую строку  $y = E_n(e_n, x)$ , называемую *шифrogramмой*, длина которой зависит только от  $n$  и длины  $x$ . Алгоритм  $D$  получает на вход  $n$ ,  $y$  и  $e_n$  и выдает строку  $D_n(e_n, y)$ .

Мы хотим, чтобы при любом  $x$   $D(e_n, E_n(e_n, x)) = x$  с примерно единичной вероятностью, причем при случайно выбранном ключе  $e_n$  шифrogramма  $E_n(e_n, x)$  не должна нести никакой информации об  $x$ . Эти требования уточняются следующим образом.

(а) Правильность расшифровки: для любой последовательности сообщений  $x_n$  длины  $\text{poly}(n)$  с вероятностью приблизительно равной 1 выполнено  $D(e, E(e, x)) = x$ . (Для упрощения обозначений мы опускаем индекс  $n$ .) Источником случайности здесь является не только случайная величина  $e$ , но и случайные биты алгоритмов  $E, D$ . Мы предполагаем,

что случайные биты алгоритмов  $E, D$  независимы от  $e$ .

(б) Неразглашение информации: для любого полинома  $p(n)$  и для любых двух последовательностей сообщений  $a_n, b_n$  таких, что длины  $a_n, b_n$  равны  $p(n)$ , случайные величины  $E(e, a_n)$  и  $E(e, b_n)$  вычислительно неотличимы. Источник случайности в этом определении имеется в  $e$  и в случайных битах алгоритмов  $E, D$ , которые предполагаются независимыми от  $e$ .

Условие (б) очевидно эквивалентно своему частному случаю: неотличимости случайных величин  $E(e, a_n)$  и  $E(e, 00 \dots 0)$  (подряд  $p(n)$  нулей). Вторую из этих случайных величин можно генерировать, зная только  $n$ . Поэтому, какое бы сообщение мы не зашифровывали случайно выбранным ключом, шифрограмма не будет нести никакой информации — некоторую неотличимую от нее случайную величину можно сгенерировать и не зная исходного сообщения. Заметим, что в наших определениях важно, что ключ выбирается случайно, а не фиксирован. Мы не можем требовать неотличимости шифрограмм  $E(e, a_n)$  и  $E(e, b_n)$  при любом фиксированном ключе  $e$ , поскольку любой фиксированный ключ  $e$  можно “запаять” в схему-отличитель, которая будет применять алгоритм  $D$  и, скажем, выдавать первый бит расшифрованного сообщения.

Из условия (б) следует, что, не зная ключа, никакой бит исходного сообщения нельзя найти по шифрограмме с вероятностью существенно больше  $1/2$ , если сообщение выбирается случайно с равномерным распределением среди всех строк какой-то фиксированной полиномиальной от  $n$  длины  $k$ . Действительно, пусть противник  $C$  (схема полиномиальной от  $n$  размера) пытается по шифрограмме восстановить, скажем, первый бит сообщения. Из условия (б) следует, что вероятности событий  $C(E(e, 0a)) = 1$  и  $C(E(e, 1a)) = 1$  (где  $a$  случайная строка длины  $k - 1$ ) приблизительно равны. Другими словами сумма вероятностей событий  $C(E(e, 0a)) = 0$  и  $C(E(e, 1a)) = 1$  приблизительно равна 1, следовательно, вероятность события  $C(E(e, b)) = b[1]$  (где  $b$  случайная строка длины  $k$ ) примерно равна  $1/2$ .

*Задача 35.* Пусть противник интересуется некоторой информацией  $f(x)$  о передаваемом сообщении  $x$  (где  $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$  — некоторая функция). Допустим противник применяет к шифрограмме  $E(e, x)$  некоторую схему  $D_n$  полиномиальной от  $n$  размера для извлечения этой информации. Докажите, что существует доступная случайная величина  $\alpha_n$  такая, что для любой последовательности сообщений  $x_n$  полиномиальной

длины, вероятности событий  $D_n(E(e, x_n)) = f(x_n)$  (вероятность успеха атаки) и  $\alpha_n = f(x_n)$  приблизительно равны. То есть, информация  $f(x_n)$  может быть вычислена и без подслушивания с примерно той же вероятностью успеха.

Пара алгоритмов  $E, D$  и последовательность случайных величин  $e_n$  называется *схемой шифрования с закрытым ключом*, если выполнены все перечисленные условия. Схему шифрования с закрытым ключом можно построить на основе любого генератора ПСЧ.

**Теорема 15.** *Если существует генератора  $G_n$ , то существует и схема шифрования с закрытым ключом.*

*Доказательство.* Сначала построим схему шифрования с закрытым ключом, для которой свойство (б) выполнено только для какого-нибудь одного фиксированного полинома  $p(n)$  (а не для всех полиномов, как требуется). Эта схема называется *гаммированием (one-time pad)*. В качестве ключа возьмем случайную строчку  $\gamma_n$  длины  $p(n)$  и положим  $E(\gamma_n, x) = \gamma_n \oplus x$  (побитовое сложение по модулю 2) и  $D(\gamma_n, y) = \gamma_n \oplus y$ . (Если длина сообщения не равна  $p(n)$ , то положим  $E(\gamma_n, x) = x$ ,  $D(\gamma_n, y) = y$ ; второе условие в этом случае, разумеется, не выполнено, но это и не требуется.) Условие (а) очевидно выполнено. В условии (б) вероятность события  $C_n(\gamma_n \oplus a_n) = 1$  равна вероятности события  $C_n(\gamma_n) = 1$ , а потому не зависит от  $a_n$ . При этом нам даже не важно, что схема  $C_n$  имеет полиномиальный размер — случайные величины  $\gamma_n \oplus a_n$  и  $\gamma_n$  имеют одинаковое распределение.

Недостатком гаммирования является то, что длина ключа равна длине сообщения, поэтому на практике его можно применять только для шифрования коротких сообщений.

Теперь модифицируем схему гаммирования, заменив случайную величину  $\gamma_n$  на псевдослучайную величину, порожденную генератором. Ключ  $s$  равномерно распределен среди строк длины  $n$ . Алгоритм  $E(s, x)$  является детерминированным и выдает побитовую сумму  $x$  и первых  $|x|$  битов последовательности  $G_n(s)$ . Будем обозначать выданную им последовательность через  $x \oplus G(s)$ . Алгоритм  $D(s, y)$  выдает  $y \oplus G(s)$ , побитовую сумму  $y$  и первых  $|y|$  битов последовательности  $G(s)$ . Ясно, что условие (а) выполнено. Для проверки условия (б) фиксируем произвольный полином  $p(n)$ . Нам надо доказать неотличимость случайных величин  $a_n \oplus G(s)$  и  $b_n \oplus G(s)$ , где  $a_n, b_n$  любые последовательности слов длины

$p(n)$ . Поскольку функция  $G$  является генератором, случайная величина  $G(s)[1 : p(n)]$  вычислительно неотличима от случайной величины  $\gamma_{p(n)}$ , равномерно распределенной среди слов длины  $p(n)$ . Значит, случайная величина  $a_n \oplus G(s)$  неотличима от случайной величины  $a_n \oplus \gamma_{p(n)}$ . В самом деле, если бы они были отличимы тестом  $T_n(y)$  размера  $\text{poly}(n)$  то тест  $T_n(a_n \oplus y)$ , размер которого лишь немного превышает размер  $T_n$ , отличал бы  $G(s)$  и  $\gamma$ . Случайная величина  $a_n \oplus \gamma_{p(n)}$  имеет равномерное распределение, значит случайная величина  $a_n \oplus G(s)$  неотличима от случайной величины  $\gamma_{p(n)}$ . То же самое справедливо для случайной величины  $b_n \oplus G(s)$ , значит она неотличима от  $a_n \oplus G(s)$ .  $\square$

## 5.2 Многоразовые схемы

Схему, построенную в доказательстве этой теоремы нельзя применять дважды с одним и тем же ключом. Действительно, узнав шифрограммы сообщений  $a$  и  $b$ , то есть, узнав  $a \oplus G(s)$  и  $b \oplus G(s)$ , противник узнает побитовую сумму переданных сообщений  $a \oplus b$ . Поэтому схемы шифрования, которые удовлетворяют условиям (а) и (б) мы называли “одноразовыми”. В каком случае схему шифрования можно назвать многоразовой? Неформально, многоразовая схема не должна разглашать информации, если ее применять с одним и тем же ключом для шифрования полиномиального числа произвольных сообщений. Требование правильности расшифровки в определении многоразовой схемы то же самое.

А вот требование неразглашения информации теперь формулируется так:

(б) Для любых полиномов  $p(n), q(n)$  существует полиномиально моделируемая последовательность  $\xi_n$  с таким свойством. Для любой последовательности из  $q(n)$  слов  $x_n^1, \dots, x_n^{q(n)}$  длины  $p(n)$  последовательность случайных величин

$$E(e, x_n^1), E(e, x_n^2), \dots, E(e, x_n^{q(n)})$$

вычислительно неотличима от  $\xi_n$ .

В многоразовой схеме алгоритм  $E$  обязан быть вероятностным. Действительно, иначе шифрограммы двух одинаковых сообщений были бы одинаковыми, а значит, шифрограмма  $E(e, x), E(e, x)$  была бы вычислительно отличимой от шифрограммы  $E(e, x), E(e, y)$  при  $y \neq x$ : в первой



из них первое слово равно второму, а во второй не равно (поскольку из условия (а) следует, что шифрограммы разных сообщений различны).

Достаточно построить многоразовую схему шифрования для шифрования однобитовых сообщений: сообщение из нескольких битов можно шифровать по биту. Условие надежности (б) гарантирует, что так построенная схема снова надежна. А схема шифрования однобитовых сообщений может быть построена из семейства ПСФ.

*Конструкция многоразовой схемы шифрования из семейства ПСФ.* Пусть дано семейство ПСФ  $f_s^n : \{0, 1\}^n \rightarrow \{0, 1\}$ . Требуется построить многоразовую схему шифрования однобитовых сообщений. Ключом многоразовой схемы является идентификатор  $s$  функции  $f_s$  из данного семейства ПСФ. Шифрограмма сообщения  $m \in \{0, 1\}$  состоит из случайно выбранного слова  $z$  длины  $n$  и еще одного бита  $m \oplus f_s(z)$ :

$$\tilde{E}(s, x) = (z, m \oplus f_s(z)).$$

Алгоритм расшифровки  $D$  в применении к ключу  $s$  и шифрограмме  $(z, c)$  выдает  $c \oplus f_s^n(z)$ :

$$D(s, (z, c)) = c \oplus f_s^n(z).$$

Очевидно, что алгоритмы генерации ключей, шифровки и расшифровки полиномиальны по времени.

**Теорема 16.** *Если семейство ПСФ удовлетворяет требованию (б) на стр. 53, то построенная схема шифрования удовлетворяет требованию (б).*

*Доказательство.* Пусть дана последовательность слов  $x_n$  длины  $q(n) = \text{poly}(n)$ , для которых надо установить свойство (б). В качестве  $\xi_n$  рассмотрим случайную величину

$$\xi_n = (z^1, r_1), \dots, (z^{q(n)}, r_{q(n)}),$$

где  $z^1, \dots, z^{q(n)}$  — независимые строки длины  $n$ , а  $r^1, \dots, r^{q(n)}$  — независимые биты. Нам надо доказать вычислительную неотличимость этой случайной величины от случайной величины

$$\eta_n = (z^1, x_n^1 \oplus f_s(z^1)), \dots, (z^{q(n)}, x^{q(n)} \oplus f_s(z^{q(n)})).$$

Для этого докажем вычислительную неотличимость обоих случайных величин от третьей случайной величины  $\theta_n$ , которая получится, если в

определении  $\eta_n$  псевдослучайную функцию заменить на случайную:

$$\theta_n = (z^1, x_n^1 \oplus g(z^1)), \dots, (z^{q(n)}, x^{q(n)} \oplus g(z^{q(n)})).$$

Случайные величины  $\eta_n$  и  $\theta_n$  неотличимы потому, что семейство ПСФ удовлетворяет свойству (б), а случайные величины  $\xi_n$  и  $\theta_n$  неотличимы потому, что с приблизительно единичной вероятностью все строки  $z_i$  попарно различны, поэтому статистическое расстояние между ними пренебрежимо мало.  $\square$

Говорят, что посол, получив по дипломатическим каналам зашифрованную ноту, прежде чем оглашать ее текст, должен хотя бы немного изменить ее текст (например, переставив пару слов), чтобы не разгласить информации о ключе, использованном при шифровании (противнику, который уже знает содержание шифрограммы этой ноты). Для многоразовых схем, удовлетворяющих нашему определению, в этом нет необходимости. В самом деле, пусть мы зашифровали дипломатическую ноту  $x_1$  с помощью ключа  $e$ , а потом с помощью того же ключа  $e$  зашифровали некоторое секретное сообщение  $x_2$ . Тогда случайная величина  $x_1 E(e, x_1) E(e, x_2)$  вычислительно неотличима от случайной величины  $x_1 E(e, 00 \dots 0) E(e, 00 \dots 0)$ , то есть мы разгласили только  $x_1$  и длину  $x_2$ .

### 5.3 Более экономная многоразовая СШЗК

Недостатком построенной СШЗК является большая длина шифрограммы: она примерно равна произведению параметра безопасности  $n$  и длины сообщения. Небольшим усложнением схемы произведение можно заменить на сумму.

[Дописать.]

## Глава 6

# Шифрование с открытым ключом

### 6.1 Определение схемы шифрования с открытым ключом

В схеме шифрования с открытым ключом имеется два ключа — открытый и закрытый. Открытый ключ используется для шифровки, а закрытый для расшифровки. От схемы шифрования требуется, чтобы противник из открытого ключа и шифрограммы любого сообщения не смог получить никакой информации о сообщении. Точнее, для любых двух сообщений  $x_1, x_2$  одной длины случайная величина  $(e, c_1)$ , состоящая из открытого ключа  $e$  и шифрограммы  $s_1$  сообщения  $x_1$ , была вычислительно неотличима от случайной величины  $(e, c_2)$ , состоящей из открытого ключа и шифрограммы сообщения  $x_2$ . Открытый ключ можно опубликовать, тогда любой может посылать зашифрованные сообщения, расшифровывать которые может только владелец закрытого ключа.

Схема шифрования с открытым ключом состоит из полиномиальных вероятностных алгоритмов  $E$  и  $D$ , называемых, алгоритмами *шифровки* и *расшифровки* и полиномиально моделируемой последовательности случайных величин  $\langle e_n, d_n \rangle$ . Строки  $e_n$  и  $d_n$  называются *открытым и закрытым ключами*.

Алгоритм  $E$  получает на вход  $1^n$ , ключ  $e_n$  и сообщение  $x \in \{0, 1\}^*$  и выдает некоторую строку  $y = E_n(e_n, x)$ , называемую *шифрограммой*, длина которой зависит только от длин  $n$  и  $x$ . Алгоритм  $D$  получает на

вход  $1^n$ ,  $y$  и  $d$  и выдает некоторую строку  $D_n(d_n, y)$ . Тройка  $(E, D, \langle e_n, d_n \rangle)$  должна удовлетворять следующим требованиям.

(а) Правильность расшифровки: для любой последовательности сообщений  $x_n$  длины  $\text{poly}(n)$  с вероятностью приблизительно равной 1 выполнено  $D(e, E(d, x)) = x$  (то есть, вероятность отличается от 1 на пренебрежимо малую величину).<sup>1</sup> Источником случайности здесь является не только случайная величина  $\langle e, d \rangle$ , но и случайные биты алгоритмов  $E, D$ , которые предполагаются независимыми в совокупности.

Следующее условие удобно сформулировать с помощью такого понятия. Пусть  $\alpha_n, \beta_n, \gamma_n$  совместно распределенные случайные величины. Будем говорить, что случайные величины  $\alpha_n$  и  $\beta_n$  *вычислительно (статистически) неотличимы при известном  $\gamma_n$* , если случайные величины  $\alpha_n \gamma_n$  и  $\beta_n \gamma_n$  вычислительно (статистически) неотличимы.

(б) Для любого полинома  $p(n)$  и для любых двух последовательностей сообщений  $a_n, b_n$  таких, что длины  $a_n, b_n$  равны  $p(n)$ , случайные величины  $E(e, a_n)$  и  $E(e, b_n)$  вычислительно неотличимы при известном  $e_n$ . Источником случайности здесь является не только случайная величина  $\langle e, d \rangle$ , но и случайные биты алгоритма  $E$ . Мы предполагаем, что случайные биты алгоритма  $E$  независимы от  $\langle e, d \rangle$ .

Пара алгоритмов  $E, D$  и последовательность случайных величин  $\langle e_n, d_n \rangle$  называется *схемой шифрования с открытым ключом*, если выполнены все перечисленные условия. Для существования схемы шифрования с открытым ключом требуется более сильная гипотеза, чем существование генератора. Такую схему удастся построить при условии существования односторонней перестановки с секретом (определение дано в следующем разделе).

*Задача 36.* Пусть дана схема шифрования  $K, E, D$  с открытым ключом. Преобразуем ее в схему шифрования с закрытым ключом соединив открытый и закрытый ключи  $e, d$  в один закрытый ключ  $\tilde{d} = \langle e, d \rangle$  и модифицировав алгоритмы  $K, E, D$  соответствующим образом. Докажите, что мы получим в самом деле схему шифрования с закрытым ключом.

<sup>1</sup>Для упрощения обозначений мы опускаем индекс  $n$ .

## 6.2 Построение схемы шифрования с открытым ключом

**Теорема 17.** *Если существует односторонняя проверяемая перестановка с секретом, то существует и схема шифрования с открытым ключом.*

*Доказательство.* Сначала научимся шифровать один бит. Гаммирование нам уже не поможет, потому что в схеме гаммирования закрытый ключ совпадал с открытым.

Пусть дана односторонняя проверяемая функция  $g^e$  с секретом. Как мы видели, немного и изменив ее, мы можем построить трудный бит  $h^e$  для нее. Теперь построим схему шифрования с открытым ключом одного бита  $b$ . Случайную величину  $\langle e_n, d_n \rangle$  заимствуем из необратимой функции. Алгоритм шифрования выдает пару  $y = \langle b \oplus h^e(x), g^e(x) \rangle$ . Здесь  $x$  выбирается по распределению  $\xi^e$  из условия (в) для функции  $g^e$ .

Зная  $d$  расшифровать  $y$  просто: надо найти  $x$ , обратив  $g^e(x)$ , и сложить первую компоненту  $y$  с  $h^e(x)$ . Вероятность ошибки при расшифровке складывается из статистического расстояния между случайными величинами  $(e, \xi^e)$  и  $(e, \gamma^e)$  (где  $\gamma^e$  равномерно распределено в  $D^e$ ) и вероятности ошибки алгоритма из условия (г). Обе вероятности по условию пренебрежимо малы.

Шифрограмма любого из двух битов вместе с открытым ключом  $e$  будет вычислительно неотличима от случайной величины  $\langle r, g^e(x), e \rangle$ , где  $r$  случайный равномерно распределенный бит, независимый от  $x, e$ . Действительно, по лемме 3.4 случайная величина  $\langle r, g^e(x), e \rangle$  вычислительно неотличима от случайной величины  $\langle h^e(x), g^e(x), e \rangle$ . Поэтому для любого бита  $b$  случайные величины  $\langle b \oplus r, g^e(x), e \rangle$  и  $\langle b \oplus h^e(x), g^e(x), e \rangle$  вычислительно неотличимы. Первая из них имеет то же распределение, что и  $\langle r, g^e(x), e \rangle$ . При замене равномерного распределения на  $D^e$  на распределение, генерируемое алгоритмом из пункта (в), получим случайную величину, статистически неотличимую от исходной.

Теперь построим схему шифрования произвольного числа битов. Опять заимствуем алгоритм генерации ключей из необратимой функции. Шифрограмму сообщения  $b_1 b_2 \dots b_m$  из  $m$  битов определим как

$$b_1 \oplus h^e(x), b_2 \oplus h^e(g^e(x)), \dots, b_m \oplus h^e((g^e)^{m-1}(x)), (g^e)^m(x).$$

Все условия, кроме условия (б), очевидны. Условие (б) следует из

леммы 3.4. Пусть  $p(n)$  произвольный полином. Из неё следует вычислительная неотличимость случайных величина

$$e, b_1 \oplus h^e(\gamma^e), b_2 \oplus h^e(g^e(\gamma^e)), \dots, b_{p(n)} \oplus h^e((g^e)^{p(n)-1}(\gamma^e)), (g^e)^{p(n)}(\gamma^e)$$

и  $e, b_1 \oplus r_1, b_2 \oplus r_2, \dots, b_{p(n)} \oplus r_{p(n)}, \gamma^e$  вычислительно неотличимы. Последняя из них имеет то же распределение, что и случайная величина

$$e, r_1, r_2, \dots, r_{p(n)}, \gamma^e,$$

которая не зависит от  $b_1 \dots, b_{p(n)}$ . Поэтому при разных сообщениях  $b_1 \dots, b_{p(n)}$  случайные величины, состоящие из открытого ключа и шифрограммы, вычислительно неотличимы друг от друга.  $\square$

*Задача 37.* Докажите, что можно из любой схемы шифрования с открытым ключом одного бита можно изготовить схему шифрования с открытым ключом произвольного количества битов с помощью шифрования каждого бита по отдельности с помощью одного и того же открытого ключа.

## Глава 7

# Протоколы привязки к биту (bit commitment)

Протоколы привязки к биту имитируют сундучок с замком, в который первый игрок (отправитель) может положить один бит, закрыть его на ключ и передать второму игроку (получателю). Получатель не может открыть сундучок до тех пор, пока отправитель не передаст ему ключ. Отправитель не может подменить бит, положенный в сундучок. Различают два вида таких протоколов — интерактивные и неинтерактивные. Начнем со вторых, как более простых.

Неинтерактивный протокол привязки к биту состоит из двух алгоритмов: вероятностного полиномиального алгоритма  $S$  и детерминированного полиномиального алгоритма  $R$ . Алгоритм  $S$  получает на вход параметр безопасности  $n$  и один бит  $\sigma$  и выдает на выход *ключ*  $k_n(\sigma)$  и *привязку*  $c_n(\sigma)$ . Случайные величины  $k(\sigma)$  и  $c(\sigma)$  (мы опускаем параметр безопасности в обозначениях) имеют параметр  $\sigma$  и являются зависимыми, поскольку имеют общий источник случайности — результаты бросаний, выполненных алгоритмом  $S(\sigma)$ . Привязка имитирует закрытый сундучок. Алгоритм  $R$  получает на вход две строки  $k$  (ключ) и  $c$  (привязку) и выдает на выход один бит или специальный символ  $\perp$ , означающий, что ключ  $k$  не подошел к замку.

Требования к этим алгоритмам такие.

(а) Для любого  $n$  и любого бита  $\sigma$  с вероятностью, приблизительно равной 1, выполнено  $R(c(\sigma), k(\sigma)) = \sigma$ . Это требование означает, что при открытии сундучка настоящим ключом там обнаруживается положенный туда бит.

(б) Последовательности случайных величин  $c_n(0)$  и  $c_n(1)$  вычислительно неотличимы. Это свойство протокола формализует непрозрачность сундучка: не имея ключа, получатель не имеет никакой информации о содержимом сундучка.

(в) Ни для какого  $n$  не существует таких  $c, k_0, k_1$ , что  $R(c, k_0) = 0$  и  $R(c, k_1) = 1$ . Это требование выражает невозможность, подбирая ключи, обнаружить в сундучке два разных бита: любая строка  $c$  может быть привязкой только к одному биту.

Если выполнены все три условия, то пара алгоритмов  $(R, S)$  называется *протоколом привязки к биту*. Заметим, что с помощью перебора по привязке  $c(\sigma)$  можно найти  $\sigma$  (с почти единичной вероятностью). В самом деле, мы можем перебрать все строки  $k$ , для каждой из них применить алгоритм  $R$  к паре  $c(\sigma), k$  и выдать первый результат, отличный от  $\perp$ . По условию (а) для некоторого  $k$  (а именно для  $k = k(\sigma)$ ) алгоритм  $R$  выдаст  $\sigma$ , и по условию (в) все полученные результаты, отличные от  $\perp$ , равны  $\sigma$ . Поэтому в свойстве (б) важно, что неотличимость  $c_n(0)$  и  $c_n(1)$  вычислительная, а не статистическая — статистическое расстояние между  $c_n(0)$  и  $c_n(1)$  близко к 1.

Протокол привязки к биту можно построить, имея любую одностороннюю перестановку  $f_n : D_n \rightarrow D_n$ . Правда, нужно еще, чтобы перестановка имела следующее дополнительное свойство: алгоритм генерации распределения, статистически неотличимого от равномерного распределения на  $D_n$ , при любых исходах бросаний выдает либо строку из  $D_n$ , либо пустую строку. Это дополнительное условие имеет место для всех трех известных нам кандидатов на одностороннюю перестановку: функции Рабина, RSA и дискретной экспоненты. Для этих трех функций алгоритм генерации равномерного распределения на  $D_n$  знает, когда ему не удалось найти элемент из  $D_n$ , и в этом случае может выдать пустую строку.

**Теорема 18.** *Если существует односторонняя перестановка  $f_n : D_n \rightarrow D_n$  с таким свойством, то существует и протокол привязки к биту.*

*Доказательство.* По теореме Левина-Голдрайха из условия теоремы следует существование функции  $g_n$ , удовлетворяющей условию теоремы и трудного бита  $h_n$  к ней. Алгоритм  $S$  выбирает случайную строку  $x$  из области определения  $g_n$  (или пустую строку) с помощью полиномиального алгоритма  $K$  генерации распределения, статистически неотличимого от



равномерного распределения на  $D_n$ . При этом он запоминает использованные случайные биты  $r$  и выдает их в качестве ключа  $k$ . В качестве привязки  $s$  он выдает пару  $\langle \sigma \oplus h(x), g(x) \rangle$  (мы опускаем в обозначениях параметр безопасности  $n$ ). Алгоритм  $R$ , получив на вход строку случайных битов  $r$  и пару  $\langle \delta, y \rangle$ , состоящую из одного бита и строки, сначала находит строку  $x$ , запуская алгоритм  $K$  на  $r$  и  $1^n$ , затем проверяет равенство  $g(x) = y$ . Если равенство неверно или  $x$  пусто, он выдает  $\perp$ . Иначе он выдает  $\delta \oplus h(x)$ .

Условие (а) очевидно выполнено. Условие (в) следует из инъективности  $g$ . Действительно, если  $g(x_0) = y = g(x_1)$ , и  $x_0, x_1 \in D_n$ , то  $x_0 = x_1$ , следовательно,  $\delta \oplus h(x_0) = \delta \oplus h(x_1)$ .

Условие (б) означает вычислительную неотличимость случайных величин  $\langle h(x), g(x) \rangle$  и  $\langle 1 \oplus h(x), g(x) \rangle$  (где  $x$  выбирается равномерно из  $D_n$ ). Это следует из того, что они обе неотличимы от случайной величины  $\langle \gamma, g(x) \rangle$ , где  $\gamma$  случайный бит, независимый от  $x$ .  $\square$

*Замечание 2.* В доказательстве мы не использовали того, что функция  $f$  является наложением, поэтому протокол привязки к биту можно построить, исходя из любой односторонней инъективной функции  $f_n : D_n \rightarrow \{0, 1\}^*$  при вышеуказанном условии на алгоритм генерации равномерного распределения на  $D_n$ .

*Задача 38.* Докажите, что без ограничения общности можно считать, что в неинтерактивных протоколах привязки к биту ключ всегда равен  $\sigma r$  (где  $\sigma$  — исходный бит, а  $r$  случайные биты алгоритма  $S$ ).

*Задача 39.* Пусть дана односторонняя перестановка  $g_n$  и трудный бит  $h_n$  для нее. Рассмотрим следующий протокол привязки к биту. Посылающий выбирает случайное  $x$  из  $D_n$  и проверяет, верно ли, что  $h_n(x)$  совпадает с битом, к которому надо привязаться. Если это так, то в качестве привязки выдается  $g_n(x)$ , а в качестве ключа случайные биты, использованные при генерации  $x$ . Если это не так, то все повторяется сначала. Так делается  $n$  раз, и если все попытки неудачны, то выдается  $g_n(x)$  для последнего  $x$ . Алгоритм раскрытия сундучка проверяет, что значение функции  $g_n$  на ключе равно привязке и в этом случае выдает  $h_n(x)$  (а иначе  $\perp$ ). Докажите, что этот протокол удовлетворяет требуемым свойствам.

В следующем разделе мы построим интерактивный протокол привязки к биту.

74 ГЛАВА 7. ПРОТОКОЛЫ ПРИВЯЗКИ К БИТУ (BIT COMMITMENT)

## Глава 8

# Интерактивные протоколы привязки

### 8.1 Протоколы привязки к биту

В интерактивном протоколе получатель также участвует в положении бита в сундучок. Отправитель и получатель обмениваются сообщениями, которые записываются на магнитофонную ленту. Эта запись и играет роль привязки. Определение интерактивного протокола привязки немного сложнее, чем неинтерактивного. Но зато интерактивный протокол можно построить, исходя из любого генератора псевдослучайных чисел.

Интерактивный протокол привязки к биту состоит из трех вероятностных полиномиальных интерактивных алгоритмов  $S, T$  и детерминированного полиномиального алгоритма  $R$  (требование детерминированности алгоритма  $R$  не очень существенно, но для детерминированных алгоритмов определение проще). Алгоритм  $S$  получает на вход параметр безопасности  $n$  (в унарной записи) и один бит  $\sigma$ , а алгоритм  $T$  получает на вход только параметр безопасности (тот же, что и алгоритм  $S$ ). После этого эти алгоритмы общаются друг с другом, то есть, по очереди посылают друг другу сообщения (без ограничения общности можно считать, что общение начинает  $S$ ). Через  $c_{T,S(\sigma)}$  мы будем обозначать последовательность всех переданных сообщений между  $S$  и  $T$ , если  $S$  получил на вход  $\sigma$ . Последовательность  $c_{T,S(\sigma)}$  является случайной величиной, причем зависящей от параметра безопасности (который

мы для краткости опускаем в обозначениях). Она выполняет роль привязки в неинтерактивных протоколах. Кроме того, алгоритм  $S$  выдает на выход строку  $k_{T,S(\sigma)}$ , называемую ключом, которая может зависеть от сообщений, полученных от  $T$  (поэтому мы пишем  $T$  в качестве дополнительного индекса). Случайные величины  $c_{T,S(\sigma)}$  и  $k_{T,S(\sigma)}$  имеют параметр  $\sigma$  и являются зависимыми (при любом значении  $\sigma$ ), поскольку используют общий источник случайности — результаты бросаний, выполненных вероятностными алгоритмами  $S(\sigma)$  и  $T$ . Алгоритм  $R$  получает на вход параметр безопасности  $n$ , строку  $c$  (привязку) и строку  $k$  (ключ) и выдает на выход один бит или сообщение о том, что один из двух игроков заблокировал протокол. Будем эти сообщения обозначать  $\perp_S$ ,  $\perp_R$  ( $\perp_S$  означает, что протокол заблокирован посылающим, а  $\perp_R$  — получателем).

Тройка алгоритмов  $T, S, R$  такого вида называется интерактивным протоколом привязки, если выполнены следующие три условия (а), (б) и (в). В первом и втором условиях выражены интересы посылающего. Они состоят в том, чтобы при применении правильного ключа в сундучке обнаруживался положенный туда бит (а) и чтобы сундучок был непрозрачным (б). Третье условие (в) выражает интересы получателя — невозможно один и тот же сундучок открыть двумя разными ключами так, чтобы в первый раз там был обнаружен 0, а во второй раз — 1. При формулировке первого и второго свойств мы должны предусмотреть, что получатель может жульничать, применяя вместо алгоритма  $T$  а какую-то стратегию  $T^*$ , вычислимую схемой  $T_n^*$  полиномиального от  $n$  размера (схема зависит от параметра безопасности  $n$ ). В третьем свойстве мы должны предусмотреть, что жульничать может отправитель, применяя вместо алгоритма  $S$  какую-то стратегию  $S^*$ , вычислимую схемой  $S_n^*$  полиномиального от  $n$  размера.

Итак, тройка алгоритмов  $T, S, R$  называется *интерактивным протоколом привязки*, если для любой последовательности схем  $\{T_n^*\}$  полиномиального размера выполнены условия следующие условия (а), (б) и для любой последовательности схем  $\{S_n^*\}$  полиномиального размера выполнено следующее условие (в).

(а) С вероятностью, приблизительно равной 1 (при  $n \rightarrow \infty$ ), для любого бита  $\sigma$  алгоритм  $R$  на входе  $c_{T^*,S(\sigma)}$  и  $k_{T^*,S(\sigma)}$  выдает либо  $\sigma$ , либо  $\perp_R$  (для краткости мы опускаем параметр безопасности  $n$  в обозначениях).

(б) Случайные величины  $c_{T^*,S(0)}$  и  $c_{T^*,S(1)}$  вычислительно неотличимы.

(в) Пренебрежимо мала вероятность каждого из двух событий

- существуют такие слова  $k_0, k_1$ , что  $R(c_{T,S^*}, k_0) = 0$  и  $R(c_{T,S^*}, k_1) = 1$ ,
- существует  $k$ , для которого  $R(c_{T,S^*}, k) = \perp_R$ .

Здесь  $c_{T,S^*}$  обозначает запись беседы схемы  $S_n^*$  с вероятностным алгоритмом  $T(1^n)$ . Если существуют строки  $k_0, k_1$ , для которых  $R(c, k_0) = 0$  и  $R(c, k_1) = 1$ , то мы будем говорить, что привязка  $c$  двусмысленна. Таким образом первая часть условия (в) гарантирует, что запись беседы  $S^*$  и  $T$  является двусмысленной с пренебрежимо малой вероятностью. Заметим, что мы не предполагаем, что схема  $S_n^*$  получает на вход некоторый бит  $\sigma$ , как это делал “честный” алгоритм  $S$ . Это делается потому, что жульничающий отправитель заинтересован только в том, чтобы создать двусмысленную привязку, а не привязаться к какому-то конкретному биту.

Как обычно, во всех трех условиях можно считать схемы  $T_n^*$  и  $S_n^*$  вероятностными. Точнее, если тройка  $T, S, R$  является интерактивным протоколом привязки, то условия (а), (б) и (в) выполнены и для любых последовательностей вероятностных схем  $S_n^*$  и  $T_n^*$  полиномиального размера. В частности, для любых полиномиальных вероятностных алгоритмов  $S', T'$  условия (а), (б) и (в) выполнены для стратегий  $S_n^* = S'(1^n)$  и  $T_n^* = T'(1^n)$ . По этой причине из условия (а) и второй части условия (б) следует, что для всех  $\sigma$  с приблизительно единичной вероятностью  $R(c_{T,S(\sigma)}, k_{T,S(\sigma)}) = \sigma$  (если и отправитель, и получатель действуют честно, то в сундучке обнаруживается положенный туда бит). В самом деле, рассмотрим в качестве  $S', T'$  исходные алгоритмы  $S, T$ . Тогда вторая часть условия (в) гарантирует, что вероятность результата  $\perp_R$  пренебрежимо мала. В то же время по требованию (а) с приблизительно единичной вероятностью результат равен либо  $\sigma$ , либо  $\perp_R$ , а значит равен  $\sigma$ .

Любой неинтерактивный протокол привязки  $(S, R)$  легко переделать в интерактивный протокол. Для этого нужно оставить алгоритмы  $S, R$ , как они есть, а в качестве  $T$  взять тривиальный алгоритм, который в качестве своего единственного сообщения посылает пустое слово.

*Задача 40.* Докажите, что для любого протокола привязки к биту существует функция  $b(c)$  со значениями 0,1 такая, что для любой последовательности стратегий  $\{S_n^*\}$  с приблизительно единичной вероятностью для всех  $k$  алгоритм  $R$  на входе  $c_{S^*,T}, k$  выдаёт либо  $b(c_{S^*,T})$ , либо  $\perp_S$ . Говоря неформально,  $b(c)$  есть тот бит, который закрыт в сундучке.

**Теорема 19.** *Если существует генератор ПСЧ, то существует интерактивный протокол привязки к биту.*

*Доказательство.* Пусть  $G$  генератор ПСЧ  $\{0, 1\}^n \rightarrow \{0, 1\}^{3n}$ . Алгоритм  $T$  посылает случайную строчку  $r$  длины  $3n$ , после чего останавливается. Алгоритм  $S$  ждет от  $T$  строчку  $r$  длины  $3n$ . Если присланная строка имеет другую длину, то он заканчивает диалог (посылая пустое слово). Иначе, он выбирает случайную строчку  $s$  длины  $n$  и посылает  $G(s)$ , если  $\sigma = 0$ , и посылает  $G(s) \oplus r$  иначе (то есть, он посылает  $G(s) \oplus \sigma \cdot r$ ). Ключ, выдаваемый алгоритмом  $S$  есть  $\sigma s$ .

Входом алгоритма  $R$ , кроме  $n$ , является пара строк  $r, x$  и строка  $k$ . Если все строки имеют правильную длину, причем  $x = G(s) \oplus \sigma \cdot r$ , где  $\sigma$  первый бит строки  $k$ , а  $s$  — остальные ее биты, то алгоритм  $R$  выдает  $\sigma$ . Иначе, алгоритм  $R$  разбирается, кто заблокировал протокол: если длина  $r$  не равна  $3n$ , то заблокировал получатель. Если длина  $r$  равна  $3n$ , но длина  $k$  не равна  $n + 1$  или  $x \neq G(s) \oplus \sigma \cdot r$ , то заблокировал отправитель.

Ясно, что условие (а) и вторая часть условия (в) выполнены. Проверим первую часть условия (в). Пусть дана последовательность стратегий  $S_n^*$ . Алгоритм  $R$  получает на вход ключ  $k$  и диалог, который состоит из строк  $r, S^*(r)$ . Допустим, алгоритм  $R$ , получив на вход ключи  $k_0, k_1$  и привязку  $r, S^*(r)$ , выдает соответственно  $0, 1$ . Ясно, что тогда  $k_0 = 0s_0$  и  $k_1 = 1s_1$  для некоторых строк  $s_0, s_1$  длины  $n$ . Поэтому достаточно доказать, что вероятность события

$$(\exists s_0, s_1) \quad S^*(r) = G(s_0), \quad S^*(r) = G(s_1) \oplus r$$

пренебрежимо мала. Вероятность этого события не больше вероятности того, что найдутся  $s_0, s_1$  такие, что  $G(s_0) = G(s_1) \oplus r$ . При фиксированных  $s_0, s_1$  вероятность этого события равна  $2^{-3n}$ . Суммируя по всем  $s_0, s_1$ , получаем не больше  $2^{-n}$ , что стремится к нулю быстрее любого обратного полинома от  $n$ .

Проверим условие (б). Пусть дана последовательность схем  $T_n^*$  полиномиального от  $n$  размера. Схема  $T_n^*$  применяется только дважды — первый раз к пустой последовательности, а второй раз — к первому сообщению Получателя (если второе сообщение, посланное схемой, не пусто, то общение оборвёт Посылающий). Ограничимся случаем, когда первое сообщение, посланное схемой, имеет длину  $3n$ , а второе пусто, как и у алгоритма  $T(1^n)$ . На общий случай рассуждения переносятся очевидным

образом. Обозначим первое сообщение, посланное схемой, через  $r_n^*$ . Диалог Посылающего и Получателя в случаях для  $\sigma = 0$  и  $\sigma = 1$  выглядит, соответственно, так

$$\langle r^*, G(s), \Lambda \rangle, \quad \langle r^*, G(s) \oplus r^*, \Lambda \rangle,$$

соответственно. Нам нужно доказать вычислительную неотличимость этих случайных величин. Это следует из того, что обе они вычислительно неотличимы, от случайной величины  $\langle r_n^*, \gamma_{3n}, \Lambda \rangle$ , где  $\gamma_{3n}$  равномерно распределенная среди слов длины  $3n$  случайная величина.  $\square$

*Замечание 3.* Отметим, что для построенного в доказательстве протокола условие (б) выполнено не только для схем  $S_n^*$  полиномиального размера, но и для всех вообще последовательностей стратегий  $S_n^*$ .

*Задача 41.* Докажите, что без ограничения общности можно считать, что интерактивный протокол привязки к биту в качестве ключа всегда выдает свои случайные биты и  $\sigma$ .

## 8.2 Протоколы привязки к строке

По аналогии с протоколами привязки к биту можно определить понятие протокола привязки к строке фиксированной или даже полиномиальной от  $n$  (параметра безопасности) длины. Пусть  $m(n)$  — произвольный полином. Протоколом привязки к строке длины  $m = m(n)$  называется тройка полиномиальных вероятностных алгоритмов  $\langle T, S, R \rangle$  (последний из них должен быть детерминированным), для которых выполнены следующие условия.

(а) Для любой последовательности схем  $\{T_n^*\}$  полиномиального размера с вероятностью, приблизительно равной 1 (при  $n \rightarrow \infty$ ), для любой строки  $\sigma$  длины  $m(n)$  алгоритм  $R$  на входе  $c_{T^*, S(\sigma)}$  и  $k_{T^*, S(\sigma)}$  выдает либо  $\sigma$ , либо  $\perp_R$  (для краткости мы опускаем параметр безопасности  $n$  в обозначениях).

(б) Алгоритм  $S$  имеет нулевое разглашение.

(в) Для любой последовательности схем  $\{S_n^*\}$  полиномиального размера пренебрежимо мала вероятность каждого из двух событий

- существуют такие слова  $k', k''$ , что результаты работы алгоритма  $R(c_{T, S^*}, k')$ ,  $R(c_{T, S^*}, k'')$  на одном и той и той же записи с разными ключами различны и каждое из них отлично от  $\perp_S$ ,

- существует  $k$ , для которого  $R(c_{T,S^*}, k) = \perp_R$ .

*Задача 42.* Докажите, что для любого протокола привязки к строке существует функция  $F(c)$  со значениями в множестве строк длины  $m(n)$  такая, что для любой последовательности стратегий  $\{S_n^*\}$  с приблизительно единичной вероятностью для всех  $k$  выполнено  $R(c_{T,S^*}, k) \in \{F(c_{T,S^*}), \perp_S\}$ .

Протокол привязки к строке можно построить из любого протокола  $(S, T, R)$  привязки к биту. Посылающий и получающий привязываются по очереди к каждому из битов этой строки. Другими словами, новая стратегия  $\tilde{S}$  есть  $p(n)$ -кратное последовательное повторение стратегии  $S$ , новая стратегия  $\tilde{T}$  есть  $p(n)$ -кратное последовательное повторение стратегии  $T$ , а новый алгоритм раскрытия сундучка есть  $p(n)$ -кратное применение старого алгоритма  $R$ . Если хотя бы в одном из применений  $R$  выдал  $\perp_S$  или  $\perp_T$ , то новый алгоритм выдает, соответственно,  $\perp_S$  или  $\perp_T$  (если в одном из применений было выдано  $\perp_S$ , а в другом  $\perp_R$ , то неважно, что выдать). Иначе надо выдать полученную битовую строку.

*Задача 43.* Докажите, что построенный протокол удовлетворяет требованиям (а), (б) и (в).



## Глава 9

# Протоколы бросания монетки

Представим себе, что Алиса и Боб, находясь в разных городах, хотят подбросить монетку, не имея общего источника случайности. Каждый из них может самостоятельно выполнить подбрасывание, но тогда не будет гарантии, что полученные ими результаты совпадут. А Алиса и Боб хотят прийти к одному и тому же выводу о результате произведенного бросания. Можно доверять бросить монетку Алисе, которая потом сообщит Бобу результат бросания. Этот протокол нехорош тем, что позволяет Алисе сжульничать. Например, он не годится для того, чтобы разыграть, кому достанется машина, находящаяся в общей собственности разводящихся и не доверяющих друг другу супругов. Проблему решают так называемые протоколы бросания монетки.

*Протоколом бросания монетки* называется тройка полиномиальных алгоритмов  $(A, B, J)$ . Алгоритмы  $A, B$  вероятностные интерактивные, каждый из них получает на вход параметр безопасности  $n$  в унарной записи, а выхода не имеет. Алгоритм  $J$  детерминированный (это важно) и неинтерактивный, он получает на вход пару, состоящую из параметра безопасности  $n$  в унарной записи и двоичной строки  $s$ , а на выход выдает один из четырех результатов  $0, 1, \perp_A, \perp_B$ .

Бросание монетки осуществляется следующим образом. Сначала Алиса и Боб выполняют алгоритмы  $A, B$ , соответственно, для достаточно большого параметра безопасности  $n$ . Эти алгоритмы общаются по телефону (без ограничения общности можно считать, что первое сообщение посылается алгоритмом  $A$ ). Потом каждый из них запускает алгоритм  $J$ , дав ему на вход запись телефонного общения  $s_{A,B}$  вместе с параметром безопасности. Алгоритм  $J$  определяет результат бросания. Результаты

$\perp_A, \perp_B$  означают, что Алиса или Боб заблокировали протокол.

Результат работы  $J(c_{A,B})$  алгоритма  $J$  на протоколе  $c_{A,B}$  общения  $A, B$  есть случайная величина (поскольку  $A, B$  вероятностные алгоритмы) с параметром  $n$ . Будем обозначать ее через  $r_{A,B}$ . В этих обозначениях мы опускаем вход алгоритмов  $A, B$ , который есть  $n$  в унарной записи. Полностью следовало бы писать  $c_{A(1^n), B(1^n)}$  и  $r_{A(1^n), B(1^n)}$  вместо  $c_{A,B}$  и  $r_{A,B}$ . Требуется, чтобы случайная величина  $r_{A,B}$  принимала каждое из значений  $0, 1$  с вероятностью приблизительно  $1/2$  (при стремлении  $n$  к бесконечности). Детерминированность алгоритма  $J$  обеспечивает, что Алиса и Боб приходят к одному и тому же выводу о результате выполненного бросания. Кроме того, требуется, чтобы если один из партнеров жульничает, и вместо положенного алгоритма  $A(1^n)$  или  $B(1^n)$  запускает некоторую полиномиального размера схему  $A_n^*$  или  $B_n^*$ , соответственно, то вероятности обоих результатов  $0, 1$  не превосходили  $1/2$  плюс некоторая пренебрежимо малая величина, а также, чтобы вероятность обвинить честного партнера в блокировке протокола была пренебрежимо мала.

Итак, мы называем тройку алгоритмов  $(A, B, J)$  вышеописанного вида *протоколом бросания монетки*, если выполнены следующие три требования.

(а) Вероятность каждого из двух событий,  $r_{A,B} = 0$ ,  $r_{A,B} = 1$ , приблизительно равна  $1/2$  (то разность вероятности и  $1/2$  стремится с ростом  $n$  к нулю быстрее любого обратного полинома).

(б) Для любой неравномерно полиномиальной стратегии  $B_n^*$  вероятность каждого из событий,  $r_{A,B^*} = 0$  и  $r_{A,B^*} = 1$  не превосходит  $1/2$  плюс некоторая пренебрежимо малая величина, а вероятность события  $r_{A,B^*} = \perp_A$  пренебрежимо мала.

(в) Для любой неравномерно полиномиальной стратегии  $A_n^*$  вероятность каждого из событий,  $r_{A^*,B} = 0$  и  $r_{A^*,B} = 1$  не превосходит  $1/2$  плюс некоторая пренебрежимо малая величина, а вероятность события  $r_{A^*,B} = \perp_B$  пренебрежимо мала.

*Замечание 4.* Как обычно, во всех трех условиях можно предполагать, что схемы  $B_n^*$  и  $A_n^*$  вероятностные — требования для вероятностных схем следуют из аналогичных требований для детерминированных. Поэтому требование (а) следует из условий (б) и (в), примененных к  $B_n^* = B(1^n)$  и  $A_n^* = A(1^n)$ , соответственно. Действительно, в силу (б) и (в), во-первых, вероятность каждого из событий  $r_{A,B} = \perp_A$ ,  $r_{A,B} = \perp_B$  пренебрежимо

мала, а, во-вторых, вероятность каждого из событий  $r_{A,B} = 0$ ,  $r_{A,B} = 1$  не может существенно превосходить  $1/2$ . Поскольку сумма вероятностей этих четырёх событий равна 1, каждое из двух последних событий происходит с вероятностью приблизительно  $1/2$ . Поэтому достаточно проверять только условия (б) и (в).

**Теорема 20.** *Если существует интерактивный протокол привязки к биту, то существует и протокол подбрасывания монетки по телефону.*

*Доказательство.* Пусть нам дан некоторый протокол  $(S, T, R)$  привязки к биту.

Алгоритмы  $A$  и  $B$  действуют так (см. Рис. 9.1). Сначала алгоритм  $A$  подбрасывает монетку один раз. Обозначим через  $\sigma$  случайный бит, полученный в результате бросания. Затем алгоритм  $A$  запускает алгоритм  $S(\sigma)$ , а алгоритм  $B$  — алгоритм  $T$ , и эти алгоритмы общаются друг с другом. После того, как общение закончится и алгоритм  $S(\sigma)$  выдаст некоторый ключ  $k$ , алгоритм  $B$  подбрасывает монетку один раз и посылает результат бросания  $\tau$  алгоритму  $A$ . Наконец, алгоритм  $A$  посылает алгоритму  $B$  ключ  $k$ .

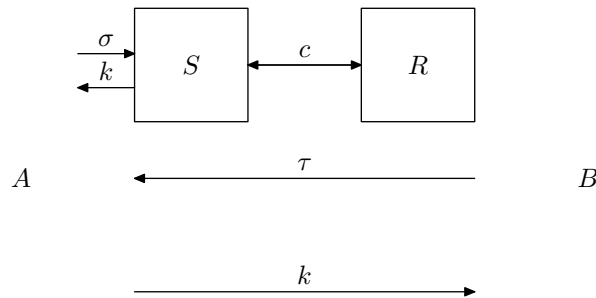


Рис. 9.1: Алгоритмы  $A$  и  $B$ .

Алгоритм  $J$  действует так. Пусть  $c$  обозначает протокол общения на первой стадии. (Чтобы, глядя на протокол, можно было понять, где заканчивается общение первой стадии, алгоритмы  $S$  и  $T$  надо модифицировать так, чтобы они приписывали ко всем сообщениям 0 слева. На второй стадии все сообщения будут начинаться на 1.) Запускаем алгоритм  $R$  на паре  $c, k$ , где  $k$  последнее сообщение Алисы. Объявляем, что Алиса или Боб блокировали протокол, если  $R$  выдаст соответствующее

сообщение. Иначе, выдаём сумму (по модулю 2) бита, посланного Бобом, и бита, выданного  $R$ .

Проверим, что этот протокол соблюдает интересы обоих игроков, то есть, условия (а) и (б).

(а) Интересы Алисы. Пусть Алиса действует честно, а Боб руководствуется некоторой неравномерно полиномиальной стратегией  $B_n^*$ . Это означает, что на первой стадии он выполняет некоторую неравномерно полиномиальную стратегию  $T_n^*$ , а на второй стадии применяет к протоколу беседы  $c$ , полученному на первой стадии, некоторую схему  $\tau_n^*$  полиномиального от  $n$  размера, выдающую на выход один бит. Поскольку на обеих стадиях Алиса действует честно, алгоритм  $S$  с приблизительно единичной вероятностью выдает бит  $\sigma$  или же объявит Боба блокирующим протокол (это следует из свойства (а) для протокола привязки). Значит вероятность того, что Алиса будет объявлена блокирующей протокол, пренебрежимо мала.

Теперь докажем, что вероятность обоих результатов 0,1 не превосходит  $1/2$  с точностью до пренебрежимо малой добавки. Это делается одинаково для 0 и 1, поэтому разберем случай 0. Будем обозначать результат работы схемы  $\tau^*$  на протоколе общения алгоритмов  $S(\sigma)$  и  $T^*$  через  $\tau(\sigma)$ . Алгоритм  $J$  выдаёт 0, если алгоритм  $R$  выдаст бит, равный  $\tau(\sigma)$ . В силу свойства (а) протокола привязки, с приблизительно единичной вероятностью алгоритм  $R$  выдаёт  $\sigma$  или  $\perp_R$ . Поэтому с точностью до пренебрежимо малой добавки вероятность результата 0 не превосходит вероятности того, что  $\tau(\sigma) = \sigma$  (с точностью до пренебрежимо малой величины). Так как Алиса выбирает свой бит  $\sigma$  случайно, вероятность этого события равна среднему арифметическому вероятности событий  $\tau(0) = 0$  и  $\tau(1) = 1$ . В силу условия (в) на протокол привязки, запись общения  $S(0)$  и  $T^*$  вычислительно неотличима от записи общения  $S(1)$  и  $T^*$ . В частности, они неотличимы и схемой  $\tau^*$ . Поэтому, вероятность события  $\tau(1) = 1$  приблизительно равна вероятности события  $\tau(0) = 1$ . Отсюда следует, что вероятность события  $\tau(\sigma) = \sigma$  приблизительно равна среднему арифметическому вероятности событий  $\tau(0) = 0$  и  $\tau(0) = 1$ , которая очевидно не больше  $1/2$ , поскольку эти события не пересекаются.

(б) Интересы Боба. Пусть Боб действует честно, а Алиса руководствуется некоторой неравномерно полиномиальной стратегией  $A_n^*$ . Это означает, что на первой стадии она выполняет некоторую неравномерно полиномиальную стратегию  $S_n^*$ , а на второй стадии применяет к прото-

колу беседы  $s$ , и биту  $\tau$ , посланному Бобом, некоторую схему  $k_n^*$  полиномиального от  $n$  размера. Тогда по свойству (б) протокола привязки, вероятность того, что алгоритм  $R$  объявит Боба блокировавшим протокол пренебрежимо мала.

Докажем, что вероятность обоих исходов 0,1 не превосходит  $1/2$  плюс пренебрежимо малая величина. Опять это делается одинаково для обоих исходов, поэтому разберем случай нуля. Ключ, выданный схемой  $k_n^*$  зависит от записи общения  $T$  и  $S^*$ , и от бита  $\tau$ . Нам будет важна главным образом зависимость от  $\tau$ , поэтому мы будем обозначать ключ, выданный схемой  $k_n^*$ , через  $k_\tau$ . Поскольку свой бит  $\tau$  Боб выбирает случайно, вероятность выдачи 0 равна среднему арифметическому двух событий: первое событие заключается в том, что алгоритм  $R$ , получив на вход запись общения  $T$  и  $S^*$  и ключ  $k_0$ , выдает 0, а второе — в том, что алгоритм  $R$ , получив на вход ту же запись общения, но другой ключ  $k_1$ , выдает 1. По свойству (б) протокола привязки, вероятность того, что для записи общения  $T$  и  $S^*$  существуют такие два ключа, пренебрежимо мала. Теперь воспользуемся очевидным свойством вероятностей: сумма вероятностей любых двух событий не превосходит 1 плюс вероятность их пересечения (в самом деле, эта сумма равна вероятности объединения событий плюс вероятность их пересечения, а вероятность объединения не может быть больше 1). Отсюда следует, что среднее арифметическое вероятностей указанных событий не превосходит  $1/2$  с точностью до пренебрежимо малой величины.  $\square$

С помощью протоколов бросания монетки можно играть в орлянку по телефону следующим образом. Для каждого  $n$  рассмотрим игру  $G_n$  между Алисой и Бобом. Количество ходов в этой игре равно полиному  $p(n)$ , ограничивающим время работы алгоритмов  $A$  и  $B$ . Каждый ход является словом длины не более  $p(n)$ , где  $p$  есть полином, ограничивающий время работы алгоритмов  $A$  и  $B$ . Игроки ходят по очереди, начиная с Алисы и игра продолжается до тех пор, пока один из ходов не будет пустым словом или не будет сделано  $p(n)$  ходов. Последовательность сделанных ходов дается на вход алгоритму  $J$ , который определяет, кто выиграл:  $0$  и  $\perp_B$  означают выигрыш Алисы, а  $1$  и  $\perp_A$  — выигрыш Боба. Из условий (а) и (б) следует, что у каждого из игроков имеется полиномиально вычислимая стратегия, гарантирующая ему выигрыш с вероятностью не менее  $1/2$  (минус пренебрежимо малая величина), если противник использует любую неравномерно полиномиальную стратегию.

По известной теореме Цермело [5], в любой игре двух игроков с конечным числом ходов и без ничьих один из игроков имеет детерминированную стратегию, которая обыгрывает любую стратегию противника. В частности, это верно и для игры  $G_n$  для каждого  $n$ . Поэтому в условиях (а) и (б) важно, что стратегии  $B_n^*$ ,  $A_n^*$  разрешаются не любые, а только неравномерно полиномиальные. В самом деле, если для бесконечно многих  $n$  в игре  $G_n$  Алисы имеет некоторую выигрышную стратегию  $A_n^*$ , то условие (б) не выполнено для последовательности стратегий  $A_n^*$  (для тех  $n$ , для которых выигрышной стратегии не существует, можно определить  $A_n^*$  произвольным образом). Действительно, в этом случае суммарная вероятность исходов  $0, \perp_B$  равна 1 для бесконечно многих  $n$  (а должно быть не больше  $1/2$  плюс пренебрежимо малая величина). А в противном случае для почти всех  $n$  в игре  $G_n$  выигрышная стратегия есть у Боба, и условие (а) не выполнено для некоторой последовательности стратегий Боба  $A_n^*$ .

*Задача 44.* Пусть в качестве протокола привязки в протоколе бросания монетки использован неинтерактивный протокол, основанный на однозначной односторонней функции. У кого из двух игроков есть выигрышная стратегия в игре  $G_n$  (без ограничения на равномерную полиномиальность стратегий игроков)?

*Задача 45.* Пусть в качестве протокола привязки в протоколе бросания монетки использован интерактивный протокол из предыдущего раздела, основанный на существовании генератора ПСЧ. У кого из двух игроков есть выигрышная стратегия в игре  $G_n$  (без ограничения на равномерную полиномиальность стратегий игроков)?

## Глава 10

# Неразглашение информации

Мы уже трижды встречались с понятием неразглашения информации. В определении схем шифрования с закрытым ключом мы требовали, чтобы алгоритм шифрования разглашал только длину сообщения. В определении схем шифрования с открытым ключом мы требовали, чтобы алгоритм шифрования разглашал только длину сообщения при условии, что открытый ключ общеизвестен. В определении протоколов привязки к биту мы требовали, чтобы алгоритм  $S$ , используемый Посылающим, не разглашал своего входного бита. В этом разделе мы дадим общее определение неразглашения информации вероятностными интерактивными алгоритмами.

Пусть имеется полиномиальный вероятностный интерактивный алгоритм  $A$ , получающий на вход параметр безопасности  $n$  в унарной записи и строку  $x$ . Неформально говоря, строка  $x$  секретна и алгоритм стремится при общении с любой схемой полиномиального от  $n$  размера сохранить ее в секрете при условии, что противник уже имеет некоторую информацию об использованных алгоритмом  $A$  случайных битах и самой строке  $x$ . (Например, противник может знать открытый ключ, использованный в схеме шифрования.) Точнее, пусть имеется некоторые функции  $f = f_n(x, r)$  и  $e = e_n(x, r)$  от параметра безопасности, входа и случайных битов алгоритма  $A$ . Пусть эти функции принимают значения среди строк полиномиальной от  $n, |x|, |r|$  длины. Мы хотим определить, что значит, что алгоритм  $A$  на входе  $x$  при использовании случайных битов  $r$  разглашает только  $f$ , если противнику уже известна информация  $e$ .

[Определение станет более естественным, если  $f = f_n(x, r)$  и  $e =$

$e_n(x, r)$  заменить на произвольные совместно распределенные случайные величины с параметром  $x$ .]

*Определение 4.* Мы говорим, что алгоритм  $A$  *разглашает только  $f$  при известном  $e$* , если существует полиномиальный вероятностный алгоритм  $M$  с оракулом, для которого выполнено следующее. Для любой последовательности слов  $\{x_n\}$  полиномиальной длины и любой последовательности схем  $\{C_n\}$  полиномиального размера, алгоритм  $M$ , получив на вход  $f$ , и используя в качестве оракула схему  $C_n^e$ , генерирует случайную величину, вычислительно неотличимую от записи общения между алгоритмом  $A(x_n)$  и схемой  $C_n^e$  при известном  $e$ .<sup>1</sup>

В этом определении под схемой  $C_n^e$  мы понимаем схему, полученную из  $C_n$  фиксацией первых  $|e|$  входов равными битами  $e$ . Случайные биты  $r$  алгоритма  $A$  выбираются с равномерным распределением среди строк той длины, которая требуется алгоритму  $A$  для работы на входе  $x_n$  (мы фиксируем полином от  $n, |x|$ , ограничивающий время работы алгоритма  $M$ , и длина  $r$  равна этому полиному). Случайные величины  $e, f$  и диалог  $A(x_n)$  со схемой  $C^e$  используют одну и ту же строку  $r$ , и поэтому могут быть зависимыми.

Алгоритм  $M$ , удовлетворяющий этому определению, будем называть *симулятором*. В определении важно, что симулятору разрешается не просто общаться со схемой  $C$ , как бы это делал алгоритм  $A$ , а запрашивать значение  $C$  на произвольных словах. Алгоритм  $A$  запрашивает только значения на словах вида  $ec_1, ec_1c_2, ec_1c_2c_3, \dots$ .

Кванторы всеобщности по последовательностям слов и схем полиномиального размера можно заменить на кванторы всеобщности по словам данной длины и схемам данного размера. Для этого в определении в соответствующем месте надо сказать так: для любых полиномов  $p, q, r$  найдется пренебрежимо малая величина  $\alpha_n$  с таким свойством: для любого слова  $x$  длины не более  $p(n)$  и любых схем  $C, T$  размера не более  $q(n), r(n)$ , соответственно, вероятности событий “ $T(e, \text{диалог } A(x) \text{ и } C^e) = 1$ ” и “ $T(e, M^{C^e}(f(x, e))) = 1$ ” отличаются не более чем на  $\alpha_n$ .

Определение неразглашения тем сильнее, чем меньше информации в  $f$  и больше в  $e$ . Это означает, что для любых полиномиально вычислимых функций  $U, V$ , если  $A$  разглашает только  $U(f)$  при известном  $e$ , то  $A$

<sup>1</sup>Напомним, что случайные величины  $\alpha_n$  и  $\beta_n$  называются вычислительно неотличимыми при известном  $\gamma_n$ , если случайные величины  $\alpha_n\gamma_n$  и  $\beta_n\gamma_n$  вычислительно неотличимы.



разглашает только  $f$  при известном  $V(e)$ .

[Надо подробно объяснить, почему это верно.]

В определении разглашения можно считать, что симулятору дается в качестве оракула схема  $C$ , а не  $C^e$ . Определение станет более слабым, однако все результаты сохранятся. Можно и еще больше ослабить определение, разрешив симулятору использовать схему, как дополнительный вход (наряду с  $f$ ), а не как оракул.

*Определение 5.* Если  $A$  разглашает только  $f$  при известном  $e$  и при этом функция  $e$  тождественно равна пустому слову, то мы просто говорим, что  $A$  разглашает только  $f$ . Если  $A$  разглашает только  $e$  при известном  $e$  (то есть  $f$  и  $e$  — это одна и та же функция), то мы говорим, что  $A$  имеет *нулевое разглашение при известном  $e$* .

*Пример 1.* Алгоритм  $A$  посылает побитовую сумму (по модулю 2) слова  $x$  и случайной строки  $r$  той же длины и затем останавливается. Тогда алгоритм разглашает только длину слова  $x$ . Симулятор выдает на выход случайную строку длины  $|x|$  (схему  $C$  он не использует, поскольку алгоритм  $A$  в сущности не интерактивный и поэтому запись общения  $A$  и  $C$  не зависит от  $C$ ).

Этот пример очень похож на схему шифрования с помощью гаммирования ( $x$  есть шифруемое сообщение, а  $r$  — закрытый ключ). Вообще, в любой схеме  $K, E, D$  шифрования с закрытым ключом алгоритм  $E(K(r), x)$  разглашает только длину шифруемого сообщения  $x$ . Действительно, по условию шифрограмма  $E(K(r), x)$  сообщения  $x$  вычислительно неотличима от шифрограммы  $E(K(r), 00 \dots 0)$  сообщения той же длины. Поэтому симулятор  $M$ , зная длину зашифрованного сообщения  $l$ , выбирает случайное  $r$  подходящей длины и выдает на выход  $E(K(r), 0^l)$ . Схему  $C$  он не использует, поскольку алгоритм  $A$  опять не интерактивный.

Этот пример показывает, что в определении разглашения важно, что случайные биты  $r$  алгоритма  $A$  и его вход  $x$  трактуются по-разному. А именно, в определении разглашения требуется, чтобы для любого  $x$ , случайная величина, равная диалогу между  $A$  и  $C$  при случайном выборе  $r$ , могла быть порождена по  $f$ . В примере 1 невозможно породить диалог для любых  $x, r$ , зная только длину  $x$ .

Различие в трактовке  $x$  и  $r$  можно увидеть и на более простом примере. Рассмотрим вероятностный алгоритм, игнорирующий свой вход и печатающий случайную строку длины  $n$ . Он имеет нулевое разглашение,

однако алгоритм, который печатает на выход свой вход длины  $n$  не имеет нулевого разглашения.

Как показывает следующая задача, обратное преобразование обычного входа в случайный сохраняет неразглашение.

*Задача 46.* Пусть вероятностный алгоритм  $A$  на входе  $x$  использует в качестве случайности независимые строки  $r, s$  каких-то длин  $l(|x|), k(|x|)$ . Преобразуем его в вероятностный алгоритм  $\tilde{A}$ , который на входе  $x, r$  ( $|r| = l(|x|)$ ) выбирает случайным образом строку  $s$  длины  $k(|x|)$  и выполняет алгоритм  $A$ . Докажите, что если  $\tilde{A}$  разглашает только  $f$  при известном  $e$ , то и  $A$  разглашает только  $f$  при известном  $e$ . Докажите, что полученный алгоритм так же разглашает только  $f$  при известном  $e$ .

*Пример 2.* Пусть дана любая схема шифрования с открытым ключом  $K, E, D$ . Обозначим через  $e(t)$  открытый ключ, полученный при использовании алгоритмом  $K$  генерации ключей случайных битов  $t$ . Тогда алгоритм  $x, r, t \mapsto E(e(t), r, x)$  (где  $x$  — это вход алгоритма, а  $r, t$  — случайные биты) разглашает только длину  $x$  и  $e$  при известном  $e$ . Действительно, по условию  $E(e, r, x)$  вычислительно неотличима от  $E(e, r, 0^{|x|})$  при известном  $e$ . Поэтому симулятор  $M$ , зная длину зашифрованного сообщения  $l$  и открытый ключ  $e$ , выбирает случайное  $r$  подходящей длины и выдает на выход  $E(e, r, 0^l)$ .

Как мы видели, всякая схема шифрования с открытым ключом может считаться схемой шифрования с закрытым ключом (если взять в качестве нового закрытого ключа пару из старого закрытого и открытого ключей). Это в частности означает, что если шифрограмма разглашает только открытый ключ и длину сообщения при известном открытом ключе, то она разглашает только длину сообщения. Следующая задача обобщает это свойство неразглашения.

*Задача 47.* Пусть вероятностный алгоритм  $A$  использует в качестве случайности две независимых строки  $r, s$  и разглашает только  $f(x, r)e(f(x, r), s)$  при известном  $g(x, r)e(f(x, r), s)$ , где  $f, g$  — произвольные функции, а  $e$  — любая полиномиально вычислимая функция. Тогда алгоритм  $A$  разглашает только  $f$  при известном  $g$ . (Указанное свойство шифрограмм получится, если в качестве  $f$  взять длину  $x$ , положить  $g$  тождественно равной пустому слову, а в качестве  $e = e(s)$  взять открытый ключ.)

*Пример 3.* В любом интерактивном протоколе  $S, R, T$  привязки алгоритм  $S$  имеет нулевое разглашение. (Мы считаем, что алгоритм  $S$  воспринимает любой вход  $x$  длины, отличной от 1, как, скажем, нулевой бит.)

Симулятор, имея схему  $C$ , моделирует общение алгоритма  $S(0)$  и схемы  $C$  и выдает на выход запись их общения. В этом примере симулятор запускает схему  $C$  в “штатном режиме”, то есть, общается с ней. В других примерах будет использоваться и нештатный режим.

Следующая задача показывает, что нулевое разглашение при известном  $e$  немонотонно по  $e$ .

*Задача 48.* Рассмотрим алгоритм  $A$ , получив входной бит  $x$ , складывает его со случайным битом  $r$  и выдает полученный бит на выход (если на входе не ровно один бит, то просто выдается случайный бит). Докажите, что этот алгоритм имеет нулевое разглашение, но не имеет нулевого разглашения при известном  $r$ , и имеет нулевое разглашение при известном  $r, x$ . (Последнее верно вообще для любого полиномиального алгоритма.)

Заметим, что в этой задаче в двух случаях функция  $e$  существенно зависит от случайных битов ( $e = r$  и  $e = (r, x)$ ). Это не случайно, поскольку если  $e$  не зависит от случайных битов алгоритма, то алгоритм разглашает только  $f$  при известном  $e$  тогда и только тогда, когда он просто разглашает только  $f$ . Действительно, разница между разглашением при известном  $e$  и просто разглашением в том, что в первом из них схематический отличитель настоящего диалога от смоделированного получает  $e(x_n)$  на вход, а во втором — нет. Но поскольку  $e(x_n)$  становится фиксированным после выбора последовательности  $x_n$ , эта разница не имеет значения, так как  $e(x_n)$  можно запаковать в отличающую схему.

Бывают алгоритмы, которые не разглашают информацию только общаясь с определенными собеседниками (а не с любыми схемами полиномиального размера). Дадим определение этому понятию.

*Определение 6.* Пусть  $B$  вероятностный интерактивный алгоритм, который получает на вход параметр безопасности и  $e(x, r)$ . Мы говорим, что алгоритм  $A$  *разглашает только  $f$  при известном  $e$  в беседе с  $B$* , если существует полиномиальный вероятностный алгоритм  $S$ , для которого выполнено следующее. Для любой последовательности слов  $\{x_n\}$  полиномиальной длины алгоритм  $S$ , получив на вход  $f(x, r)$  (мы опускаем индекс  $n$  у  $x$  и  $f$ ), порождает случайную величину, вычислительно неотличимую от записи общения алгоритмов  $A(x, r)$  и  $B(e(x, r), s)$  при известных  $e(x, r), s$ . Здесь  $s$  обозначает случайные биты алгоритма  $B$ . В этом контексте случайная величина (запись общения алгоритмов  $A(x, r)$  и  $B(e(x, r), s)$ ,  $s, e$ ) обозначается через  $\text{view}_B(A(x), B(e))$ .

*Задача 49.* Докажите, что если вероятностный алгоритм  $A$  разглашает только  $f$  при известном  $e$ , то и для любого вероятностного полиномиального алгоритма  $B$  он в беседе с  $A$  разглашает только  $f$  при известном  $e$ .

## 10.1 Неразглашение информации при последовательном выполнении алгоритмов

Установим одно важное свойство понятия неразглашения. Для этого определим операцию последовательного запуска вероятностных интерактивных алгоритмов. Пусть  $A, B$  вероятностные полиномиальные алгоритмы с одним и тем же входом  $x$ . И пусть, кроме того, алгоритмы  $A, B$  получают на вход некоторую общую полиномиально моделируемую случайную величину  $e = e_n$ . Наконец, пусть алгоритмы  $A, B$  имеют свои независимые источники случайности  $r, s$ , которые независимы также и от  $e$ . Выполним сначала алгоритм  $A$  на входе  $x, e$ , а затем выполним алгоритм  $B$  на входе  $x, e$ . Полученный алгоритм обозначим через  $(A, B)$ . Его входом мы считаем пару  $x$ , а случайными битами —  $r, s$  и случайные биты, использованные при порождении  $e$ .

Такая ситуация возникает, например, при использовании алгоритма шифрования  $E$  из схемы шифрования с открытым ключом для последовательной шифровки двух сообщений  $y, z$  с помощью одного и того же открытого ключа: алгоритм  $A$  применяет алгоритм  $E$  к первой компоненте входа  $x = \langle y, z \rangle$ , а алгоритм  $B$  — ко второй.

**Теорема 21.** Пусть даны произвольные полиномиально вычислимые функции  $g = g_n(x, r, e)$  и  $h = h_n(x, s, e)$  от общего входа  $x$  алгоритмов  $A, B$ , из случайных битов  $r, s$  и  $e$ . Допустим, что алгоритм  $A$  разглашает только пару  $g, e$  при известном  $e$ , а алгоритм  $B$  разглашает только пару  $h, e$  при известном  $e$ . Тогда алгоритм  $(A, B)$  разглашает только тройку  $g, h, e$  при известном  $e$ .

*Доказательство.* Будем обозначать пару  $g, h$  через  $f$ . Зафиксируем симуляторы  $M, N$  для алгоритмов  $A, B$  и построим симулятор для алгоритма  $(A, B)$ . Новый симулятор должен, получив на вход  $f, e$ , и имея в качестве оракула некоторую схему  $C$ , сгенерировать случайную величину, вычислительно неотличимую от записи общения  $(A, B)$  и  $C^e$  при известном  $e$ .

## 10.1. НЕРАЗГЛАШЕНИЕ ИНФОРМАЦИИ ПРИ ПОСЛЕДОВАТЕЛЬНОМ ВЫПОЛНЕНИИ АЛГ.

В процессе беседы алгоритма  $A$  со схемой  $C^e$  делаются некоторые сообщения. Обозначим через  $u$  конкатенацию этих сообщений. С алгоритмом  $B$  уже беседует схема, в которой следующие  $|u|$  входов (после первых  $|e|$ ) зафиксированы равными битам  $u$ . Будем обозначать эту схему через  $C^{eu}$ . Диалог алгоритма  $(A, B)$  на входе  $x$  со схемой  $C^e$  имеет вид  $uD(C^{eu})$ , где  $u$  — диалог алгоритма  $A(x, e)$  и схемы  $C^e$ , а  $D(C^{eu})$  — диалог алгоритма  $B(x, e)$  и схемы  $C^{eu}$ . Обе случайные величины  $u, D(C^{eu})$ , разумеется, зависят и от  $x$ .

Нам нужно, имея на входе  $f$  и используя как оракул схему  $C^e$ , сгенерировать некоторую случайную величину, вычислительно неотличимую от  $uD(C^{eu})$  при известном  $e$ . Чтобы сгенерировать такую случайную величину, запускаем данный нам симулятор  $M$  на входе  $g, e$  с оракулом  $C^e$ . Он выдаст некоторую последовательность сообщений  $\tilde{u}$ . После этого запустим симулятор  $N$  на входе  $h, e$ , но на этот раз с оракулом  $C^{e\tilde{u}}$ . Он нам выдаст последовательность, которую мы обозначим через  $\tilde{D}(C^{e\tilde{u}})$ . Выдаём в качестве результата  $\tilde{u}\tilde{D}(C^{e\tilde{u}})$ .

Зафиксируем произвольную последовательность строк  $x_n$  полиномиальной длины. Нам нужно установить вычислительную неотличимость (при известном  $e$ ) случайных величин

$$\alpha = \tilde{u}\tilde{D}(C^{e\tilde{u}})$$

и

$$\gamma = uD(C^{eu}),$$

полученных фиксацией входов алгоритмов  $A, B$  как  $x_n$ . В силу транзитивности, достаточно доказать, что они обе неотличимы (при известном  $e$ ) от случайной величины

$$\beta = u\tilde{D}(C^{eu}).$$

Неотличимость  $\beta$  и  $\alpha$  доказывается так. Рассмотрим преобразование

$$ev \mapsto ev\tilde{D}(C^{ev}).$$

Это преобразование можно вычислить вероятностной схемой полиномиального размера: схема выбирает случайные биты  $s$  алгоритма  $B$ , вычисляет  $h(e, s)$ , и применяет симулятор  $N$  ко входу  $e, h$  со схемой  $C^{ev}$  в качестве оракула. В применении к случайной величине  $eu$  это преобразование даст случайную величину  $e\beta$ , а в применении к  $e\tilde{u}$  оно даст  $e\alpha$ .

Поскольку исходные случайные величины неотличимы, то таковы и  $e\alpha$  и  $e\beta$ .

Осталось доказать неотличимость  $\gamma$  и  $\beta$ . Нам достаточно доказать их неотличимость при любой фиксации случайных битов алгоритма  $A$ . После этой фиксации случайная величина  $u$  становится функцией  $u(e)$  от  $e$ , вычислимой схемой полиномиального размера. Поэтому достаточно доказать неотличимость случайных

$$eD(C^{eu}) \text{ и } e\tilde{D}(C^{eu}).$$

По условию, для любой последовательности схем  $F_n$  полиномиального размера случайные величины

$$eD(F^e) \text{ и } e\tilde{D}(F^e).$$

вычислительно неотличимы. Поэтому нам достаточно построить последовательность схем  $F_n$  полиномиального размера, для которой схема  $F^e$  реализует ту же функцию, что и схема  $C^{eu}$ :

$$C^{eu} \equiv F^e.$$

Поскольку по  $e$  можно вычислить  $u$  схемой полиномиального размера, такая последовательность схем  $F_n$  в самом деле существует.  $\square$

Аналогичное утверждение верно и для композиции произвольного полиномиального числа алгоритмов. Для его аккуратной формулировки, будем считать, что имеется один алгоритм  $A$ , который применяется  $k = \text{poly}(n)$  раз. Причем при  $i$ -ом применении он получает на вход  $x$  и ещё само число  $i$ . Например, при  $i = 1$  алгоритм  $A$  может применять ко входу некоторый алгоритм  $B$ , а при  $i = 2$  совсем другой алгоритм  $C$ , как в теореме 21. Итак пусть дан вероятностный интерактивный алгоритм  $A$ , получающий на вход строку  $x$ , число  $i$  и полиномиально моделируемую случайную величину  $e$ . Выполним  $k$  раз алгоритм  $A$  последовательно на входах  $(x, 1, e), \dots, (x, k, e)$  (строка  $e$  во всех запусках одна и та же), используя в  $i$ -ом запуске свежие случайные биты  $r_i$ . Будем обозначать полученный алгоритм через  $A^{k(n)}$ .

**Теорема 22.** Пусть  $g$  любая полиномиально вычислимая функция от  $x, i, r, e$  ( $i$  параметра безопасности). Если алгоритм  $A$  разглашает только  $g, e$  при известном  $e$ , то алгоритм  $A^{k(n)}$  при известном  $e$  разглашает только

$$f(x, e) = \langle g(x, 1, r_1, e), \dots, g(x, k, r_k, e), e \rangle.$$

## 10.1. НЕРАЗГЛАШЕНИЕ ИНФОРМАЦИИ ПРИ ПОСЛЕДОВАТЕЛЬНОМ ВЫПОЛНЕНИИ АЛГ.

*Доказательство.* Обозначим через  $M$  симулятор для алгоритма  $A$  и построим симулятор для алгоритма  $A^k$ . Новый симулятор должен, получив на вход  $f$  и имея в качестве оракула некоторую схему  $C$  полиномиального размера, породить случайную величину, при известном  $e$  вычислительно неотличимую от диалога алгоритма  $A^k(x, e)$  и схемы  $C^e$ . Будем обозначать через  $D_i(C)$  случайную величину, равную диалогу алгоритма  $A(x, i, r, e)$  со схемой  $C^e$  для случайно выбранных  $r, e$ . Рассмотрим также случайную величину  $\tilde{D}_i(C)$ , равную результату работы симулятора  $M$  с оракулом  $C^e$  на входе  $g(x, i, r, e)$  для случайно выбранных  $r, e$ .

Зафиксируем произвольную схему  $C$  (точнее последовательность схем  $C_n$  полиномиального размера). Диалог  $\gamma$  алгоритмов  $A^k(x, e)$  и  $C^e$  можно получить последовательным применением к пустой строке для  $j = 1, \dots, k$  вероятностные операторы

$$u \mapsto uD_j(C^{eu}). \quad (10.1)$$

Поэтому естественно предположить, что если вместо (10.1) применить к пустой строке “близкие” операторы

$$u \mapsto u\tilde{D}_j(C^{eu}), \quad (10.2)$$

то полученная случайная величина  $\alpha$  будет вычислительно неотличима от диалога  $A^k(x, e)$  и  $C^e$  (при известном  $e$ ). Заметим, что случайная величина  $\alpha$  может быть порождена только по  $f(x, e)$ .

Зафиксируем произвольную последовательность  $x_n$  слов полиномиальной длины для алгоритма  $A^k$  и докажем, что  $\alpha$  в самом деле неотличима от  $\gamma$  при известном  $e$ . Для этого рассмотрим гибридные случайные величины  $\beta_0, \dots, \beta_k$ . Случайная величина  $\beta_i$  определяется как результат применения к пустой строке последовательно операторов (10.1) для  $j = 1, \dots, i$  и затем операторов (10.2) для  $j = i + 1, \dots, k$ . В частности,  $\beta_0$  это то же самое, что  $\alpha$ , а  $\beta_k$  — то же самое, что  $\gamma$ . Поэтому нам достаточно доказать, что  $\beta_{i-1}$  вычислительно неотличима от  $\beta_i$  при известном  $e$  (точнее, это нужно доказать для любой последовательности индексов  $i_n$ , в дальнейшем мы считаем, что такая последовательность зафиксирована).

Обозначим через  $\delta$  случайную величину, полученную применением к пустой строке операторов (10.1) для  $j = 1, \dots, i - 1$ , а через  $F(u)$  обозначим случайную величину, полученную из строки  $u$  путем применения операторов (10.2) для  $j = i + 1, \dots, k$ . В этих обозначениях

$$\beta_i = F(\delta D_i(C^{e\delta})), \quad \beta_{i-1} = F(\delta \tilde{D}_i(C^{e\delta})).$$

Поскольку оператор  $F$  вычислим вероятностной схемой полиномиального размера по  $e, \delta$  (напомним, что  $x_n$  фиксировано, и его можно запаять в схему), нам достаточно доказать неотличимость  $\delta D_i(C^{e\delta})$  и  $\delta \tilde{D}_i(C^{e\delta})$  при известном  $e$ . Случайная величина  $\delta$  зависит от  $e$  и случайных битов, использованных алгоритмом  $A$  в первых его  $i - 1$  запусках. Нам достаточно доказать неотличимость  $\delta D_i(C^{e\delta})$  и  $\delta \tilde{D}_i(C^{e\delta})$  при известном  $e$  для любой фиксации этих битов. После такой фиксации  $\delta$  становится полиномиально вычислимой функцией от  $e$ , поэтому достаточно доказать неотличимость  $D_i(C^{e\delta})$  и  $\tilde{D}_i(C^{e\delta})$  (при известном  $e$ ). Преобразуем схему  $C^{e\delta}$  в схему  $F$  полиномиального размера, для которой  $F^e \equiv C^{e\delta}$ . (Точнее, речь идет о последовательности схем  $F_n$  полиномиального размера — по одной схеме для каждого значения параметра безопасности.) Тогда

$$D_i(C^{e\delta}) = D_i(F^e), \quad \tilde{D}_i(C^{e\delta}) = \tilde{D}_i(F^e).$$

По условию  $D_i(F^e)$  и  $\tilde{D}_i(F^e)$  вычислительно неотличимы при известном  $e$ , что завершает доказательство.  $\square$

*Замечание 5.* Теорему можно еще немного усилить, дав алгоритму  $A$  доступ к записи диалога, возникшего между предыдущими запусками  $A$  и собеседником. Эта запись предоставляется алгоритму наряду с обычным входом  $x$ . Функция  $g$  не должна зависеть от этого дополнительного входа. Доказательство переносится без изменений на этот случай.

Следующая задача показывает, что в теореме 21 важно, что известная информация включает в себя общую часть случайности алгоритмов  $A, B$ .

*Задача 50.* Пусть алгоритм  $A$  на входе длины  $n$  выбирает случайную строку длины  $n$  и выдает на выход побитовую сумму этой строки и входной строки. Пусть алгоритм  $B$  делает то же самое со своим входом, используя ту же случайную строку, что и алгоритм  $A$ . Докажите, что оба алгоритма имеют нулевое разглашение, но их последовательное выполнение не имеет нулевого разглашения.

В следующей задаче приводится аналогичный пример и для теоремы 22.

*Задача 51.* Пусть  $g_n : \{0, 1\}^n \rightarrow \{0, 1\}^n$  сильно односторонняя функция. Рассмотрим алгоритм, который, игнорируя свой вход, выбирает случайные строки  $x, y$  длины  $n$ , сообщает пару  $\langle y, y \odot x \rangle$  и останавливается. (а) Докажите, что этот алгоритм имеет нулевое разглашение при известном



## 10.2. ЗАЩИЩЕННОСТЬ СХЕМ ШИФРОВАНИЯ ОТНОСИТЕЛЬНО АТАКИ С ВЫБОРОМ СООБЩЕНИЙ

$g(x)$ . (б) Докажите, что алгоритм  $B$ , который  $2n$  раз запускает алгоритм  $A$  с одним и тем же  $x$  и независимыми  $y$ -ами не имеет нулевого разглашения при известном  $g(x)$ .

### 10.2 Защищенность схем шифрования относительно атаки с выбором сообщений

Более общий вид атаки, выдерживаемой любой многоразовой СШЗК выглядит так. Пусть противник с помощью, используя схему  $C_n$ , зависящую от параметра безопасности  $n$ , размер которой ограничен полиномом от  $n$  играет с нами в следующую игру. На каждом ходу либо предлагает нам зашифровать один наш (секретный) бит, либо один свой бит, который даёт он нам передаёт. Игра продолжается полиномиальное число ходов  $p(n)$ . Противник узнаёт последовательность шифрограмм  $E(e, x_1), \dots, E(e, x_{p(n)})$ . Можно показать, что эта последовательность не содержит никакой полезной информации для него. Точнее, существует полиномиальный (от параметра безопасности) алгоритм, который используя как оракул схему  $C_n$  может сгенерировать случайную величину  $\xi_n$ , которая вычислительно неотличима от случайной величины

$$E(e, x_1), \dots, E(e, x_{p(n)}), x_{i_1} \dots x_{i_k},$$

где через  $x_{i_1} \dots x_{i_k}$  обозначены переданные противником биты для шифрования. В самом деле,

[Не закончено !!!!]



# Глава 11

## Протоколы идентификации

В протоколах идентификации участвуют две стороны, называемые Проверяющим и Доказывающим. Доказывающий должен убедить Проверяющего, что ему известна некоторая секретная строка, называемая закрытым ключом, не разгласив при этом эту строку.

### 11.1 Протоколы идентификации с закрытым ключом

Протоколом идентификации (с закрытым ключом) называется тройка полиномиальных вероятностных алгоритмов  $K, P, V$ . Все три алгоритма получают на вход параметр безопасности  $n$  и работают время, ограниченное полиномом от  $n$ . Алгоритм  $K$  генерирует строку  $e$ , называемых закрытым ключом. Алгоритмы  $P, V$  интерактивные и получают на вход, кроме параметра безопасности, ключ  $e$ . Алгоритм  $P$  выполняется Доказывающим, а алгоритм  $V$  — Проверяющим. Доказывающий, зная  $e$ , должен убедить Проверяющего, который тоже знает  $e$ , что он в самом деле его знает. При этом Доказывающий, не должен разгласить  $e$ , даже если Проверяющий жульничает. Формально, это выражается в следующих двух требованиях.

Требования таковы:

(а) С вероятностью, приблизительно равной 1, выполнено  $V^{P(e)}(e) = 1$ . Вероятность берется по распределению на ключах, порождаемому алгоритмом  $K$ , и по исходам случайных бросаний, выполняемых  $P$  и  $V$ .

(б) Пусть дана произвольная последовательность схем  $C_n$  полиномиального размера и произвольный полином  $k(n)$ . Рассмотрим следующую атаку: схема  $C_n$  беседует с алгоритмом  $P(e)^{k(n)}$  ( $k(n)$ -кратное последовательное выполнение независимых копий  $P$  с одним и тем же закрытым ключом). После этого схема пытается пройти идентификацию, беседуя с алгоритмом  $V(e)$ . Атака считается успешной, если алгоритм  $V(e)$  выдал 1. Требуется, чтобы вероятность успеха этой атаки была пренебрежимо мала. Вероятность считается по распределению на ключах, порождаемому алгоритмом  $K$  и по случайным выборам сделанным алгоритмами  $P, V$ .

Условие (б) имеет следующий смысл. Пусть противник изготовил фальшивый банкомат, который вместо правильной стратегии  $V$  использует некоторую неравномерно полиномиальную стратегию  $V^*$ , вычисляемую схемой полиномиального размера. Пусть владелец карточки  $k(n)$  раз прошел идентификацию в этом фальшивом банкомате. После этого, противник, используя полученную информацию, пытается сам пройти идентификацию в настоящем банкомате с помощью схемы полиномиального размера  $P^*$ . Тогда вероятность того, что противник успешно пройдет идентификацию пренебрежимо мала.

**Теорема 23.** *Если существует семейство ПСФ  $f_e^n : \{0, 1\}^n \rightarrow \{0, 1\}^n$ , удовлетворяющее усиленному условию на стр. 57, то существует протокол идентификации с закрытым ключом.*

*Доказательство.* Пусть длина идентификатора  $e$  ПСФ  $f_e^n$  равна  $l(n)$ . Алгоритм генерации ключей выбирает с равномерным распределением случайную строку  $e$  длины  $l(n)$ . Алгоритм  $V$  выбирает случайное  $x$  длины  $n$  и посылает его алгоритму  $P$ . Алгоритм  $P$  вычисляет  $f_e(x)$  и посылает его  $V$ . На этом общение заканчивается и алгоритм  $V$  выдаёт 1, если присланная ему строка совпадает с  $f_e(x)$ , и выдаёт 0 иначе.

Условие (а) очевидно выполнено. Проверим условие (б). Пусть дана последовательность схем  $C_n$ , атакующая схему идентификации и полином  $k(n)$ . Преобразуем схему  $C_n$  в схему  $D_n$ , тестирующую ПСФ. Схема  $D_n$  получает тестируемую функцию  $f_e : \{0, 1\}^n \rightarrow \{0, 1\}^n$  как внешнюю процедуру и моделирует атаку схемы  $C_n$ . А именно, сначала она моделирует общение схемы  $C_n$  с алгоритмом  $P^{k(n)}(e)$ . При этом всякий раз, когда алгоритм  $P(e)$  должен вычислить значение  $f_e(x)$  на запрошенном аргументе  $x$ , схема  $D_n$  вызывает внешнюю процедуру для вычисления этого значения. После  $k(n)$  повторений схема  $D_n$  моделирует общение  $C_n$

с  $V(e)$ . Для этого она выбирает теперь уже сама случайное  $x'$  длины  $n$  и подаёт его на вход схеме  $C_n$ . Схема  $C_n$  возвращает некоторое  $y$ . Наконец, схема  $D_n$  в последний раз запрашивает значение внешней функции  $f_e$  на  $x'$ . Если полученное значение совпадает с  $y$ , она выдаёт 1, а иначе 0.

По условию, вероятность ответа 1 изменится незначительно, если внешнюю функцию  $f_e$  заменить на случайно выбранную функцию  $g : \{0, 1\}^n \rightarrow \{0, 1\}^n$ . Убедимся, что для случайной функции вероятность выдать 1 пренебрежимо мала. Поскольку строка  $x'$  выбирается случайно, вероятность того, что она совпадёт с каким-то из  $x'$ ов, значение на которых схема  $C_n$  узнала у внешней процедуры, не превосходит  $k(n)/2^n$ . Если этого не произойдет, то  $g(x')$  является случайно выбранной независимо от  $y$  строкой, поэтому вероятность их совпадения равна  $2^{-n}$ .  $\square$

## 11.2 Протоколы идентификации с открытым ключом

### Протоколы, устойчивые против полу-честного противника

Протоколом идентификации с открытым ключом называется тройка полиномиальных вероятностных алгоритмов  $K, P, V$ . Все три алгоритма получают на вход параметр безопасности  $n$  и работают время, ограниченное полиномом от  $n$ . Алгоритм  $K$  генерирует пару строк  $e, d$ , называемых открытым и закрытым ключом. Алгоритмы  $P, V$  интерактивные и получают на вход, кроме параметра безопасности, еще одну строку — алгоритм  $P$  получает закрытый ключ, а алгоритм  $V$  — открытый ключ.

Требования таковы:

(а) С вероятностью, приблизительно равной 1, выполнено  $V^{P(d)}(e) = 1$ . Вероятность берется по распределению на ключах, порождаемому алгоритмом  $K$ , и по исходам случайных бросаний, выполняемых  $P$  и  $V$ .

(б) Для любой последовательности схем  $C_n$  полиномиального размера вероятность события  $V^{C_n(e)}(e) = 1$  пренебрежимо мала. Вероятность берется по случайному выбору  $e$  и по исходам случайных бросаний, выполняемых  $V$ .

**Задача 52.** Построить протокол идентификации, удовлетворяющий этим двум требованиям, на основе любой сильно односторонней функции.

[Здесь надо сформулировать усиление требования (б): “полу-честный” противник, подсматривает полиномиальное количество актов идентификации, а потом идентифицируется сам. В качестве задачи построить такие протоколы на основе односторонней перестановки с секретом (банк посылает образ некоторого элемента, а мы должны найти прообраз) или на основе протокола шифрования с открытым ключом: банк посылает нам шифрограмму случайного сообщения, а мы должны его расшифровать.]

### Протоколы, устойчивые против нечестного противника

Теперь усилим требование (б).

(бу) Пусть дана произвольная последовательность схем  $C_n$  полиномиального размера и произвольный полином  $k(n)$ . Рассмотрим следующую атаку: атакующая схема  $C_n$ , получив на вход  $e$  беседует с алгоритмом  $P(d)^{k(n)}$  ( $k(n)$ -кратное последовательное выполнение независимых копий  $P$  с одним и тем же закрытым ключом). После этого схема пытается пройти идентификацию, беседуя с алгоритмом  $V(e)$ . Атака считается успешной, если алгоритм  $V(e)$  выдал 1. Требуется, чтобы вероятность успеха этой атаки была пренебрежимо мала. Вероятность считается по распределению на ключах, порождаемому алгоритмом  $K$  и по случайным выборам, сделанным алгоритмами  $P, V$ .

*Задача 53.* Объясните, почему для протокола, основанного на односторонней функции с секретом, условие (бу) может не выполняться.

Протокол идентификации можно построить на основе функции Рабина, RSA или дискретной экспоненты (при предположении их необратимости).

Условие (бу) будет следовать из условия (б) и следующего условия (в) Алгоритм  $P$  имеет нулевое разглашение при известном  $e$ .

**Лемма.** *Свойство (бу) следует из (б) и (в).*

*Доказательство.* Пусть  $C_n$  — произвольная последовательность схем полиномиального размера в условии (бу). По условию (в) и теореме 22 алгоритм  $P^k$  имеет нулевое разглашение при известном  $e$ . Обозначим через  $M$  симулятор для алгоритма  $P^k$ . Применим его к открытому ключу

## 11.2. ПРОТОКОЛЫ ИДЕНТИФИКАЦИИ С ОТКРЫТЫМ КЛЮЧОМ<sup>103</sup>

$e$  с оракулом  $C^e$ . Он выдаст случайную величину  $\tilde{u}$ , вычислительно неотличимую от записи общения  $u$  схемы  $C^e$  и алгоритма  $P^k(d)$  при известном  $e$ . Рассмотрим теперь следующую попытку их отличить. Получив на вход  $\langle v, e \rangle$ , запускаем беседу стратегий  $V(e)$  и  $C^{ev}$  и выдаем в качестве результата ответ  $V$ . Этот отличитель задается схемой полиномиального размера. Вероятность того, что он выдаст 1 на входе  $\langle \tilde{u}, e \rangle$  пренебрежимо мала по свойству (б). Из неотличимости  $\langle u, e \rangle$  и  $\langle \tilde{u}, e \rangle$  следует, что и вероятность того, что он на входе  $\langle u, e \rangle$  выдаст 1, также пренебрежимо мала. Это и надо было доказать.  $\square$

Мы изложим метод, которым можно построить протокол идентификации с открытым ключом на основе любой из трех предположительно односторонних перестановок, функции Рабина, RSA или дискретной экспоненты. Изложим её, например, для функции Рабина.

**Теорема 24.** *Если функция Рабина необратима, то существует протокол идентификации.*

*Доказательство.* Напомним, что функция Рабина  $f$  определена как

$$f_n(m, x) = \langle x^2 \bmod m, m \rangle,$$

где  $m = pq$  и  $p, q$  — простые  $n$ -битные числа, а  $x$  —  $2n$ -битное число, взаимно простое с  $m$ . Алгоритм  $K$  на входе  $1^n$  выбирает случайные  $m, x$  из области определения  $f_n$ . Закрытый ключ  $d$  равен паре  $\langle m, x \rangle$  а открытый ключ  $e$  равен паре  $\langle m, x^2 \bmod m \rangle$ .

Алгоритмы  $P, V$  работают так.

- 1) Алгоритм  $P$  выбирает случайный обратимый вычет  $y$  и посылает  $z = y^2 \bmod m$  (с пренебрежимо малой вероятностью ему не удастся найти такого вычета, в этом случае он останавливает беседу).
- 2) Алгоритм  $V$  посылает случайный бит  $\alpha$ .
- 3) Алгоритм  $P$  проверяет, является ли присланное ему одним битом (если это не так, то в качестве  $\alpha$  он берет первый бит полученного слова) и посылает  $u = yx^\alpha$ . Заметим, что  $u^2 = zx^{2\alpha} \pmod{m}$ .
- 4) Алгоритм  $V$  проверяет это равенство (он может это сделать, поскольку знает  $x^2, m$ ) и проверяет, взаимно просты ли  $u$  и  $m$ . Если что либо из этого не выполнено, он останавливает беседу досрочно и выдает 0. Иначе повторяются пункты 1)–3), причем каждая из сторон использует новые случайные биты. Если после  $n$  повторений общение не остановлено досрочно, алгоритм  $V$  останавливает беседу и выдает 1.

Условие (а) очевидно выполнено.

Условие (в) проверяется следующим образом. По теореме 22 нам достаточно доказать, что в ходе однократного выполнения пунктов 1)–3) разглашается только  $m, x^2$  при известном  $m, x^2$ . Пусть банкомат использует схему  $F = F_n$ , которая по квадратичному вычету  $z$  сообщает бит  $F(z)$  (если  $F(z)$  состоит более чем из одного бита, то в дальнейших рассуждениях надо заменить  $F(z)$  на первый бит  $F(z)$ ). Нам надо, используя  $F$ , как оракул, сгенерировать случайную величину, вычислительно неотличимую от случайной величины

$$\langle y^2, F(y^2), yx^{F(y^2)} \rangle. \quad (11.1)$$

Здесь  $y$  выбирается по распределению, статистически неотличимому от равномерного распределения среди всех обратимых вычетов по модулю  $m$ . Если мы заменим распределение  $y$  на равномерное, то новое распределение на тройках  $\langle y^2, F(y^2), yx^{F(y^2)} \rangle$  будет статистически неотлично от старого, поэтому достаточно сгенерировать случайную величину, вычислительно неотличимую от (11.1) при равномерном распределении на  $y$ . На самом деле, мы сгенерируем величину, даже статистически неотличимую от неё.

Для этого заметим, что каждая тройка (11.1) имеет вид  $\langle z, \alpha, u \rangle$ , где  $F(z) = \alpha$ ,  $u^2 = zx^{2\alpha}$  и  $u$  взаимно просто с  $m$ . Будем тройки такого вида называть *удачными*. Естественно пытаться генерировать удачные тройки «с конца»: сначала выбрать случайным образом  $u$  (с равномерным распределением среди всех обратимых вычетов), затем выбрать случайным образом  $\alpha$  (с помощью симметричной монетки), а  $z$  положить равным  $u^2x^{-2\alpha}$ . Если нам повезет и полученная тройка удовлетворяет равенству  $F(z) = \alpha$ , то мы получим удачную тройку.

Нетрудно понять, что вероятность удачи в таком эксперименте равна  $1/2$ . В самом деле, случайные величины  $u^2x^{-2\alpha}$  и  $\alpha$  независимы: при любом  $\alpha$  случайная величина  $u^2x^{-2\alpha}$  имеет равномерное распределение среди всех обратимых квадратичных вычетов. Поэтому и случайные величины  $F(u^2x^{-2\alpha})$  и  $\alpha$  независимы. Отсюда следует, что вероятность события  $F(u^2x^{-2\alpha}) = \alpha$  равна  $1/2$ .

Будем повторять наш эксперимент с поиском удачной тройки, скажем  $n$  раз и выдавать первую найденную удачную тройку. Если же нам не повезёт все  $n$  раз, то выдадим любую тройку. В результате мы сгенерируем некоторое распределение на тройках, которое статистически неотлично от равномерного распределения на всех удачных тройках.



## 11.2. ПРОТОКОЛЫ ИДЕНТИФИКАЦИИ С ОТКРЫТЫМ КЛЮЧОМ 105

Осталось понять, что в беседе в самом деле появляется равномерное распределение на удачных тройках (точнее, статистически неотличимое от него). Тройка (11.1) целиком определяется выбором  $y$ , поэтому нам достаточно установить, что для любой удачной тройки  $\langle u^2x^{-2\alpha}, \alpha, u \rangle$  существует и притом единственное  $y$ , для которого

$$u^2x^{-2\alpha} = y^2, \quad \alpha = F(y^2), \quad u = yx^\alpha.$$

Единственность очевидна — последнее равенство однозначно задаёт  $y$ . Существование доказывается так: возведя последнее равенство в квадрат, мы получим первое равенство, а второе гарантировано удачностью исходной тройки  $\langle u^2x^{-2\alpha}, \alpha, u \rangle$ .

Осталось проверить условие (б'). Пусть для некоторой последовательности схем  $F = F_n$  вероятность события  $V^{F(m,x^2)}(m, x^2) = 1$  не пренебрежимо мала. Чтобы получить противоречие, нам достаточно, используя  $F_n$ , построить последовательность схем  $D_n$  полиномиального размера, которая для бесконечно многих  $n$  по паре  $m, x^2$  выдает некоторый корень из  $x^2$  с вероятностью не менее  $1/\text{poly}(n)$ .

Итак, нам известно, что для бесконечно многих  $n$  вероятность события  $V^{F(m,x^2)}(m, x^2) = 1$  больше  $\varepsilon = 1/\text{poly}(n)$ . Фиксируем одно из таких  $n$ . Сначала рассмотрим совсем простой случай: вероятность события  $V^{F(m,x^2)}(m, x^2) = 1$  равна 1. Схема  $F$  для любых данных  $m, x^2$  сначала посылает Проверяющему некоторый вычет  $z = z(m, x^2)$ . Затем в зависимости от присланного ему  $\alpha$  она присылает некоторый другой вычет. Будем обозначать вычет, присланный в ответ на  $\alpha = 0$  через  $u_0 = u_0(m, x^2)$ , а вычет, присланный в ответ на  $\alpha = 0$ , через  $u_1 = u_1(m, x^2)$ . По предположению с вероятностью 1 выполнено  $u_0^2 = z$ ,  $u_1^2 = zx^2$  и  $u_0, u_1$  взаимно просты с  $m$ , а следовательно,  $x^2 = (u_1/u_0)^2$ . Таким образом, следующий алгоритм обращает функцию Рабина с единичной вероятностью:

Для данных  $m, x^2$  применяем  $F$  к паре  $(m, x^2)$ . Обозначим через  $z$  полученный вычет. Применяем  $F$  к обоим тройкам  $((m, x^2), z, 0)$  и  $((m, x^2), z, 0)$ . Мы получим два вычета  $u_0, u_1$ . Выдаем в качестве ответа их частное.

Заметим, что реальный Доказывающий отказался бы в одном раунде отвечать на два разных  $\alpha$ . Когда мы обращаем функцию Рабина, мы используем схему, реализующую его стратегию, и можем делать это так, как запрещено в процессе идентификации.

Теперь рассмотрим чуть более сложный случай: пусть известно, что вероятность события  $V^{F(m,x^2)}(m, x^2) = 1$  не меньше  $3/4$ . В этом случае опять достаточно использовать, что с вероятностью не менее  $3/4$  беседа не будет остановлена Проверяющим в первом раунде. Среднее арифметическое вероятностей событий “ $u_0^2 = z$  и  $u_0$  взаимно просто с  $m$ ” и “ $u_1^2 = zx^2$  и  $u_1$  взаимно просто с  $m$ ” не меньше  $3/4$ . Значит сумма их вероятностей не меньше  $3/2$ , поэтому вероятность их пересечения не меньше  $1/2$ . Таким образом, с вероятностью не меньше  $1/2$  будет выполнено  $x^2 = (u_1/u_0)^2$ . Тот же самый алгоритм будет теперь обращать функцию Рабина с вероятностью не менее  $1/2$ .

Теперь рассмотрим общий случай. Обозначим через  $p_1$  вероятность того, что беседа не остановлена Проверяющим в первом раунде, а через  $p_i$  — вероятность того, что беседа не остановлена Проверяющим в  $i$ -ом раунде при условии, что она не остановлена раньше. Нам известно, что произведение чисел  $p_1, p_2, \dots, p_n$  не меньше  $\varepsilon = 1/\text{poly}(n)$ . Отсюда следует, что это произведение больше  $(3/4)^n$  при всех достаточно больших  $n$ . Значит, существует такое  $i$ , для которого  $p_i$  больше  $3/4$ . Зафиксируем любое такое  $i$ , и рассмотрим следующую вероятностную процедуру обращения функции Рабина.

Для данных  $m, x^2$ , моделируем диалог  $F$  и  $V$  и находим последовательность сообщений  $c$ , которая появится в беседе в первых  $i - 1$  раундах. При этом  $\alpha_1, \dots, \alpha_{i-1}$  (биты, посылаемые Проверяющим) выбираем случайным образом. Находим  $z = F(c)$ , и  $u_0 = F(c, z, 0)$  и  $u_1 = F(c, z, 1)$ . Выдаем в качестве ответа частное  $u_1/u_0$ .

Нам известно, что с вероятностью более  $1/2$ , вычеты  $u_0, u_1$  обратимы и квадрат их частного равен  $x^2$ , при условии, что беседа дойдет до  $i$ -ого раунда, не меньше  $1/2$ . Поскольку вероятность самого условия не меньше  $\varepsilon$ , безусловная вероятность того, что мы найдем корень из  $x^2$ , не меньше  $\varepsilon/2$ .

В приведенном рассуждении есть неточность — мы не знаем, чему равно  $i$ . Ее легко исправить: например, можно пробовать все  $i$ . А именно, для каждого  $i$  применять  $F$  к  $\alpha = 0$  и к  $\alpha = 1$ . Если нам не повезло и корень из  $x^2$  не найден, то выбираем  $\alpha$  случайно и переходим к следующему раунду. А можно рассуждать и так: поскольку обратитель является схемой полиномиального размера, которая не обязана вычисляться эффективно по  $n$ , можно запаковать  $i$  в схему-обратитель.  $\square$

*Замечание 6.* Как видно из доказательства, алгоритм  $P$  из протокола идентификации имеет нулевое разглашение в более сильном смысле, чем требует условие (в). А именно, нулевое разглашение имеет место даже, если открытый ключ считать частью входа (а не частью случайных битов).

*Задача 54.* Постройте аналогичные протоколы идентификации в предположении необратимости функции RSA и дискретной экспоненты.

### 11.3 Конструкция на основе любой односторонней функции

[Здесь надо изложить построение протокола идентификации с открытым ключом можно построить и на основе любой сильно односторонней функции. Открытым и закрытым ключом является пара  $f(x), x$  для некоторой сильно односторонней функции  $f$ . Идентификация состоит в доказательстве с нулевым разглашением знания прообраза  $f(x)$ .]



## Глава 12

# Доказательства знания с нулевым разглашением

Пусть даны произвольное полиномиально разрешимая последовательность отношений  $R_n \subset \{0, 1\}^{p(n)} \times \{0, 1\}^{q(n)}$ , где  $p, q$  — некоторые полиномы. Полиномиальная разрешимость означает, что за полиномиальное время можно по  $1^n, x, y$  выяснить, принадлежит ли пара  $(x, y)$  к  $R_n$  (то есть, верно ли  $R_n(x, y)$ ). Представим следующую ситуацию: у нас и нашего партнера имеется  $n$  некоторое слово  $y$ , а кроме этого, мы (но не партнер) знаем некоторое  $x$  для которого выполнено  $R_n(x, y)$  (будем такие  $x$  называть свидетелями для  $y$  об  $R$ ). Мы хотим доказать партнеру, что мы в самом деле знаем некоторого свидетеля для  $y$ . При этом мы хотим разгласить только  $y$  и ещё один бит, равный  $R(x, y)$ . Для этого мы и наш партнер выполняем некоторые полиномиальные вероятностные алгоритмы  $P(x, y)$  и  $V(y)$ , которые общаются между собой. После остановки второй из них должен сказать, убедился ли он, что алгоритм  $P(x, y)$  в самом деле знает некоторого свидетеля.

Перейдем к формальным определениям. Пусть дан вероятностный интерактивный полиномиальный алгоритм  $V$ , получающий на вход (кроме параметра безопасности) одну строку  $y$ , и вероятностный интерактивный полиномиальный алгоритм  $P$ , получающий на вход (кроме параметра безопасности) две строки  $x, y$ . Такая пара называется *интерактивным протоколом доказательства знания свидетелей об  $R$* , если выполнены следующие три условия

(а) Для всех  $y$  и всех свидетелей  $x$  для  $y$  об  $R$  с приблизительно единичной вероятностью алгоритм  $V(y)$  после общения с алгоритмом

$P(x, y)$  выдает 1.

(б) Алгоритм  $V$  проверяет знание свидетелей для  $R$ . Пусть дана любая последовательности схем  $C_n$  полиномиального размера. Обозначим через  $p_n(C_n, y)$  вероятность того, что  $V(y)$ , общаясь с  $C_n$ , выдает 1. Должен существовать полиномиальный вероятностный алгоритм  $M$  такой, что для любой последовательности  $C_n$  для всех  $n$  и всех  $y$  длины  $q(n)$  с вероятностью не менее  $p_n(C_n, y)/\text{poly}(n)$  (с точностью до некоторой пренебрежимо малой величины, зависящей от последовательности  $C_n$ ) алгоритм  $M$  на входе  $y$  с оракулом  $C_n$  находит некоторого свидетеля для  $y$ .

(в) Алгоритм  $P$  разглашает только пару  $x, R(x, y)$ .

В требовании (б) выражена возможность сведения задачи поиска свидетеля для  $y$  к задаче убедить  $V(y)$  выдать 1. Если последняя может быть решена схемой полиномиального размера с не пренебрежимо малой вероятностью, то то же самое верно и для первой задачи.

**Теорема 25.** *Если существует протокол привязки, то для любого  $R$  существует протокол доказательства знания свидетелей об  $R$ .*

Мы докажем эту теорему позже, а сейчас установим, как используя эту теорему можно построить протокол идентификации с открытым ключом.

**Теорема 26.** *Если существует сильно односторонняя функция, то существует и протокол идентификации с открытым ключом.*

*Доказательство.* Сначала заметим, что из условия следует существование протокола привязки (теорема 19), а значит (теорема 25) и существование протокола доказательства знания с нулевым разглашением для любого полиномиально разрешимого отношения  $R$ .

Пусть дана произвольная сильно односторонняя функция  $f_n : \{0, 1\}^{p(n)} \rightarrow \{0, 1\}^{q(n)}$ . Пусть  $R_n(x, y)$  истинно, если  $|x| = p(n)$ ,  $|y| = q(n)$  и  $f_n(x) = y$ . Пусть  $P, V$  протокол, существующий по этой теореме 25 для отношения  $R$ . Рассмотрим следующий протокол идентификации  $\tilde{K}, \tilde{P}, \tilde{V}$ : алгоритм генерации ключей  $\tilde{K}$  выбирает случайное  $x$  длины  $p(n)$  и выдает  $x$  в качестве закрытого ключа и  $f_n(x)$  в качестве открытого ключа. В качестве алгоритма  $\tilde{P}$  возьмем алгоритм, который на входе  $x$  сначала вычисляет  $y = f(x)$ , а затем выполняет алгоритм  $P$  на входе  $x, y$ . И наконец положим  $\tilde{V} = V$ .

Условие (а) для протокола доказательства знания эквивалентно условию (а) для построенного протокола идентификации.

Условие (в) для протокола доказательства знания влечет условие (в) для построенного протокола идентификации. В самом деле, рассмотрим следующий симулятор: получив открытый ключ  $y$ , добавим к нему бит 1 (поскольку свидетели есть для всех открытых ключей  $y$ ) и запустим на полученной паре симулятор из условия (в).

Докажем, что условие (б) для протокола доказательства знания влечет условие (б') для построенного протокола идентификации. Зафиксируем произвольную последовательность схем  $C_n$  полиномиального размера, для которой нужно установить, что вероятность события

$$V^{C(f(x))}(f(x)) = 1 \quad (12.1)$$

пренебрежимо мала. Обозначим через  $p(a)$  вероятность этого события при условии  $x = a$ . По условию при любом  $a$  алгоритм  $M^C$  обращает  $f(a)$  с вероятностью не менее  $p(a)/\text{poly}(n)$ . Усредняя по  $a$ , получаем, что вероятность события “ $M^C$  обращает случайное  $f(x)$ ” не меньше, чем вероятность события (12.1), делённая на  $\text{poly}(n)$ . Поскольку  $f$  необратима, вероятность первого события пренебрежимо мала, а значит такова же и вероятность события (12.1).  $\square$

## 12.1 Доказательство теоремы 25

Сначала докажем теорему 25 для конкретного отношения  $R$ , означающего что  $x$  является корректной раскраской неориентированный графа  $y$  с  $n$  вершинами. А потом используем NP полноту задачи поиска корректной раскраски.

Итак рассмотрим отношение  $R_n \subset \{0, 1\}^{p(n)} \times \{0, 1\}^{q(n)}$ , где  $q(n) = n(n-1)/2$  и строки  $y$  этой длины отождествляются с неориентированными графами  $G = (\{1, \dots, n\}, E)$  с  $n$  вершинами. Строка  $x$  имеет длину  $p(n) = 2n$  и задает произвольную раскраску вершин графа в три цвета 1, 2, 3 (двух битов достаточно для задания цвета одной вершины). Отношение  $R(x, y)$  выполнено, если раскраска *корректна*, то есть концы любого ребра графа имеют разные цвета.

**Теорема 27.** *Если существует протокол привязки к строке длины 2, то существует интерактивный протокол с нулевым разглашением доказательства знания корректной раскраски графа в три цвета.*

*Доказательство.* Заметим, что протокол привязки к строке длины 2 можно использовать для привязки к одному из трех цветов (с помощью привязки к его двухбитной двоичной записи).

Итак, нам достаточно построить протокол  $(P, V)$ , имеющий следующие свойства.

(а) Если раскраска, данная Доказывающему, корректна, то с приблизительно единичной вероятностью Проверяющий выдаёт 1.

(б) Если нам дана в качестве оракула произвольная последовательность  $C_n$  схем полиномиального размера и граф  $G$  из  $n$  вершин, то с вероятностью не менее  $p/2$  (минус пренебрежимо малая величина) мы можем найти некоторую корректную раскраску  $G$ , где  $p$  есть вероятность того, что  $C_n$  заставит  $V(G)$  выдать 1.

(в) Алгоритм  $P(\phi)$  разглашает только  $R(\phi, G)$  (то есть, является ли раскраска  $\phi$  корректной).

Интерактивный протокол  $(P, V)$  состоит в следующем. Если данная алгоритму  $P$  раскраска  $\phi$  не корректна, то он сразу останавливает общение. Иначе алгоритм  $P$  выбирает случайную перестановку  $\pi$  множества цветов  $\{1, 2, 3\}$  и применяет ее к данной раскраске  $\phi$ . Полученную раскраску  $\pi\phi$  будем обозначать через  $\alpha$ . Затем  $P$  и  $V$  последовательно для каждой вершины  $u \in \{1, \dots, n\}$  запускают алгоритмы  $S, T$  привязки к цвету  $\alpha(u)$  вершины  $u$ . То есть, происходит беседа интерактивных алгоритмов  $S_\alpha = S_{\alpha(1)} \dots S_{\alpha(n)}$  и  $T^n$ . Обозначим для дальнейшего через  $k_u$  ключ, выданный  $u$ -ым запуском алгоритма  $S$  (на входе  $\alpha(u)$ ).

Затем алгоритм  $V$  выбирает случайным образом некоторое ребро  $(u, v)$  в графе и посылает Доказывающему пару  $(u, v)$ . В ответ алгоритм  $P$  посылает  $k_u, k_v$ . Алгоритм  $V$ , применяя алгоритм  $R$  раскрытия сундучка, проверяет что цвета  $\alpha(u)$  и  $\alpha(v)$  различны. Если они оказались совпадающими или алгоритм  $T$  сообщил  $\perp_S$  для  $u$  или  $v$ , то алгоритм  $V$  выдает 0 и останавливается.

Указанная процедура повторяется в  $2n|E|$  раз (одно ее применение мы будем называть раундом) и если ни в одном раунде  $V$  не выдал 0, то он выдает 1 и останавливается. (Конец описания интерактивного протокола.)

Требование (а) очевидно выполнено.

Проверим требование (б). По требованиям к протоколу привязки существует функция  $F(c)$  со значениями 1, 2, 3 такая, что для любой схемы  $S^*$  полиномиального размера с приблизительно единичной вероятностью



для всех  $k$  выполнено

$$T(c_{S^*,T}, k) \in \{F(c_{S^*,T}), \perp_S\}. \quad (12.2)$$

Поэтому мы можем считать, что в каждом раунде после стадии привязки имеется некоторая функция  $F : \{1, \dots, n\} \rightarrow \{1, 2, 3\}$  такая, что любой ключ  $k_u$  примененный для открытия сундучка номер  $u$ , открывает  $F(u)$  или  $\perp_S$ . Точнее, такая функция существует с приблизительно единичной вероятностью.

Пусть имеется схема  $C$ , которая убеждает  $V(G)$  выдать 1 с некоторой вероятностью  $p$ . Мы построим полиномиальный вероятностный алгоритм с оракулом  $C$ , который находит некоторую корректную раскраску в вероятностью примерно  $p/2$ . Этот алгоритм для каждого  $i = 1, \dots, 2n|E|$  моделирует общение  $V$  и  $C$ , в течение  $i - 1$  раундов, а в  $i$ -ом раунде после стадии привязки просит схему  $C$  по очереди раскрыть ключи для концов *всех* ребер. Обычное моделирование позволяет раскрыть ключи только для концов одного ребра. Но поскольку схема нам дана в виде внешней процедуры, мы можем сделать это для всех ребер. Затем применяем алгоритм  $R$  раскрытия сундучка для полученных ключей. Если мы таким образом получаем корректную раскраску (всех вершин положительной степени), то выдаем ее.

Будем предполагать, что  $p > e^{-n}$ , поскольку иначе доказывать нечего (если пренебрежимо малая величина в условии меньше  $e^{-n}/2$ ). Вероятность успеха атаки  $C$  равна произведению чисел  $p_1, p_2, \dots, p_{2n|E|}$ , где  $p_i$  вероятность того,  $i$ -ый раунд завершен успешно (Проверяющий не выдал 0), при условии, что все предыдущие раунды также завершились успешно. Нам известно, что произведение чисел  $p_1, p_2, \dots, p_{2n|E|}$  не меньше  $p > e^{-n}$ . Значит, существует такое  $i$ , для которого  $p_i \geq e^{-1/2|E|} > 1 - 1/2|E|$ . Докажем, что для этого  $i$  вероятность найти раскраску в  $i$  раунде не меньше  $p/2$ . А для этого достаточно доказать, что вероятность этого при условии, что все  $i - 1$  предыдущих раундов завершились успешно, не менее  $1/2$ .

Неравенство  $p_i > 1 - 1/2|E|$  означает, что условная вероятность неудачи атаки  $i$ -ом раунде меньше  $1/2|E|$ . Пусть  $F$  обозначает случайную величину, равную раскраске графа спрятанного в сундучках после стадии привязки в  $i$ -ом раунде (смотри начало доказательства). Будем называть ребро  $u, v$  плохим, если при посылке этого ребра схеме  $C$  она выдает ключи, при использовании которых не раскрываются  $F(u), F(v)$  или

$F(u) \neq F(v)$ . Вероятность того, что найдется плохое ребро не больше  $1/2$  (с точностью до пренебрежимо малой величины). Действительно, иначе, вероятность выбора плохого ребра была бы не меньше  $1/2|E|$ , а мы знаем, что она меньше этого. Поэтому, с вероятностью не менее  $1/2$ , узнав ключи для концов всех ребер, мы найдем корректную раскраску вершин положительной степени.

Осталось проверить свойство (в). Пусть Проверяющий вместо предписанной стратегии применяет некоторую последовательность схем  $V^*$  полиномиального размера. Будем обозначать через  $c_{P,V^*}$  запись общения  $P(\phi)$  и  $V^*(G)$ . Если раскраска  $\phi$  некорректна, то  $c_{P,V^*}$  состоит из пустого слова и может быть порождено по известной нам информации, что  $R(\phi, G)$  ложно. Пусть раскраска  $\phi$  корректна. Нам нужно придумать алгоритм с оракулом  $V^*$  порождающий случайную величину, вычислительно неотличимую от случайной величины  $c_{P,V^*}$ .

Пусть  $\beta$  произвольная, возможно некорректная, раскраска графа. Рассмотрим следующую стратегию Доказывающего, которую мы обозначим через  $P_\beta$ : запускаем алгоритмы  $S_\beta$  и  $V^*$ , и по получении ребра  $(u, v)$  выдаём пару ключей  $k_u, k_v$ . Таким образом, исходная стратегия Доказывающего  $P$  заключается в выполнении  $P_\alpha$  для случайной корректной раскраски  $\alpha$  вида  $\pi\phi$ .

Алгоритм порождения случайной величины, неотличимой от  $c_{P,V^*}$ , таков. Выберем случайную раскраску  $\beta$  (все раскраски, корректные и некорректные выбираются с одинаковой вероятностью) и смоделируем общение  $P_\beta$  с  $V^*$ . Скажем, что нам повезло, если  $V^*$  после выполнения алгоритма  $S_\beta$  выдаст ребро  $(u, v)$ , для которого  $\beta(u) \neq \beta(v)$ . В этом случае выдаём полученную запись беседы в качестве результата. Иначе повторяем то же самое, используя новую случайную раскраску  $\beta$ . И так действуем до тех пор, пока нам не повезет или не произойдет  $n$  неудачных попыток.

Докажем, что вероятность везения в каждой попытке примерно равна  $2/3$ , поэтому вероятность того, что во всех  $n$  попытках не повезло, пренебрежимо мала. Зафиксируем ребро  $(u, v)$  и два различных цвета  $a, b$ . Вероятность везения равна сумме по всем ребрам  $(u, v)$  и парам различных цветов  $a, b$  вероятности того, что  $V^*(c_{S_\beta, V^*}) = (u, v)$  (то есть,  $V^*$ , изучив запись своего общения с  $S_\beta$ , выдаёт  $(u, v)$ ) и при этом  $\beta(u) = a$ ,  $\beta(v) = b$ . Поскольку мы зафиксировали цвета вершин  $u$  и  $v$ , при подсчете этой вероятности алгоритм  $S_\beta$  может быть заменен на композицию

алгоритмов

$$S_{\beta(1)}, \dots, S_a, \dots, S_b, \dots, S_{\beta(n)}.$$

Будем эту композицию обозначать через  $S_{\beta abuv}$ . Поскольку события

$$V^*(c_{S_{\beta abuv}, V^*}) = (u, v) \quad (12.3)$$

и  $\beta(u) = a, \beta(v) = b$  независимы, вероятность их пересечения равна произведению их вероятностей, то есть, одной девятой вероятности события (12.3). Теперь воспользуемся тем, что каждая из стратегий  $S_{\beta(1)}, \dots, S_a, \dots, S_b, \dots, S_{\beta(n)}$  имеет нулевое разглашение, а значит и их композиция тоже. Следовательно, существует полиномиальный алгоритм с оракулом  $V^*$ , генерирующий случайную величину  $\theta$ , вычислительно неотличимую от протокола общения  $S_{\beta abuv}$  и  $V^*$ . Поэтому для любых  $\beta, a, b, u, v$  вероятность события (12.3) примерно равна вероятности того, что  $V^*(\theta) = (u, v)$ . Отсюда следует, что вероятность везения примерно равна одной девятой суммы по ребрам  $(u, v)$  и парам  $a, b$  разных цветов вероятности того, что  $V^*(\theta) = (u, v)$ . Действительно, в этой сумме полиномиальное от  $n$  количество слагаемых, поэтому от изменения каждого слагаемого на пренебрежимо малую величину сумма изменится незначительно. Сумма по всем  $(u, v)$  вероятности того, что  $V^*(\theta) = (u, v)$ , равна 1. Поскольку существует ровно 6 пар различных цветов  $a, b$ , мы в результате получаем примерно  $6 \cdot (1/9) = 2/3$ .

Итак, наш алгоритм порождает случайную величину, статистически неотличимую от следующей случайной величины  $\eta$ . Ее значение равно  $c_{P_\beta, V^*}$  при условии, что в раскраске  $\beta$  концы ребра  $V^*(c_{S_\beta, V^*})$  имеют различные цвета. То есть,

$$\Pr[\eta = d] \approx \Pr[c_{P_\beta, V^*} = d \mid (\exists u, v) V^*(c_{S_\beta, V^*}) = (u, v), \beta(u) \neq \beta(v)] \quad (12.4)$$

Пусть  $A = A_n$  произвольная последовательность схем полиномиального от  $n$  размера. Нам надо доказать, что вероятность события

$$A(c_{P_\alpha, V^*}) = 1 \quad (12.5)$$

примерно равна вероятности события  $A(\eta) = 1$ . Поскольку вероятность условия в (12.4) приблизительно равна  $2/3$ , вероятности события  $A(\eta) = 1$  примерно равна  $3/2$  вероятности события

$$A(c_{P_\beta, V^*}) = 1, \beta(u) \neq \beta(v). \quad (12.6)$$

Представим каждое из событий (12.5) и (12.6) как объединение полиномиального числа непересекающихся событий. А именно, для каждого ребра  $(u, v)$  и каждой пары различных цветов  $a, b$  рассмотрим пересечение события (12.5) и события “Проверяющий послал ребро  $u, v$  и концы этого ребра имеют цвета  $a, b$ ”:

$$V^*(c_{P_\alpha, V^*}) = (u, v), \quad \alpha(u) = a, \quad \alpha(v) = b. \quad (12.7)$$

Аналогичным образом, для каждого ребра  $(u, v)$  и каждой пары различных цветов  $a, b$  рассмотрим пересечение события (12.6) с аналогичным событием:

$$V^*(c_{P_\beta, V^*}) = (u, v), \quad \beta(u) = a, \quad \beta(v) = b. \quad (12.8)$$

Нам достаточно доказать, что при любых  $(u, v)$  и  $a \neq b$  вероятности событий  $(12.5) \cap (12.7)$  и  $(12.6) \cap (12.8)$  относятся примерно как 3 к 2.

Заметим, что вероятности событий  $\alpha(u) = a, \alpha(v) = b$  и  $\beta(u) = a, \beta(v) = b$  равны  $1/6$  и  $1/9$ , соответственно, за счет того, что  $\alpha$  есть композиция корректной раскраски и некоторой перестановки цветов, а  $\beta$  является случайной раскраской. То есть они относятся как раз, как 3 к 2. А ещё заметим, что при подсчёте вероятности события  $(12.6) \cap (12.8)$  мы можем не учитывать условия  $\beta(u) \neq \beta(v)$ , поскольку это условие следует из того, что  $\beta(u) = a, \beta(v) = b$ . Наконец заметим, что вероятность события  $(12.5) \cap (12.7)$  не изменится, если алгоритм  $P_\alpha$  заменить на алгоритм  $P_{\alpha abuv}$ , который выполняет алгоритмы  $S_{\alpha(1)}, \dots, S_a, \dots, S_b, \dots, S_{\beta(n)}$  а после получения ребра выдает ключи  $k_u, k_v$  (независимо от того, какое ребро пошлет Проверяющий). Аналогичную замену можно произвести в событии  $(12.6) \cap (12.8)$ . После такой замены условие  $\alpha(u) = a, \alpha(v) = b$ , имеющееся в событии  $(12.5) \cap (12.7)$  становится независимым от остальных двух условий. То же самое верно и для события  $(12.6) \cap (12.8)$ . Поэтому вероятность  $(12.5) \cap (12.7)$  равна одной шестой вероятности события

$$A(c_{P_{\alpha abuv}, V^*}) = 1, \quad V^*(c_{S_{\alpha abuv}, V^*}) = (u, v), \quad (12.9)$$

а  $(12.6) \cap (12.8)$  — одной девятой вероятности события

$$A(c_{P_{\beta abuv}, V^*}) = 1, \quad V^*(c_{S_{\beta abuv}, V^*}) = (u, v). \quad (12.10)$$

Следовательно, нам достаточно доказать, что вероятности событий (12.9) и (12.10) примерно равны.

В чём разница между событиями (12.9) и (12.10)? Разница только в способе выбора раскраски: раскраски  $\alpha$  и  $\beta$  выбирается по различным распределениям. Поэтому нам достаточно доказать, что вероятность события (12.10) при фиксированной раскраске  $\beta$  мало зависит от  $\beta$ . Точнее, нам достаточно доказать, что при фиксированной  $\beta$  вероятность события (12.10) равна (с точностью до пренебрежимо малой величины) некоторому числу  $p_{abuv}$ , которое не зависит от  $\beta$ . Это следует из того, что алгоритм  $S$  имеет нулевое разглашение. Отсюда и из теоремы 22 следует, что алгоритм  $S_{\beta abuv}$ , получающий на вход  $\beta, a, b, u, v$ , имеет нулевое разглашение. Однако для алгоритма  $P_{\beta abuv}$ , получающего на вход  $\beta, a, b, u, v$ , это неверно (он сообщает ключи  $k_u, k_v$ ). Докажем, что он разглашает только  $a, b, u, v$  и случайные биты алгоритмов  $S_a, S_b$ . В самом деле алгоритм  $P_{\beta abuv}$  можно представить как последовательное выполнение следующих алгоритмов  $A_1, \dots, A_{n+2}$ . Все алгоритмы получают на вход  $\beta, a, b, u, v$  и пару случайных строк  $r, s$ . Алгоритм  $A_i$  при  $i \leq n$  и  $i \neq u, v$  запускает  $S_{\beta(i)}$ , а алгоритмы  $A_u, A_v$  запускают  $S_a, S_b$ , используя в качестве случайности  $r, s$ , соответственно. Наконец алгоритмы  $A_{n+1}$  и  $A_{n+2}$  сообщают  $k_u$  и  $k_v$ , соответственно. Напомним, что по задаче 41 без ограничения общности можно считать, что ключ, выдаваемый алгоритмом  $S$ , состоит из его входа и случайных битов алгоритма  $S$ . А можно, по замечанию 5 считать, что алгоритмам  $A_{n+1}, A_{n+2}$  доступен диалог  $A_u, A_v$  и  $C$ , и поэтому они могут вычислить ключ  $k_u, k_v$ . Таким образом, мы представили  $P_{\beta abuv}$ , как последовательность независимо работающих алгоритмов, и можем применить к ним теорему 22. Поскольку алгоритм  $A_i$  разглашает только  $i, a, b, u, v, r, s$  (что следует из нулевого разглашения алгоритма  $S$ ), алгоритм  $P_{\beta abuv}$ , разглашает только  $a, b, u, v, r, s$ . Поэтому существует симулятор  $M$ , который по  $a, b, u, v, r, s$  и  $V^*$  выдает случайную величину, вычислительно неотличимую от  $c_{P_{\beta abuv}, V^*}$ . Это же будет верно и если  $r, s$  выбирать случайным образом.

Итак, рассмотрим тест, который получив на вход запись общения алгоритмов  $P_{\beta abuv}$  и  $V^*$  применяет к ней схему  $A$ , затем еще применяет схему  $V^*$  к той ее части, которая произошла на стадии привязки. Если схема  $A$  выдает 1 и схема  $V^*$  выдает ребро  $(u, v)$ , то тест выдает 1, а иначе — 0. Этот тест (как и любой тест, задаваемый схемой полиномиального размера) не может отличить исходную запись общения от результата работы симулятора. Но поскольку последний от раскраски  $\beta$  не зависит, то и вероятность события (12.10) от раскраски зависит лишь незначительно.

Теперь объясним как с помощью теории NP полноты доказать утверждение теоремы для любого полиномиально разрешимого  $R$ . Пусть дана полиномиально разрешимая последовательность отношений

$$R_n \subset \{0, 1\}^{p(n)} \times \{0, 1\}^{q(n)}.$$

Тогда можно указать три полиномиально вычислимые функции  $f_n, g_n, h_n$  с таким свойством. Функция  $f_n$  по любому  $y$  длины  $q(n)$  указывает некоторый граф  $G$  с  $r(n)$  вершинами (где  $r$  — некоторый полином). Функция  $g_n$  по любой паре  $x, y$  в отношении  $R_n$  выдает корректную раскраску графа  $f(y)$ . И наконец функция  $h_n$  по  $y$  и любой корректной раскраске графа  $f(y)$  указывает некоторого свидетеля  $x$  для  $y$ .

Пусть  $P, V$  интерактивный протокол доказательства знания раскраски графа. Рассмотрим протокол  $P(g(x, y), f(y)), V(f(y))$ . Точнее, если  $x, y$  находятся в отношении  $R$ , то Доказывающий запускает алгоритм  $P$  на входе  $g(x, y), f(y)$ , а иначе сразу останавливается. Проверяющий же просто запускает  $V(f(y))$ . Проверим условия (а), (б) и (в).

(а) Если  $x, y$  находятся в отношении  $R$ , то по условию  $g(x, y)$  корректная раскраска графа  $f(x)$ . Поэтому  $P(g(x, y))$  убеждает  $V(f(y))$  с примерно единичной вероятностью.

(б) Пусть дана схема  $C$  полиномиального размера, убеждающая  $V(f(y))$  выдать 1 с некоторой вероятностью  $p$ . Запустим симулятор на графе  $V(f(y))$ , взяв в качестве оракула схему  $C$ . С вероятностью примерно  $p/2$  мы найдем раскраску графа  $f(y)$ . Применяя к этой раскраске функцию  $h$  мы найдем некоторого свидетеля для  $y$ .

(в) По условию  $P(g(x, y), f(y))$  разглашает только  $f(y)$  и то, является ли  $g(x, y)$  раскраской графа  $f(y)$ . Мы запускаем  $P(g(x, y), f(y))$  только, если выполнено  $R(x, y)$ , а, следовательно,  $g(x, y)$  есть корректная раскраска графа  $f(y)$ . Поэтому, если  $R(x, y)$  верно, то мы можем запустить симулятор для  $P$  на входе  $f(y), 1$  и получить случайную величину, вычислительно неотличимую от диалога  $P(g(x, y), f(y))$  и любой схемы полиномиального размера. А если  $R(x, y)$  ложно, то просто выдаём пустую последовательность сообщений.  $\square$

# Глава 13

## Семейства хэш-функций

Говоря неформально, хэш-функцией называется функция  $h : \{0, 1\}^* \rightarrow \{0, 1\}^k$ , для которой трудно найти такие  $x_1 \neq x_2$ , что  $h(x_1) = h(x_2)$ . Любая такая пара слов  $\langle x_1, x_2 \rangle$  называется коллизией функции  $h$ . Поскольку мощность множества значений меньше мощности области определения, коллизии обязательно есть, но мы хотим, чтобы их было трудно обнаружить. Чтобы формализовать это понятие, будем говорить не об одной хэш-функции, а о семействе хэш-функций. Мы рассмотрим две разных формализации: сильную — семейства хэш-функций с трудно обнаруживаемыми коллизиями (СТОК) и слабую — универсальные семейства одно-сторонних хэш-функций (УСОХ).

### 13.1 Семейства хэш-функций с трудно обнаруживаемыми коллизиями

Будем рассматривать полиномиально вычислимые семейства функций вида  $h^n : \{0, 1\}^l \times \{0, 1\}^* \rightarrow \{0, 1\}^k$ , где  $l, k$  являются полиномами от  $n$ . Полиномиальная вычислимость означает, что по  $n, t, x$  за время ограниченное полиномом от  $n + |x|$  можно найти  $h^n(t, x)$ . При каждом фиксированном  $n$  и  $t$  из  $\{0, 1\}^l$  получаем функцию  $x \mapsto h^n(t, x)$ , которую мы будем обозначать через  $h_t^n$ . Число  $n$  называется параметром безопасности, а строка  $t$  — идентификатором хэш-функции. Пусть еще имеется доступная случайная величина  $\alpha_n$ , принимающая значения в  $\{0, 1\}^{l(n)}$  (то есть, распределение вероятности на идентификаторах хэш-функций). Пару, состоящую из  $h^n$  и  $\alpha_n$ , называют семейством хэш-функций с трудно об-

наружимыми коллизиями (СТОК), если для любой последовательности схем  $C_n$  размера  $\text{poly}(n)$  вероятность события “ $C_n(\alpha_n)$  есть коллизия для  $h_{\alpha_n}^n$ ” пренебрежимо мала.

Для построения СТОК нам понадобится вспомогательный примитив — семейства перестановок с трудно обнаружимыми зацеплениями (claw-free). Зацеплением функций  $f^0, f^1$  называется пара слов  $z, y$ , для которых  $f^0(z) = f^1(y)$ . Разрешается, чтобы  $z, y$  совпадали. Неформально, пара полиномиально вычислимых перестановок  $f_t^0, f_t^1$  одного и того же множества  $D_t$  называется семейство с трудно обнаружимыми зацеплениями (СТОЗ), если по  $t$  трудно найти зацепление функций  $f_t^0, f_t^1$ .

Пусть задана полиномиально вычислимая последовательность отображений  $f_n : \{0, 1\} \times \{0, 1\}^{l(n)} \times \{0, 1\}^{m(n)} \rightarrow \{0, 1\}^{k(n)}$ . Фиксируя  $n$  и последовательность  $t$  из  $l(n)$  битов, мы получим функции  $f_t^0(x) = f(0, t, x)$  и  $f_t^1(x) = f(1, t, x)$  (параметр безопасности в обозначениях опускаем). Пусть еще имеется доступная последовательность случайных величин  $\alpha_n$ , принимающая значения в  $\{0, 1\}^{l(n)}$ . И кроме того, для каждого возможного значения  $t$  случайной величины  $\alpha_n$  имеется множество  $D_t \subset \{0, 1\}^{m(n)}$ , на котором обе функции  $f_t^0, f_t^1$  являются перестановками. При этом требуется, чтобы существовал полиномиальный алгоритм, который по  $n$  и  $\alpha$  с приблизительно единичной вероятностью находит некоторый элемент  $\beta_\alpha$  из  $D_\alpha$ . Вероятность здесь берется по данному нам распределению случайной величины  $\alpha_n$ .

Будем такое семейство функций называть СТОЗ, если для любой последовательности схем  $C_n$  полиномиального размера вероятность того, что  $C_n$  на входе  $\alpha$  выдает пару слов  $y, z \in D_\alpha$ , являющуюся зацеплением функций,  $f_\alpha^0, f_\alpha^1$ , пренебрежимо мала. Вероятность здесь берется по данному нам распределению случайной величины  $\alpha_n$ .

Семейство с трудно обнаружимыми зацеплениями можно построить, если предположить сильную необратимость любой из трех функций: функции Рабина, функции RSA или дискретной экспоненты.

**Теорема 28.** *Если функция Рабина необратима, то существует СТОЗ.*

*Доказательство.* Случайная величина  $\alpha_n$  равномерно распределена среди всех  $(m, x)$  из области определения функции Рабина ( $m = pq$ , где  $p, q$  — простые  $n$ -битовые числа вида  $4k + 3$ , а  $x$  — обратимый квадратичный вычет по модулю  $m$ ). Множество  $D_{m,x}$  состоит из всех обратимых квадратичных вычетов по модулю  $m$ . Функция  $f_{m,x}^0(y)$  определена как



$xy^2 \bmod m$ , а  $f_{m,x}^1(y) = y^2 \bmod m$  (вторая функция от  $x$  не зависит). Парты  $m, x$  из области определения функции Рабина можно генерировать с почти равномерным распределением, причем по  $m$  можно найти элемент из  $D_{m,x}$  за полиномиальное время (скажем,  $1 \in D_{m,x}$ ). Поэтому все исходные условия выполнены.

Проверим условие трудности обнаружения зацеплений. Допустим, мы можем не с пренебрежимо малой вероятностью по паре  $(m, x)$  из области определения функции Рабина находить такие обратимые квадратичные вычеты  $y, z$ , для которых  $xy^2 \bmod m = z^2 \bmod m$ . Поскольку вычет  $(z/y) \bmod m$  является квадратичным и его квадрат равен  $x$ , с этой же вероятностью мы можем обращать функцию Рабина.  $\square$

*Задача 55.* Докажите, что существует СТОЗ, если функция RSA необратима или экспонента в поле вычетов по простому модулю необратима.

Осталось построить СТОК, исходя из СТОЗ. Пусть дано семейство перестановок  $f_t^0, f_t^1$  множества  $D_t$  с трудно обнаружимыми зацеплениями, причем распределение на  $t$  дается доступной случайной величиной  $\alpha$ . Сопоставим каждому слову  $x = x_1x_2 \dots x_m$  его префиксный код  $\hat{x} = x_1x_1x_2x_2 \dots x_mx_m01$ . Этот код обладает следующим свойством: если  $x \neq z$ , то ни одно из слов  $\hat{x}, \hat{z}$  не является началом другого. Рассмотрим следующее семейство хэш-функций

$$h_t(x) = f_t^{\hat{x}_1}(f_t^{\hat{x}_2}(\dots f_t^{\hat{x}_l}(\beta_t) \dots)),$$

где  $\hat{x}_1\hat{x}_2 \dots \hat{x}_l$  обозначает префиксный код  $x$ . Слово  $t$  выбирается по распределению  $\alpha$ , данному в определении СТОЗ.

**Теорема 29.** Построенное семейство функций является семейством с трудно обнаружимыми коллизиями.

*Доказательство.* Допустим, что схема  $S$  полиномиального размера с не пренебрежимо малой вероятностью находит коллизии у  $h_t(x)$ . То есть, она выдает пару слов  $x \neq z$ , для которых

$$f_t^{\hat{x}_1}(f_t^{\hat{x}_2}(\dots f_t^{\hat{x}_l}(\beta_t) \dots)) = f_t^{\hat{z}_1}(f_t^{\hat{z}_2}(\dots f_t^{\hat{z}_m}(\beta_t) \dots)).$$

Пусть  $i$  наименьшее число такое, что  $\hat{x}_i \neq \hat{z}_i$  (такое существует по свойству префиксного кода). Поскольку  $f_t^0, f_t^1$  — перестановки  $D_t$  и  $\hat{x}_1 = \hat{z}_1$ ,

$\dots, \hat{x}_{i-1} = \hat{z}_{i-1}$ , мы можем сократить это равенство слева на композицию функций  $f_t^{\hat{x}^1}, \dots, f_t^{\hat{x}^{i-1}}$ :

$$f_t^{\hat{x}^i}(f_t^{\hat{x}^{i+1}}(\dots f_t^{\hat{x}^1}(\beta_t) \dots)) = f_t^{\hat{z}^i}(f_t^{\hat{z}^{i+1}}(\dots f_t^{\hat{z}^m}(\beta_t) \dots)).$$

Значит слова  $f_t^{\hat{x}^{i+1}}(\dots f_t^{\hat{x}^1}(\beta_t) \dots)$  и  $f_t^{\hat{z}^{i+1}}(\dots f_t^{\hat{z}^m}(\beta_t) \dots)$  являются зацеплением функций  $f_t^0, f_t^1$ .

Чтобы найти эти слова, мы вычисляем  $f_t^{u_1}(f_t^{u_2}(\dots f_t^{u_k}(\beta_t) \dots))$  для всех суффиксов  $u$  слов  $\hat{x}, \hat{z}$  и перебором находим среди них зацепление.  $\square$

## 13.2 Универсальные семейства односторонних хэш-функций

Полиномиально вычисляемое отображение  $h^n : \{0, 1\}^{q(n)} \times \{0, 1\}^{p(n)} \rightarrow \{0, 1\}^n$  (здесь  $p(n), q(n)$  — некоторые многочлены) называется универсальным семейством односторонних хэш-функций (УСОХ), если существует полиномиально моделируемое распределению  $\mu_n$  на словах длины  $q(n)$  со следующим свойством. Для любой последовательности схем  $C_n$  размера  $\text{poly}(n)$  и любой последовательности слов  $x_n \in \{0, 1\}^{p(n)}$  вероятность события “пара слов  $\langle x_n, C_n(t) \rangle$  есть коллизия для  $h_t^n$  пренебрежимо мала. Здесь  $h_t^n$  обозначает функцию  $h^n(t, x)$  и  $t$  выбирается по распределению  $\mu_n$ .

Заметим, что по сравнению со СТОК в определении УСОХ два послабления: во-первых в области определения каждой хэш-функции семейства слова фиксированной длины, полиномиальной зависящей от параметра безопасности (а в определении СТОК — произвольной длины) и во-вторых от атакующего требуется предъявить первый элемент коллизии, не зная идентификатора хэш-функции (а в определении СТОК первый элемент коллизии мог зависеть от идентификатора). Если ограничить область определения каждой функции в любом СТОК словами полиномиальной (от параметра безопасности) длины (полином можно выбрать любым), то мы получим УСОХ. Поэтому неудивительно, что универсальные семейства односторонних хэш-функций можно построить при более слабых предположениях, чем СТОК. А именно, достаточно существования любой односторонней функции [4].

**Теорема 30.** *Если существуют частичная сильно односторонняя функция, то для любого полинома  $p(n)$  существует полином  $q(n)$  и УСОХ  $h^n : \{0, 1\}^{q(n)} \times \{0, 1\}^{p(n)} \rightarrow \{0, 1\}^n$ .*

Мы не приводим доказательства этой довольно сложной теоремы. Вместо этого мы приведем конструкцию УСОХ на основе односторонней перестановки  $f_n : \{0, 1\}^n \rightarrow \{0, 1\}^n$ .

**Теорема 31.** *Если существуют сильно односторонняя перестановка  $f_n : \{0, 1\}^n \rightarrow \{0, 1\}^n$ , то для любого полинома  $p(n)$  существует полином  $q(n)$  и УСОХ  $h^n : \{0, 1\}^{q(n)} \times \{0, 1\}^{p(n)} \rightarrow \{0, 1\}^n$ .*

*Доказательство.* Сначала построим УСОХ  $h_s : \{0, 1\}^{n+1} \rightarrow \{0, 1\}^n$ . Идентификатором хэш-функции будет невырожденная над полем мощности 2 матрица  $A$  порядка  $n \times (n + 1)$ , а сама хэш-функция  $h_A$  отображает  $x$  в  $Af_{n+1}(x)$ . (Матрица называется невырожденной, если ее ранг равен  $n$ , то есть, строки линейно независимы.) Распределение  $\mu$  — равномерное на невырожденных матрицах.

**Задача 56.** Докажите, что равномерное распределение на невырожденных матрицах порядка  $n \times (n + 1)$  доступно.

**Лемма.** *Если перестановка  $f_n : \{0, 1\}^n \rightarrow \{0, 1\}^n$  сильно односторонняя, то так определенное семейство  $h_A$  является УСОХ.*

*Доказательство.* Пусть нам дана схема  $C$  полиномиального от  $n$  размера и строка  $x$  длины  $n + 1$ , для которых вероятность события “ $C(A)$  есть коллизия для  $h_A$ ” не пренебрежимо мала. Будем с помощью  $C$  и  $x$  обращать  $f$  следующим образом.

Пусть дана строка  $y$  длины  $n + 1$ , подлежащая обращению. Сначала проверим, не является ли  $x$  обратным к  $y$ . Если да, то выдаем  $x$ . Иначе мы генерируем случайную матрицу  $A$  со свойством  $Af(x) = Ay$  и выдаем  $C(A)$  в качестве результата.

**Задача 57.** Докажите, что по ненулевому вектору  $z$  можно сгенерировать за полиномиальное время распределение, статистически неотличимое от равномерного распределения на невырожденных матрицах со свойством  $Az = 0$ .

Будем считать, что исходное данное выбирается среди всех строк  $y$ , отличных от  $f(x)$ . Очевидно, нам достаточно доказать, что вероятность успеха обращения по этому распределению не пренебрежимо мала. Для

этого надо связать эту вероятность с вероятностью того, что  $C$  находит коллизию  $h_A$ . Заметим, что если пара  $\langle z = C(A), x \rangle$  является коллизией для  $h_A$ , то  $Af(z) = Af(x) = Ay$ . В силу невырожденности  $A$  существует не более двух различных строк  $u$  с данным значением  $Au$ . Поэтому  $f(z)$  равно  $f(x)$  или  $y$  (напомним, что мы предполагаем, что  $f(x) \neq y$ ). Поскольку  $z$  не равно  $x$  и  $f$  инъективна, первый случай невозможен, то есть  $z$  является прообразом  $y$ .

Закончено ли доказательство? Еще нет, поскольку нам нужно проверить, что в результате двухступенчатого процесса (сначала выбираем случайное  $y \neq f(x)$ , затем выбираем случайную невырожденную матрицу  $A$ , удовлетворяющую уравнению  $Af(x) = Ay$ ) генерируется равномерное распределение на невырожденных матрицах. Если это в самом деле так, то вероятность успеха в обращении случайно взятого  $y \neq f(x)$  в точности равна вероятности успеха  $C$  в поиске коллизии.

Как найти вероятность того, что в результате указанного двухступенчатого процесса появится данная невырожденная матрица? Рассмотрим множество  $S$  всех пар  $(A, y)$ , удовлетворяющих нашим свойствам:  $Ay = Af(x)$ ,  $y \neq f(x)$ ,  $A$  не вырождена. Будем себе представлять эти пары расположенными на плоскости, где  $A$  откладывается по вертикальной оси, а  $y$  — по горизонтальной. В нашем двухступенчатом процессе генерируется равномерное распределение на этом множестве, поскольку все вертикальные сечения множества  $S$  содержат одинаковое число элементов (число невырожденных матриц, аннулирующих данный вектор, не зависит от этого вектора). С другой стороны все горизонтальные сечения множества  $S$  тоже равномошны (все горизонтальные сечения одноэлементны, поскольку для данной невырожденной матрицы  $A$  существует ровно один вектор  $y \neq f(x)$ , удовлетворяющий равенству  $Ay = Af(x)$ ). Поэтому при случайном равномерном выборе пары из  $S$  ее первая компонента имеет также равномерное распределение.  $\square$

Пусть теперь дан произвольный многочлен  $p(n) > n$  и УСОХ  $h_t^n : \{0, 1\}^{n+1} \rightarrow \{0, 1\}^n$ . Преобразуем его в семейство хэш-функций  $g_t^n : \{0, 1\}^{p(n)} \rightarrow \{0, 1\}^n$  следующим образом. Идентификатором  $\tilde{t}$  новой хэш-функции будет строка из  $m = k(n) - n$  идентификаторов  $t_{k-1}, \dots, t_n$  функций из исходного семейства, причем  $g_t(x)$  есть композиция функций  $g_{t_{k-1}}, \dots, g_{t_n}$ . Применение каждой из функций уменьшает длину исходной строки на единицу, поэтому полученная функция как раз отображает строки длины  $p(n)$  в строки длины  $n$ .

**Лемма.** Если исходное семейство  $h_t$  является УСОХ, то и построенное семейство  $g_{\bar{t}}$  является УСОХ.

*Доказательство.* Утверждение следует из следующего простого наблюдения. Пусть  $x \neq y$  есть коллизия  $g_{\bar{t}}$ , то есть,  $g_{\bar{t}}(x) = g_{\bar{t}}(y)$ . Будем вычислять левую и правую часть этого равенства последовательно: сначала применим к  $y$  и  $z$  функцию  $f_{t_{k-1}}$ , получим новую пару строк. Применим к этим строкам функцию  $f_{t_{k-2}}$  и так далее. Рассмотрим тот первый момент когда строки в паре стали совпадать. В этот момент мы получим коллизия одной функции  $f_{t_j}$  для некоторого  $j = n, \dots, k-1$ .

Теперь допустим, что некоторая схема  $C_n$  полиномиального размера для бесконечно многих  $n$  атакует семейство  $g_{\bar{t}}$  с вероятностью успеха не менее  $\varepsilon_n = 2/\text{poly}(n)$  и при этом первый элемент коллизии равен  $x_n$ . Тогда для каждого из этих  $n$  можно найдется  $j = n, \dots, p(n) - 1$ , для которого с вероятностью не менее  $\varepsilon_n/(p(n) - n)$  в указанном процессе будет найдена коллизия функции  $h_{t_j}^j$ . Таких  $j$  очевидно бесконечно много. Для каждого из них, зная  $x_n$  и  $C_n$  (размеры которых ограничены полиномом от  $j$ , поскольку  $j \geq n$ ), можно атаковать семейство  $h^j$  так.

В качестве первого элемента коллизии выдаем результат последовательного применения к  $x_n$  функций  $h_{t_{p(n)-1}}, \dots, h_{t_{j+1}}$ . При этом идентификаторы этих функций выбираем случайно. После получения случайного идентификатора  $j$ -ой функции  $t_j$ , применяем к нему схему  $C_n$  и выдаем результат в качестве второй строки коллизии. Вероятность успеха этой атаки не меньше  $\varepsilon_n/(p(n) - n)$ , поскольку все идентификаторы  $h_{t_{p(n)-1}}, \dots, h_{t_{j+1}}, h_{t_j}$  были выбраны случайно.

Полученная вероятностная атакующая схема обычным образом может быть преобразована в детерминированную. □

□



## Глава 14

# Протоколы электронной подписи с закрытым ключом

Протоколом электронной подписи с закрытым ключом называется последовательность тройка вероятностных полиномиальных алгоритмов  $K, S, V$ , называемыми алгоритмами генерации ключей, алгоритмом подписи и алгоритмом проверки, соответственно. Все алгоритмы получают на вход параметр безопасности  $n$  в унарной записи. Алгоритм  $K$  (получив на вход параметр безопасности) генерирует строку  $d$ , называемую закрытым ключом. Алгоритм подписи, кроме параметра безопасности, получает на вход еще пару строк — закрытый ключ  $d$  и сообщение  $x$  — и выдает на выход строку, называемую подписью. Алгоритм  $V$ , кроме параметра безопасности, он получает на вход три строки — ключ  $d$ , сообщение  $x$  и подпись  $s$  — и выдает 0 или 1 (отвергает или принимает подпись). Требования таковы:

(а) Корректность. Для любой последовательности слов  $x_n$  (длина слова  $x_n$  должна быть ограничена полиномом от  $n$ ) с вероятностью, приблизительно равной 1, выполнено  $V(d, x, S(d, x)) = 1$  (параметр безопасности  $n$  мы во всех обозначениях опускаем). Вероятность берется по распределению вероятностей ключах  $d$ , порождаемому алгоритмом  $K$ , и по бросаниям монетки, выполняемым алгоритмами  $S, V$ .

(б) Невозможность подделать подпись. Мы требуем защищенности протокола от следующей атаки. Противник просит Подписывающего подписать некоторое сообщение  $x_1 = C()$ . Подписывающий дает ему требуемую подпись  $s_1 = S(d, x_1)$ . Противник изучает ее и просит Подписывающего подписать другое сообщение  $x_2 = C(s_1)$ . Подписывающий

опять дает ему требуемую подпись  $s_2 = S(d, x_2)$ . Так продолжается  $k$  раз, где  $k$  ограничено полиномом от  $n$ . Изучив все  $k$  подписей  $s_1, \dots, s_k$  противник выдает сообщение  $x_{k+1} = C(s_1 \dots s_k)$  и фальшивую подпись  $s_{k+1} = C(d, s_1 \dots s_k)$ . Атака считается успешной, если  $x_{k+1}$  не равно ни одному из сообщений  $x_1, \dots, x_k$  и подпись  $s_{k+1}$  под ним принята алгоритмом  $V$ . Итак, требование состоит в следующем. Для любого полинома  $p$  и любой последовательности схем  $\{C_n\}$  полиномиального размера вероятность события

$$x_{k+1} \notin \{x_1, \dots, x_k\}, \quad V(e, x_{k+1}, s_{k+1}) = 1$$

пренебрежимо мала. Здесь  $k = p(n)$ , а  $x_1, \dots, x_k, x_{k+1}, s_{k+1}$  определяются рекурсивно (мы опускаем параметр  $n$ ):

$$\begin{aligned} x_1 &= C(), & s_1 &= S(d, x_1), \\ x_2 &= C(s_1), & s_2 &= S(d, x_2), \\ & \dots & & \\ x_k &= C(s_1, \dots, s_{k-1}), & s_k &= S(d, x_k), \\ x_{k+1} &= C(s_1, \dots, s_k), & s_{k+1} &= C(d, s_1, \dots, s_k). \end{aligned}$$

Протокол электронной подписи с закрытым ключом можно изготовить из семейства псевдослучайных функций.

[Начиная с этого места, надо переписать, поскольку здесь должна идти речь о протоколах подписи, а не идентификации.]

**Теорема 32.** *Если существует семейство ПСФ  $f_e^n : \{0, 1\}^n \rightarrow \{0, 1\}^n$ , удовлетворяющее усиленному условию на стр. 57, то существует протокол идентификации с закрытым ключом.*

*Доказательство.* Пусть длина идентификатора  $e$  ПСФ  $f_e^n$  равна  $l(n)$ . Алгоритм генерации ключей выбирает с равномерным распределением случайную строку  $e$  длины  $l(n)$ . Алгоритм  $V$  выбирает случайное  $x$  длины  $n$  и посылает его алгоритму  $P$ . Алгоритм  $P$  вычисляет  $f_e(x)$  и посылает его  $V$ . На этом общение заканчивается и алгоритм  $V$  выдаёт 1, если присланная ему строка совпадает с  $f_e(x)$ , и выдаёт 0 иначе.

Условие (а) очевидно выполнено. Проверим условие (б). Пусть дана последовательность схем  $C_n$ , атакующая схему идентификации и полином  $k(n)$ . Преобразуем схему  $C_n$  в схему  $D_n$ , тестирующую ПСФ. Схема  $D_n$  получает тестируемую функцию  $f_e : \{0, 1\}^n \rightarrow \{0, 1\}^n$  как внешнюю



процедуру и моделирует атаку схемы  $C_n$ . А именно, сначала она моделирует общение схемы  $C_n$  с алгоритмом  $P^{k(n)}(e)$ . При этом всякий раз, когда алгоритм  $P(e)$  должен вычислить значение  $f_e(x)$  на запрошенном аргументе  $x$ , схема  $D_n$  вызывает внешнюю процедуру для вычисления этого значения. После  $k(n)$  повторений схема  $D_n$  моделирует общение  $C_n$  с  $V(e)$ . Для этого она выбирает теперь уже сама случайное  $x'$  длины  $n$  и подаёт его на вход схеме  $C_n$ . Схема  $C_n$  возвращает некоторое  $y$ . Наконец, схема  $D_n$  в последний раз запрашивает значение внешней функции  $f_e$  на  $x'$ . Если полученное значение совпадает с  $y$ , она выдаёт 1, а иначе 0.

По условию, вероятность ответа 1 изменится незначительно, если внешнюю функцию  $f_e$  заменить на случайно выбранную функцию  $g : \{0, 1\}^n \rightarrow \{0, 1\}^n$ . Убедимся, что для случайной функции вероятность выдать 1 пренебрежимо мала. Поскольку строка  $x'$  выбирается случайно, вероятность того, что она совпадёт с каким-то из  $x'$ ов, значение на которых схема  $C_n$  узнала у внешней процедуры, не превосходит  $k(n)/2^n$ . Если этого не произойдет, то  $g(x')$  является случайно выбранной независимо от  $y$  строкой, поэтому вероятность их совпадения равна  $2^{-n}$ .  $\square$



## Глава 15

# Протоколы электронной подписи с открытым ключом

Протоколом подписи (произвольного количества сообщений произвольной длины) называется последовательность тройка вероятностных полиномиальных алгоритмов  $K, S, V$ , называемыми алгоритмами генерации ключей, алгоритмом подписи и алгоритмом проверки, соответственно. Все алгоритмы получают на вход параметр безопасности  $n$  в унарной записи. Алгоритм  $K$  (получив на вход параметр безопасности) генерирует пару строк  $e, d$  называемых открытым и закрытым ключами. Алгоритм подписи, кроме параметра безопасности, получает на вход еще пару строк — закрытый ключ  $d$  и сообщение  $x$  — и выдает на выход строку, называемую подписью. Алгоритм  $V$ , кроме параметра безопасности, он получает на вход три строки — открытый ключ  $e$ , сообщение  $x$  и подпись  $s$  — и выдает 0 или 1 (отвергает или принимает подпись). Требования таковы:

(а) Корректность. Для любой последовательности слов  $x_n$  (длина слова  $x_n$  должна быть ограничена полиномом от  $n$ ) с вероятностью, приблизительно равной 1, выполнено  $V(e, x, S(d, x)) = 1$  (параметр безопасности  $n$  мы во всех обозначениях опускаем). Вероятность берется по распределению вероятностей на паре ключей  $(e, d)$ , порождаемому алгоритмом  $K$ , и по бросаниям монетки, выполняемым алгоритмами  $S, V$ .

(б) Невозможность подделать подпись. Мы требуем защищенности протокола от следующей атаки. Противник, изучив открытый ключ  $e$ , просит Подписывающего подписать некоторое сообщение  $x_1 = C^e$ . Подписывающий дает ему требуемую подпись  $s_1 = S(d, x_1)$ . Противник изу-

чает ее и просит Подписывающего подписать другое сообщение  $x_2 = C(e, s_1)$ . Подписывающий опять дает ему требуемую подпись  $s_2 = S(d, x_2)$ . Так продолжается  $k$  раз, где  $k$  ограничено полиномом от  $n$ . Изучив все  $k$  подписей  $s_1, \dots, s_k$  противник выдает сообщение  $x_{k+1} = C(e, s_1 \dots s_k)$  и фальшивую подпись  $s_{k+1} = D(e, s_1 \dots s_k)$ . Атака считается успешной, если  $x_{k+1}$  не равно ни одному из сообщений  $x_1, \dots, x_k$  и подпись  $s_{k+1}$  под ним принята алгоритмом  $V$ . Итак, требование состоит в следующем. Для любой неравномерно полиномиальной стратегии  $C_n$ , любой неравномерно полиномиальной функции  $D_n$  и для любого полинома  $p$  для почти всех  $n$  вероятность события

$$x_{k+1} \notin \{x_1, \dots, x_k\}, \quad V(e, x_{k+1}, s_{k+1}) = 1$$

пренебрежимо мала. Здесь  $k = p(n)$ , а  $x_1, \dots, x_k, x_{k+1}, s_{k+1}$  определяются рекурсивно (мы опускаем параметр  $n$ ):

$$\begin{aligned} x_1 &= C(e), & s_1 &= S(d, x_1), \\ x_2 &= C(e, s_1), & s_2 &= S(d, x_2), \\ & & \dots & \\ x_k &= C(e, s_1, \dots, s_{k-1}), & s_k &= S(d, x_k), \\ x_{k+1} &= C(e, s_1, \dots, s_k), & s_{k+1} &= D(e, s_1, \dots, s_k). \end{aligned}$$

В чем отличие схем подписи от протоколов идентификации с открытым ключом? Главное отличие в том, что идентификация — это интерактивный процесс, в протоколах идентификации алгоритмы  $P, V$  интерактивные, а в протоколе подписи алгоритмы  $S, V$  — не интерактивные. По этой причине алгоритмы идентификации не применимы в построении схем подписи. В обратную же сторону сведение возможно: любую схему подписи можно переделать в протокол идентификации: Доказывающий подписывает случайное выбранное Проверяющим сообщение длины  $n$ .

*Задача 58.* Докажите, что этот протокол идентификации с открытым ключом удовлетворяет требованиям (а) и (б) к протоколам идентификации.

В дальнейшем мы будем накладывать на противника разные ограничения. Первое из них — полином ограничивающий  $k$  есть тождественно единичный полином. Другими словами, противнику разрешается получить настоящую подпись только под одним сообщением, а затем самостоятельно изготовить другое сообщение и фальшивую подпись под ним.

Протоколы подписи, устойчивые относительно такой атаки, называются протоколами *одноразовой* подписи или протоколами подписи одного сообщения (поскольку с гарантией можно подписать только одно сообщение). Второе ограничение, которое мы будем рассматривать, относится к длинам подписываемых сообщений. Пусть фиксирован некоторый полином  $p(n)$ . Протоколы подписи, которые устойчивы относительно атаки, в которой длины всех сообщений  $x_1, \dots, x_k, x_{k+1}$  равны  $p(n)$ , называются протоколами подписи сообщений длины  $p(n)$ . Самыми простыми будут одноразовые схемы подписи одного бита (то есть протоколы подписи сообщений длины 1).

## 15.1 Одноразовая схема подписи одного бита

Атака на такую схему подписи вырождается в следующее. Атакующая схема  $C_n$  сначала применяется к открытому ключу  $e$  и выдает некоторый бит  $\sigma$ . Затем схема  $C = C_n$  еще раз применяется к паре  $e, s$ , где  $s = S(d, \sigma)$  есть настоящая подпись под битом  $\sigma$ . Атака считается успешной, если выданная фальшивая подпись  $s' = C(e, s)$  признана правильной подписью под противоположным битом  $\bar{\sigma}$ , то есть,  $V(e, \bar{\sigma}, s') = 1$ . Требуется, чтобы вероятность успеха атаки была пренебрежимо мала (для любой последовательности схем  $\{C_n\}$  полиномиального от  $n$  размера).

**Теорема 33.** *Если существует односторонняя функция, то существует протокол подписи одного бита.*

*Доказательство.* Пусть  $f_n$  сильно односторонняя функция и  $G$  — алгоритм генерации распределения, статистически неотличимого от равномерного распределения в области определения  $f_n$ . В качестве закрытого ключа возьмем пару  $\langle x^0, x^1 \rangle$ , где  $x^0, x^1$  получены независимыми применениями алгоритма  $G$  к параметру безопасности  $n$ . В качестве открытого ключа возьмем  $\langle y^0, y^1 \rangle$ , где  $y^0 = f_n(x^0)$  и  $y^1 = f_n(x^1)$ . Таким образом, алгоритм генерации ключей  $K$  по существу заключается в двукратном выполнении алгоритма  $G$  и последующем применении алгоритма вычисления функции  $f_n$  к полученным результатам.

Подпись под битом 0 есть  $x^0$  (первый компонент закрытого ключа), а подпись под битом 1 есть  $x^1$  (второй компонент закрытого ключа). Проверяющий, проверяя подпись  $s$  под битом  $\sigma$ , применяет  $f_n$  к  $s$  и принима-

ет подпись, если  $f_n(s) = y^\sigma$ . Очевидно, что этот протокол удовлетворяет требованию (а).

Проверим требование (б). Допустим, что существуют последовательность схем  $C_n$  полиномиального от  $n$  размера, атакующая для бесконечно многих  $n$  построенную систему подписи с вероятностью успеха не меньше  $\varepsilon = 1/\text{poly}(n)$ . Зафиксируем любое из таких  $n$ . Противник, применяя схему  $C = C_n$  к открытому ключу  $y^0, y^1$  вычисляет бит  $\sigma = C(y^0, y^1)$ . Затем, получив от подписывающего  $x^\sigma$ , он применяет схему  $C$  к открытому ключу и  $x^\sigma$  и выдаёт поддельную подпись  $s'$ . Атака признаётся успешной, если  $y^{\bar{\sigma}} = f(s')$ .

Рассмотрим следующий вероятностный алгоритм  $D$  для обращения  $f$ . Получив на вход  $y = f(x)$ , выбираем случайным образом  $x'$  из области определения  $f$  (с помощью алгоритма  $G$ ), а также выбираем случайным образом бит  $\sigma$ . Затем составляем пару  $y_0, y_1$  положив  $y_\sigma = f(x')$ , а другой компонент пары положив равным данному нам  $y$ . Применим схему  $C$  ко входу  $(y_0, y_1)$ . Будем считать, что нам повезло, если схема выдала тот же бит  $\sigma$ , который мы выбрали случайно. В этом случае выдадим на выход  $C(y_0, y_1, x')$  (а если нам не повезло, то можно выдать что угодно).

Пусть исходное  $x$  выбирается случайно с помощью алгоритма  $G$ . Докажем, что вероятность того, что мы правильно обратили  $f(x)$  не менее  $\varepsilon/2$ . Для этого заметим, что, несмотря на то, что мы формировали пару  $(y_0, y_1)$ , исходя из значения  $\sigma$ , случайные величины  $\sigma, y_0, y_1$  независимы. И при этом случайная величина  $(y_0, y_1)$  распределена так же, как и открытый ключ в схеме цифровой подписи. Поэтому  $\sigma$  не зависит от бита, выданного схемой на входе  $(y_0, y_1)$ . Следовательно, при любых фиксированных  $y_0, y_1$  условная вероятность везения равна  $1/2$ . Поскольку  $(y_0, y_1)$  имеет то же распределение, что и открытый ключ, с вероятностью не менее  $\varepsilon/2$  атака схемы  $C$  на схему подписи окажется успешной и при этом нам повезет, то есть  $C(y_0, y_1, x')$  будет равно некоторому прообразу  $y$ .

Осталось заметить, что выполняемый нами алгоритм  $D$  может быть выполнен вероятностной схемой полиномиального размера.  $\square$

*Задача 59.* Рассмотрим вместо использованного в доказательстве теоремы алгоритма обращения  $f$  следующую его модификацию: как и раньше, выбираем случайно  $x'$ , а затем применяем схему  $C$  к обоим тройкам  $(y, f(x'), x')$  и  $(f(x'), y, x')$ . Если хотя бы одно из полученных слов будет прообразом  $y$  (мы можем это проверить полиномиальной схемой), выда-

дим это слово. Докажите, что вероятность успеха этого алгоритма также не меньше  $\varepsilon/2$ , причем в худшем случае она может быть примерно равна  $\varepsilon/2$ .

## 15.2 Подпись одного сообщения фиксированной длины

Пусть  $p(n)$  произвольный полином. Для протоколов подписи одного сообщения длины  $p(n)$  условие невозможности подделывания подписи вырождается в следующее требование. Для любой последовательности схем  $C_n, D_n, E_n$  полиномиального от  $n$  размера вероятность события

$$x' \neq x, \quad |x| = |x'| = p(n), \quad V(e, x', s') = 1$$

пренебрежимо мала (опускаем параметр  $n$ ). Здесь  $x = C(e)$ ,  $x' = E(e, s)$  и  $s' = D(e, s)$ , где  $s = S(d, x)$ .

**Теорема 34.** *Если существует протокол подписи одного бита, то для любого полинома  $p$  существует протокол подписи одного сообщения длины  $p(n)$ .*

*Доказательство.* Пусть  $\langle K, S, V \rangle$  данный нам протокол подписи одного бита. Рассмотрим “ $p(n)$ -ую степень этого протокола”, определяемую так: алгоритм генерации ключей  $\tilde{K}$  порождает  $p(n)$  пар независимых ключей. В качестве открытого ключа он выдает кортеж из полученных открытых ключей  $\tilde{e} = e^1 \dots e^{p(n)}$ , а в качестве закрытого ключа — кортеж из полученных закрытых ключей —  $\tilde{d} = d^1 \dots d^{p(n)}$ . Подпись  $\tilde{s}$  под сообщением  $x_1 \dots x_{p(n)}$  состоит из конкатенации подписей  $s_1, \dots, s_{p(n)}$  с помощью  $S$  под битами  $x_1, \dots, x_{p(n)}$ , причем при подписывании бита  $x_i$  используется ключ  $d^i$ . Алгоритм проверки  $\tilde{V}$  заключается в проверке всех подписей  $s_1, \dots, s_{p(n)}$  с открытыми ключами  $e^1, \dots, e^{p(n)}$ . Если все они выдержали проверку, то подпись признается, иначе отвергается.

Условие корректности очевидно выполнено. Условие невозможности подделать подпись проверяется так. Нам нужно атакующего  $\tilde{C}_n, \tilde{D}_n, \tilde{E}_n$  новую систему подписи преобразовать в атакующего исходную систему. Пусть вероятность успешной атаки  $\tilde{C}_n, \tilde{D}_n, \tilde{E}_n$  равна  $\varepsilon = 1/\text{poly}(n)$  для бесконечно многих  $n$ . Очевидно существует  $i \leq p(n)$ , для которого

с вероятностью не меньше  $\varepsilon/p(n)$  атака будет успешной и при этом  $i$ -ые биты  $x'$  и  $x$  различны. Успешность атаки означает, в частности, что  $V(e^i, x'[i], s_i) = 1$  (параметр  $n$  опускаем). Будем атаковать исходную систему подписи одного бита следующим образом. Нам дан открытый ключ  $e$ , причем закрытый ключ, соответствующий ему, неизвестен. Выбираем случайным образом  $p(n) - 1$  пару ключей. Даем полученную последовательность из  $p(n) - 1$  открытых ключей на вход схеме  $\tilde{C}$ , вставив в нее на  $i$ -ое место данный нам открытый ключ  $e$ . Схема  $\tilde{C}$  выдаст некоторое сообщение  $x$ . Подписываем все его биты, кроме  $i$ -ого, с помощью известных закрытых ключей. А  $i$ -ый бит просим подписать Подписывающего один бит с ключом  $d$  (неизвестным нам). Полученную последовательность из  $n$  подписей даем на вход схеме  $D$ . Она выдаст  $n$  поддельных подписей. Оставляем в этой последовательности только  $i$ -ую подпись  $s'_i$ . Эта подпись с вероятностью не менее  $\varepsilon/p(n)$  будет принята Проверяющим  $V$ , как подпись под  $x'[i] = \overline{x[i]}$  с ключом  $e$ .

Описанная атака использует случайные биты (при выборе  $p(n) - 1$  пар ключей). Зафиксировав их подходящим образом, мы можем получить детерминированные схемы с не меньшей вероятностью успеха.  $\square$

Эта система подписи является в самом деле всего лишь одноразовой в следующем смысле. Имея подписи под сообщениями  $00 \dots 0$  и  $11 \dots 1$ , можно изготовить подпись под любым сообщением.

### 15.3 Протокол подписи одного сообщения произвольной длины

Для протоколов одноразовой подписи сообщений произвольной длины условие невозможности подделывания подписи вырождается в следующее требование. Для любой последовательности схем  $C_n, D_n, E_n$  полиномиального от  $n$  размера вероятность события

$$x' \neq x, \quad V(e, x', s') = 1$$

пренебрежимо мала. Здесь  $x = C(e)$ ,  $x' = E(e, s)$ , и  $s' = D(e, s)$ , где  $s = S(d, x)$ .

Протокол подписи одного сообщения произвольной полиномиальной длины можно построить, имея протокол одноразовой подписи длины  $n$  с помощью и СТОК (или даже УСОХ).



## 15.4 Построение схемы подписи одного сообщения произвольной длины, исходя из семейств хэш-функций с трудно обнаружимыми коллизиями

Для преобразования системы подписи сообщений длины  $k(n)$  в систему подписи сообщений любой длины достаточно хэш-функций, значения которых имеют  $k(n)$  битов. Новый алгоритм подписи будет сначала применять хэш-функцию, а потом с помощью старого алгоритма подписывать полученное хэш-значение.

**Теорема 35.** *Если существует СТОК  $h^n : \{0, 1\}^{l(n)} \times \{0, 1\}^* \rightarrow \{0, 1\}^{k(n)}$  и система одноразовой подписи сообщений из  $k(n)$  битов, то существует и система одноразовой подписи сообщений произвольной длины.*

*Доказательство.* Пусть  $S, V$  данные нам в условии алгоритмы подписи и проверки,  $e_n, d_n$  — открытый и закрытый ключи для нее, а  $\alpha_n$  — случайная величина из определения СТОК  $h^n : \{0, 1\}^{l(n)} \times \{0, 1\}^* \rightarrow \{0, 1\}^{k(n)}$ .

Нам нужно построить одноразовую систему подписи  $(\langle \tilde{e}_n, \tilde{d}_n \rangle, \tilde{S}, \tilde{V})$  сообщения любой длины. В качестве открытого ключа возьмем пару  $\langle e_n, \alpha_n \rangle$ , состоящую из старого открытого ключа и идентификатора хэш-функции из семейства  $h_\alpha$ . Аналогично определим закрытый ключ:  $\tilde{d}_n = \langle d_n, \alpha_n \rangle$ .

Алгоритм подписи  $\tilde{S}$ : сначала применяем к данному сообщению  $x$  хэш-функцию  $h_\alpha$ , а затем подписываем старым алгоритмом  $S$  с ключом  $d$  полученное слово. То есть  $\tilde{S}(\tilde{d}, x) = S(d, h_\alpha(x))$ .

Алгоритм проверки  $\tilde{V}$ : сначала применяем к данному сообщению  $x$  хэш-функцию  $h_\alpha$ , а затем проверяем старым алгоритмом проверки данную подпись, как подпись под полученным словом. То есть  $\tilde{V}(\tilde{e}, x, s) = V(d, h_\alpha(x), s)$ .

Очевидно, что требование (а) выполнено. Проверим условие (б). Рассуждая от противного, допустим, что имеются последовательности схем  $C_n, E_n, D_n$  полиномиального размера, для которых для бесконечно многих  $n$  вероятность успеха атаки не меньше  $\varepsilon = 1/\text{poly}(n)$ . То есть с этой вероятностью случается событие

$$T_n = (x' \neq x \wedge V(e, h_\alpha(x'), s') = 1),$$

где  $x = C(e, \alpha)$ ,  $x' = E(e, \alpha, s)$ , и  $s' = D(e, \alpha, s)$ , где  $s = S(d, h_\alpha(x))$ . Вероятность здесь берется по случайному выбору  $e, d, \alpha$  и случайным датчикам внутри  $E, S$ .

Будем атаковать семейство хэш-функций или схему подписи  $(\langle e, d \rangle, S, V)$ . От чего зависит, что именно мы атакуем? Одно из двух событий

$$A_n = (x' \neq x \wedge h_\alpha(x') = h_\alpha(x))$$

и

$$B_n = (h_\alpha(x') \neq h_\alpha(x) \wedge V(e, h_\alpha(x'), s') = 1)$$

имеет вероятность не менее  $\varepsilon/2$ . Действительно, вместе они покрывают событие  $T_n$ , которое имеет вероятность не менее  $\varepsilon$ . Либо для бесконечно многих  $n$  вероятность  $A_n$  не меньше  $\varepsilon/2$ , либо для бесконечно многих  $n$  вероятность  $B_n$  не меньше  $\varepsilon/2$  (либо и то, и другое). В первом случае мы получим противоречие с тем, что семейство хэш-функций является СТОК, а во втором, получим противоречие с надежностью данной нам системы подписи.

Семейство хэш-функций атакуется следующим образом (в первом случае). Получив идентификатор  $\alpha$  хэш-функции, мы запускаем алгоритм генерации пары  $e, d$  из данной нам системы подписи. Затем находим сообщение  $x = C(e, \alpha)$  и подписываем его (поскольку мы сами сгенерировали оба ключа, мы можем подписать, что угодно), вычисляя  $s = S(d, h_\alpha(x))$ , а затем находим  $x' = E(e, h_\alpha(x), s)$ . Вероятность успеха этой атаки равна вероятности того, что  $x' \neq x$ , но  $h_\alpha(x') = h_\alpha(x)$ , то есть вероятности события  $A_n$ . По условию она не меньше  $\varepsilon/2$  (для бесконечно многих  $n$ ). Здесь важно, что распределение на парах  $(e, d, \alpha)$ , которое мы используем, такое же, как и в определении события  $A_n$ .

Исходная система подписи  $(\langle e, d \rangle, S, V)$  атакуется следующим образом (во втором случае). Получив открытый ключ  $e$ , мы запускаем алгоритм генерации идентификатора хэш-функции  $\alpha$ , данный нам в условии. Затем находим сообщение  $x = C(e, \alpha)$ . Подписать его самостоятельно мы теперь не можем, поскольку у нас нет закрытого ключа. Поэтому мы просим обладателя закрытого ключа подписать  $h_\alpha(x)$  (мы имеем право попросить один раз подписать сообщение длины  $k(n)$ ). Получив  $s = S(d, h_\alpha(x))$ , мы вычисляем  $x' = E(e, \alpha, s)$  и  $s' = D(e, \alpha, s)$ . В качестве результата мы выдаем сообщение  $h_\alpha(x')$  и фальшивую подпись  $s'$  под ним. Вероятность успеха этой атаки равна вероятности того, что  $h_\alpha(x') \neq h_\alpha(x)$ , но  $V(e, h_\alpha(x'), s') = 1$ , то есть вероятности события  $B_n$ .

## 15.5. ПОСТРОЕНИЕ СХЕМЫ ПОДПИСИ ОДНОГО СООБЩЕНИЯ ПРОИЗВОЛЬНОЙ ДЛИНЫ

Здесь опять важно, что распределение на парах  $(e, d, \alpha)$ , которое мы используем, такое же, как и в определении события  $B_n$ .  $\square$

Эта система подписи обладает одним важным достоинством, не отраженным в определениях. Длина подписи зависит только от параметра безопасности (и не зависит от длины исходного сообщения).

### 15.5 Построение схемы подписи одного сообщения произвольной длины, исходя из универсальных семейств односторонних хэш-функций

Пусть имеется схема подписи  $\langle \langle e_n, d_n \rangle, S, V \rangle$  одного сообщения длины  $n + 1$ . Пусть длина открытого ключа  $e_n$  в этой схеме равна полиному  $p(n)$ . Пусть, кроме того, имеется универсальное семейство односторонних хэш-функций  $h^n : \{0, 1\}^{q(n)} \times \{0, 1\}^{p(n)} \rightarrow \{0, 1\}^n$ , где  $p(n)$  есть длина открытого ключа в схеме подписи, а  $q(n)$  — произвольный многочлен.

Сначала построим систему подписи в которой условие  $x' \neq x$  успеха атаки заменено на более сильное условие: ни одно из слов  $x'$  и  $x$  не является началом другого. Открытый ключ состоит из пары  $\langle e, s \rangle$ , а закрытый ключ — из пары  $\langle d, s \rangle$  (опускаем параметр безопасности). Подпись под последовательностью битов  $x = \sigma_1 \dots \sigma_l$  устроена так. Подписывающей выбирает случайно и независимо новые пары ключей  $\langle e_2, d_2 \rangle, \dots, \langle e_{l+1}, d_{l+1} \rangle$ . Затем подписывает (алгоритмом  $S$ ) последовательность  $h_s(e_2)\sigma_1$  с ключом  $d$ , последовательность  $h_s(e_3)\sigma_2$  с ключом  $d_2$ , и так далее. Последовательность  $h_s(e_{i+1})\sigma_i$  подписывается с ключом  $d_i$ . Последней подписывается последовательность  $h_s(e_{l+1})\sigma_l$  с ключом  $d_l$  (ключ  $e_{l+1}$  использоваться не будет). Обозначим через  $s_1, \dots, s_l$  полученную последовательность подписей. Общей подписью будет последовательность открытых ключей  $e_2, \dots, e_{l+1}$  и последовательность подписей  $s_1, \dots, s_l$ . Алгоритм проверки подписи состоит в применении алгоритма  $V$  с ключами  $e, e_2, \dots, e_l$  к сообщениям (соответственно)  $h_s(e_2)\sigma_1, \dots, h_s(e_l)\sigma_{l-1}, h_s(e_{l+1})\sigma_l$  и подписям  $s_1, \dots, s_l$ .

**Теорема 36.** Построенные пара алгоритмов  $\tilde{S}, \tilde{V}$  и случайная величина  $\langle \tilde{e}, \tilde{d} \rangle$  являются протоколом электронной подписи одного сообщения

произвольной длины.

*Доказательство.* Пусть противник атакует построенную систему подписи, используя схемы  $\tilde{C}, \tilde{D}, \tilde{E}$ . То есть, сначала он вычисляет

$$x = \sigma_1 \dots \sigma_l = \tilde{C}(\alpha, e).$$

Затем получает подпись  $\langle s_1 s_2 \dots s_l, e_2 e_3 \dots e_{l+1} \rangle$  под  $x$ , где

$$s_1 = S(d, h_s(e_2)\sigma_1),$$

$$s_2 = S(d_2, h_s(e_3)\sigma_2),$$

...

$$s_l = S(d_l, h_s(e_{l+1})\sigma_l).$$

После этого он сам вычисляет

$$x' = \sigma'_1 \dots \sigma'_{l'} = \tilde{E}(s, e, s_1 s_2 \dots s_l)$$

и фальшивую подпись

$$s' = \langle s'_1 s'_2 \dots s'_{l'}, e'_2 e'_3 \dots e'_{l'+1} \rangle = \tilde{D}(s, e, s_1 s_2 \dots s_l).$$

Атака успешна, если  $x'$  не согласовано с  $x$  и

$$V(e, h_s(e'_2)\sigma'_1, s'_1) = 1,$$

$$V(e_2, h_s(e'_3)\sigma'_2, s'_2) = 1,$$

...

$$V(e_{l'}, h_s(e'_{l'+1})\sigma'_{l'}, s'_{l'}) = 1.$$

Допустим для бесконечно многих почти всех  $n$  вероятность этого события больше  $\varepsilon = 1/\text{poly}(n)$ .

**Лемма.** *Если атака успешна, то найдется такое  $1 < j \leq \min\{l, l'\}$ , что  $e_j \neq e'_j$ , но  $h_s(e_j) = h_s(e'_j)$ , или найдется такое  $1 \leq j \leq \min\{l, l'\}$ , что  $e_j = e'_j$ , но  $\sigma_j h_s(e_{j+1}) \neq \sigma'_j h_s(e'_{j+1})$ .*

В формулировке этой леммы мы предполагаем, что  $e_1 = e'_1 = e$ . Эта лемма утверждает, что у противника есть только две возможности подделать подпись в новой системе: первая заключается в том, чтобы отыскать  $e'_j$  отличное от  $e_j$  с тем же хэш-значением, что и у  $e_j$ . Вторая возможность — подделать подпись в старой системе под сообщением  $\sigma'_j h_s(e'_{j+1})$ , отличным от сообщения  $\sigma_j h_s(e_{j+1})$ , подписанного владельцем секретного ключа  $d_j$ .

## 15.5. ПОСТРОЕНИЕ СХЕМЫ ПОДПИСИ ОДНОГО СООБЩЕНИЯ ПРОИЗВОЛЬНОЙ ДЛИНЫ

*Доказательство.* Допустим, что утверждение леммы неверно. В частности, вторая возможность не реализуется для  $j = 1$ . Поскольку  $e_1 = e'_1 = e$ , мы можем заключить, что  $\sigma_1 h_s(e_2) = \sigma'_1 h_s(e'_2)$ , то есть  $\sigma_1$  и  $\sigma'_1$  совпадают и  $h_s(e_2) = h_s(e'_2)$ . Пользуясь тем, что первая возможность не реализуется для  $j = 2$ , выводим, что  $e_2 = e'_2$ . Рассуждая по индукции, мы установим, что первые  $m = \min\{l, l'\}$  битов  $x$  и  $x'$  совпадают и  $e_{m+1} = e'_{m+1}$ . Это означает, что одно из слов  $x$  и  $x'$  является началом другого, то есть, атака не является успешной.  $\square$

По лемме с вероятностью не меньше  $\varepsilon/2$  то найдется такое  $1 < j \leq \min\{l, l'\}$ , что  $e_j \neq e'_j$ , но  $h_s(e_j) = h_s(e'_j)$ , или найдется такое  $1 \leq j \leq \min\{l, l'\}$ , что  $e_j = e'_j$ , но  $\sigma_j h_s(e_{j+1}) \neq \sigma'_j h_s(e'_{j+1})$  и  $V(e'_j, \sigma'_j h_s(e'_{j+1}), s'_j) = 1$ .

В первом случае мы можем успешно атаковать семейство хэш-функций. Поскольку  $j$  ограничено некоторым полиномом  $r(n)$  для некоторого фиксированного  $j \leq r(n)$  с вероятностью не менее  $\varepsilon/2r(n)$  будет выполнено  $j \leq \min\{l, l'\}$ ,  $e_j \neq e'_j$ , но  $h_s(e_j) = h_s(e'_j)$ . Найдем первый элемент коллизии, сгенерировав взяв случайный ключ исходной схемы; обозначим этот ключ через  $e_j$ . Получив случайный номер хэш-функции  $s$ , сгенерируем случайную пару ключей  $e_1, d_1$  и дадим ее противнику, использующему схемы  $\tilde{C}, \tilde{D}, \tilde{E}$ . Он попросит нас подписать некоторое сообщение  $x$ . Мы подписываем его, генерируя нужное количество пар ключей. Когда же он выдаст фальшивую подпись  $s'$ , возьмем в ней  $j$ -ый ключ  $e'_j$ . Этот ключ и будет с вероятностью не меньше  $\varepsilon/2r(n)$  вторым элементом коллизии  $h_s$ . Описанная атака на семейство хэш-функций вычисляется вероятностными схемами полиномиального размера. Зафиксировав в этих схемах подходящим образом используемые случайные биты, получим детерминированные атакующие схемы полиномиального размера.

Пусть теперь с вероятностью не меньше  $\varepsilon/2$  найдется такое  $j \leq \min\{l, l'\}$ , что  $e_j = e'_j$ , но  $\sigma_j h_s(e_{j+1}) \neq \sigma'_j h_s(e'_{j+1})$ . Опять, это событие имеет место с вероятностью не менее  $\varepsilon/2r(n)$  для некоторого фиксированного  $j \leq r(n)$ . Будем атаковать систему подписи одного сообщения длины  $n + 1$ . Пусть нам дан открытый ключ, обозначим его через  $e_j$  (он будет использован как  $j$ -ый ключ в подписи). Выберем случайно пару ключей  $e, d$  и хэш-функцию  $h_s$ . Затем дадим ключ  $\tilde{e} = \langle e, s \rangle$  на вход схеме  $\tilde{C}$ . Подпишем выданное схемой сообщение  $\sigma_1 \dots \sigma_l$ , генерируя новые пары ключей  $e_2, d_2, \dots, e_l, d_l$  (кроме  $j$ -ой пары). А подпись под  $\sigma_j h_s(e_{j+1})$  получим у Подписывающего одно сообщение длины  $n + 1$  закрытым ключом  $d_j$ . Все подписи даем на вход схемам  $\tilde{D}, \tilde{E}$ . Схемы выдадут новое сооб-

щение  $\sigma'_1 \dots \sigma'_{l'}$  и фальшивую подпись  $\langle s'_1 s'_2 \dots s'_{l'}, e'_2 e'_3 \dots e'_{l'+1} \rangle$ , про которые известно, что с вероятностью не менее  $\varepsilon/2r(n)$  выполнено  $e_j = e'_j$ ,  $\sigma_j h_s(e_{j+1}) \neq \sigma'_j h_s(e'_{j+1})$  и  $V(e'_j, \sigma'_j h_s(e'_{j+1}), s'_j) = 1$ . То есть,  $\sigma'_j h_s(e'_{j+1})$  является сообщением, отличным от того, которое мы просили подписать, и подпись  $s'_j$  под которым принимается с ключом  $e_j = e'_j$ .

Осталось построить схему в которой условие  $x' \neq x$  успеха атаки не заменено на более сильное условие. Определим для каждого двоичного слова  $y$  его префиксный код  $\hat{y}$  следующим образом. Удвоим все биты  $y$  и припишем 01 в конце, например,  $\widehat{011} = 00111101$ . По  $y$  легко найти  $\hat{y}$ , и если одно из слов вида  $\hat{y}$  является началом другого слова такого же вида, то эти слова совпадают. Модифицируем построенную систему подписи следующим образом. Будем вместо подписывания слова  $y$  подписывать его префиксный код  $\hat{y}$  (и при проверке будем проверять, подписано ли  $\hat{y}$ ). Если два слова вида  $\hat{y}$  различны, то ни одно из них не является началом другого. Поэтому модифицированная система подписи удовлетворяет требованию (б) в его исходном виде.  $\square$

## 15.6 Общий случай: подпись произвольного количества сообщений произвольной длины

### Конструкция

Схему подписи произвольного количества сообщений произвольной длины можно изготовить, исходя из протокола одноразовой подписи сообщений произвольной длины и семейства псевдослучайных функций. А именно, пусть  $\langle K, S, V \rangle$  — протокол одноразовой подписи сообщений любой длины (его мы далее будем называть “старым”, чтобы отличить от “нового”, который мы построим). Пусть  $l(n)$  — полином, ограничивающий количество случайных битов, нужных алгоритму генерации ключей  $K$ . Пусть  $f_s^n : \{0, 1\}^* \rightarrow \{0, 1\}^{l(n)}$  — семейство ПСФ.

В новом протоколе  $\langle \tilde{K}, \tilde{S}, \tilde{V} \rangle$  алгоритм алгоритм  $\tilde{K}$  запускает алгоритм  $K$  (с тем же параметром безопасности), который выдает некоторую пару  $\langle e, d \rangle$ , и алгоритм генерации идентификатора псевдослучайной функции (с тем же параметром безопасности), который выдает идентификатор  $s$  псевдослучайной функции  $f_s^n$ . В качестве закрытого ключа

## 15.6. ОБЩИЙ СЛУЧАЙ: ПОДПИСЬ ПРОИЗВОЛЬНОГО КОЛИЧЕСТВА СООБЩЕНИЙ ПРО

алгоритм  $\tilde{K}$  выдает пару  $\langle d, s \rangle$ , а в качестве открытого ключа —  $e$ . Подчеркнем, что  $s$  не включается в состав открытого ключа.

Новый алгоритм подписи основан на идее “размножения ключей”. Применим алгоритм  $K$  для генерации двух новых открытых ключей  $e_0, e_1$  вместе с парными к ним закрытыми ключами  $d_0, d_1$ . Чтобы проверяющий мог им доверять, подпишем конкатенацию  $e_0e_1$  с помощью исходного закрытого ключа и сообщим подпись и сами ключи адресату. Используя новые закрытые ключи  $d_0, d_1$ , можно подписать уже четыре новых открытых ключа,  $e_{00}, e_{01}, e_{10}, e_{11}$ , и так далее. Таким образом, можно считать что в нашем распоряжении находятся  $2^n$  новых ключей  $e_\alpha$  — для всех бинарных слов  $\alpha$  длины  $n$  — любой из них использовать для подписи данного сообщения. При этом в состав подписи нужно включить саму последовательность  $\alpha$  и ключи  $e_\beta$  для всех непустых начал слова  $\alpha$ , а также подписи под ними. Строку  $\alpha$ , которую мы дальше будем называть идентификатором сообщения, алгоритм подписи может выбирать случайно — тогда вероятность того, что при атаке идентификаторы двух разных сообщений совпадут, будет ничтожна. Поэтому этим событием можно пренебречь, а значит каждый из ключей  $d_\alpha$  (для всех строк  $\alpha$  длины  $n$ ) будет использован не более одного раза. (А можно идентификатор выбирать детерминированным образом, исходя из сообщения, — тот способ изложен в задаче, приведённой после доказательства.)

Однако нам нужно еще, чтобы и все остальные ключи  $d_\beta$ , а также исходный ключ  $d$ , использовались только один раз. Описанный выше алгоритм подписи этого не гарантирует. Действительно, допустим атакующий попросил нас подписать два сообщения. Тогда в первый раз мы подпишем с помощью исходного ключа  $d$  одну конкатенацию ключей  $e_0e_1$ , а во второй раз — другую. Исправить этот недостаток можно использованием псевдослучайной функции  $f_s$ . А именно, запустим алгоритм  $K$  с параметром безопасности  $n$ , а в качестве случайных битов дадим ему биты слова  $f_s(\alpha)$ , где  $\alpha$  — любая непустая строка длины не более  $n$ . Полученную пару ключей и будем использовать в качестве вышеупомянутых ключей  $(d_\alpha, e_\alpha)$ . Обозначение  $(e_\Lambda, d_\Lambda)$ , где  $\Lambda$  обозначает пустое слово, будем использовать для исходной пары ключей. Если длина  $\alpha$  меньше  $n$ , то ключ  $d_\alpha$  мы будем использовать только для подписи конкатенации ключей  $e_{\alpha 0}e_{\alpha 1}$ , а при  $|\alpha| = n$  — только для подписи самого сообщения. Напомним, что мы подписываем сразу конкатенацию ключей, поскольку иначе пришлось бы использовать ключ  $d_\alpha$  дважды — для подписи  $e_{\alpha 0}$  и для подписи  $e_{\alpha 1}$ .

Итак, сообщение  $x$  подписываем так. Выбираем случайным образом идентификатор  $\alpha$  (бинарную строку длины  $n$ ). Затем алгоритмом  $S$  с ключом  $d_\alpha$  подписываем  $x$  и включаем полученную строку в состав подписи под  $x$ . Кроме того, в состав подписи включаем  $e_{\gamma_0}e_{\gamma_1}$  и  $S(d_\gamma, e_{\gamma_0}e_{\gamma_1})$  для всех собственных начал (включая пустое) строки  $\alpha$  и сам идентификатор  $\alpha$ .

Проверяется новая подпись так: пусть нам дана последовательность, состоящая из идентификатора  $\alpha$ ,  $n + 1$ -ой подписи  $s_0, \dots, s_n$  для старой схемы и  $n$  пар ключей, которые мы будем обозначать через  $e'_{\gamma_0}e'_{\gamma_1}$ , где  $\gamma$  пробегает все собственные начала  $\alpha$ . Мы используем штрихи, поскольку не предполагаем заранее, что подпись честная. Таким образом, для любого непустого начала  $\beta$  последовательности  $\alpha$  у нас имеется ключ  $e'_\beta$ . Также положим  $e'_\Lambda$  равным данному нам открытому ключу. Затем алгоритмом  $V$  с ключом  $e'_\alpha$  проверяем, что  $s_n$  является подлинной подписью под данным нам сообщением, а так же, что для всех собственных начал  $\gamma$  строки  $\alpha$  слово  $s_{|\gamma|}$  является подлинной подписью под  $e'_{\gamma_0}e'_{\gamma_1}$  с ключом  $e'_\gamma$ . Если все проверки закончились успешно, то подпись признается, а иначе отвергается.

**Теорема 37.** *Построенная схема  $\langle \tilde{K}, \tilde{S}, \tilde{V} \rangle$  является схемой подписи произвольного числа сообщений.*

*Доказательство.* Условие (а) очевидно выполнено. Проверим условие (б). Пусть имеется схема  $C_n$ , которая для бесконечно многих  $n$  атакует новую схему с вероятностью успеха  $\varepsilon = 1/\text{poly}(n)$ . Заменим в новой схеме ПСФ  $f_s$  на случайную функцию  $g : \{0, 1\}^* \rightarrow \{0, 1\}^{l(n)}$ . Подписывающий алгоритм  $\tilde{S}$  вместе с атакующей схемой  $C$  и проверяющим алгоритмом  $\tilde{V}$  можно рассматривать как вероятностную схему для проверки случайности функции  $f_s$ : имея функцию  $f_s$  в качестве оракула, эта схема генерирует ключи  $e, d$  и моделирует атаку схемы  $C$  на новую систему подписи с ключами  $(d, s), e$  (заметим, что само  $s$  при моделировании атаки знать не нужно). После окончания атаки схема выдает 1, если атака прошла успешно. В силу надежности семейства ПСФ, в результате замены  $f_s$  на случайно выбранную функцию  $g$  вероятность успеха атаки изменится незначительно, поэтому для бесконечно многих  $n$  она не меньше  $\varepsilon/2$ . Таким образом, мы можем считать далее, что в состав закрытого ключа  $\tilde{e}$  вместо идентификатора псевдослучайной функции  $s$  входит случайно выбранная функция  $g$ . Другими словами, мы можем



## 15.6. ОБЩИЙ СЛУЧАЙ: ПОДПИСЬ ПРОИЗВОЛЬНОГО КОЛИЧЕСТВА СООБЩЕНИЙ ПРО

считать, что все ключи  $(e_\beta, d_\beta)$  независимо друг от друга порождаются старым алгоритмом  $K$ .

Нам нужно научиться с помощью схемы  $C$  научиться атаковать исходный протокол одноразовой подписи. Объясним сначала неформально, каким образом мы будем использовать схему  $C$  для атаки исходного протокола.

Смоделируем мысленно атаку схемы  $C$  на новый протокол подписи, выбирая самостоятельно случайным образом исходную пару ключей и случайную функцию  $g$ . А именно, сначала запустим алгоритм  $K$  и найдем исходную пару ключей, которую обозначим  $(e_\Delta, d_\Delta)$ . Затем передаем  $e_\Delta$  схеме  $C$ . После этого схема дает нам на подпись сообщение и мы подписываем его, используя для генерации ключей случайную функцию  $g$ . Так же поступаем со вторыми и последующими сообщениями.

Как мы уже отмечали, использование случайной функции  $g$  означает, что каждая новая пара ключей  $(e_\alpha, d_\alpha)$ , которая нам понадобилась, генерируется независимо от предыдущих. Но при этом, если при подписывании некоторого сообщения мы сгенерировали один раз пару ключей  $(e_\alpha, d_\alpha)$ , а затем позже нам понадобилась еще раз пара ключей  $(e_\alpha, d_\alpha)$ , то мы не генерируем ее заново, а используем вторично ту же самую (для этого мы запоминаем все сгенерированные пары ключей).

В какой-то момент атакующая схема сообщит фальшивое сообщение  $x$  и фальшивую подпись  $\tilde{s}$  под ним. После этого моделирование останавливается. Будем считать моделирование успешным, если пара (фальшивое сообщение, фальшивая подпись) прошла проверку алгоритмом  $\tilde{V}$ . Как мы уже говорили, моделирование заканчивается успешно с вероятностью не меньше  $\varepsilon_n/2$  (для бесконечно многих  $n$ ).

Будем называть строку  $\alpha$  *использованной* (в моделировании), если мы в ходе него сгенерировали пару ключей  $(e_\alpha, d_\alpha)$ .

Допустим, что моделирование закончилось успехом. Добавим фальшивое сообщение  $x$  в конец фальшивой подписи  $\tilde{s}$ . Изменённая таким образом  $\tilde{s}$  состоит некоторого идентификатора  $\beta$  (двоичной строки длины  $n$ ) и  $n + 1$  пары (подпись, сообщение) для исходной системы подписи. В последней,  $n + 1$ -ая пара сообщение состоит из  $x$ , а в первых  $n$  парах сообщение состоит из пары ключей. Будем эти ключи, аналогично настоящим ключам, использованным нами, обозначать через  $e'_\alpha$ , где  $\alpha$  обозначает соответствующее начало  $\beta$  или его брата. Положим также  $e'_\Delta$  равным ключу  $e_\Delta$ . Раз атака успешна, все пары проходят проверку алгоритмом  $S$  с подходящими ключами, то есть,  $j + 1$ -ая пара фальши-

вой подписи признана подлинной алгоритмом  $V$  с ключом  $e'_{\beta[1\dots j]}$ . Тогда идентификатор  $\beta$ , с которого начинается фальшивое сообщение, имеет начало  $\alpha = \beta[1\dots j]$  такое, что

- $\alpha$  является использованным,
- $e'_\alpha = e_\alpha$ ,
- с помощью ключа  $d_\alpha$  мы ничего не подписывали или подписанное с его помощью сообщение отличается от сообщения в  $j + 1$ -ой паре фальшивой подписи (напомним, что первый компонент этой пары прошел проверку как подпись под этим сообщением).

Действительно, если мы вообще ничего не подписывали в ходе атаки, то можно положить  $\alpha$  равным пустому слову. Иначе, мы с ключом  $d_\Delta$  подписали слово  $e_0e_1$ . Если слово  $e'_0e'_1$  не совпадает со словом  $e_0e_1$ , то опять же можно положить  $\alpha$  равным пустому слову. Иначе положим  $\alpha$  равным первому биту  $\beta$ . Заметим, что  $e'_\alpha = e_\alpha$ , поскольку  $e'_0e'_1 = e_0e_1$ . Дальше продолжаем рассуждать по индукции. На очередном шаге у нас имеется некоторое начало  $\alpha$  фальшивого идентификатора  $\beta$ , которое использовано и для которого  $e_\alpha = e'_\alpha$ . Если ни один из использованных нами идентификаторов не начинается на  $\alpha$ , то мы с ключом  $d_\alpha$  ничего не подписывали и тем самым  $\alpha$  годится. Иначе  $\alpha$  является собственным началом как слова  $\beta$ , так и использованного нами идентификатора. Поэтому мы с ключом  $d_\alpha$  подписали слово  $e_{\alpha 0}e_{\alpha 1}$ , а схема — слово  $e'_{\alpha 0}e'_{\alpha 1}$ . Если эти слова различны, то опять же текущее  $\alpha$  годится. Иначе приписываем к  $\alpha$  еще один бит  $\beta$ , сохраняя инвариант. Рано или поздно мы найдем нужное начало, поскольку в последней паре из фальшивой подписи сообщение отличается от всех подписанных нами сообщений.

Будем такие  $\alpha$ , ключ  $e_\alpha$  и  $j$ -ую пару слов из фальшивой подписи называть *особыми*. Если особых пар в фальшивой подписи несколько, то будем считать особой, скажем, только первую из них. Особая пара успешно прошла проверку алгоритмом  $V$  со случайно выбранным ключом  $e_\alpha$  и второй компонент в ней не был подписан нами в ходе моделирования атаки. Это означает, что схема  $S$  умеет подписывать выбранным нами случайным ключом не подписанные нами сообщения. Нам надо научиться использовать это ее свойство.

Теперь мы, наконец, объясним, как с помощью этого моделирования можно атаковать исходный протокол подписи. Оценим сверху общее количество пар (открытый ключ, закрытый ключ), которое может быть

## 15.6. ОБЩИЙ СЛУЧАЙ: ПОДПИСЬ ПРОИЗВОЛЬНОГО КОЛИЧЕСТВА СООБЩЕНИЙ ПРО

использовано в ходе моделирования атаки  $C$ . Ясно, что их количество не превосходит некоторого полинома  $p(n)$ . (Нам будет достаточно столько пар ключей, какова суммарная длина всех сообщений, которые нам придётся подписать. А последняя ограничена размером схемы  $C_n$ .)

Пусть нам дан открытый ключ для атаки, сгенерированной алгоритмом  $K$ . Выбираем число  $i$  от 1 до  $p(n)$ . Затем запускаем моделирование атаки схемы  $C$ , описанное выше со следующим изменением. Когда понадобится в  $i$ -ый раз запускать алгоритм  $K$  для порождения очередной пары ключей, мы этого не делаем. Вместо  $i$ -ого открытого ключа мы используем данный нам открытый ключ, а когда (и если) потребуется использовать использовать парный к нему закрытый ключ, мы просто попросим владельца этого закрытого ключа подписать требуемое сообщение. Это произойдет не более одного раза, если пренебречь тем, что какие-то два идентификатора совпали — вероятность этого события в самом деле пренебрежимо мала. Если атака схемы  $C$  оказалась успешной, причем нам повезло и особый ключ равен данному нам для атаки ключу, то выдаем особую пару на выход.

Докажем, что вероятность успеха этой атаки не меньше  $\varepsilon/2p(n)$ . Сначала заметим, что сделанная нами подмена в  $i$ -ой паре, никак не повлияла на вероятность успеха моделируемой атаки. Действительно, с точки зрения схемы  $C$  мы действуем ровно таким же образом, как в исходном моделировании атаки. Замена  $i$ -ой пары ключей на данный нам ключ никак не влияет на распределение открытых ключей, поскольку данный нам ключ выбирается случайно и независимо от остальных сгенерированных ключей. Кроме того, последовательность пар ключей после подмены и само число  $i$  независимы (в смысле теории вероятностей). В самом деле, при любом фиксированном  $i$  последовательность, полученная в результате подмены, так же распределена, как и до подмены. Наконец, все подписи, сообщенные схеме  $C$ , были получены применением алгоритма  $S$  с подлинными ключами. Итак, с вероятностью не менее  $\varepsilon/2$  фальшивая подпись будет признана подлинной и содержит особую пару. Кроме того, в силу независимости  $i$  и последовательности пар ключей, при любой фиксированной последовательности пар ключей, для которой существует особое  $\alpha$ , с вероятностью  $1/p(n)$  число  $i$  окажется равным номеру особого ключа  $e_\alpha$  в порядке генерации пар ключей. Поэтому с вероятностью  $\varepsilon/2p(n)$  нам повезет и особый ключ окажется равным данному нам для атаки ключу.  $\square$

*Задача 60.* Докажите, что для теоремы 37 достаточно надежности семейства ПСФ относительно неадаптивной атаки.

*Задача 61.* Будем в многоцветной схеме подписи в качестве идентификатора сообщения использовать его префиксный код. Докажите, что теорема 37 останется верной. Докажите, что если в качестве идентификатора использовать сообщения использовать само сообщение, то теорема 37 перестанет быть верной.

*Задача 62.* В предыдущей задаче длина подписи зависит от длины сообщения. Модифицируйте систему подписи из предыдущей задачи так, чтобы длина подписи зависела только от параметра безопасности (предполагая, что это верно для исходной одноцветной схемы).

## Глава 16

# Конфиденциальные вычисления

Пусть у Алисы имеется строка  $x$ , а у Боба строка  $y$ . Пусть еще имеется полиномиально вычислимая функция  $f(x, y)$ . Нужно придумать протокол общения, выполнив который Алиса и Боб узнают  $f(x, y)$  и ничего больше. В частности, Боб не узнает  $x$ , а Алиса не узнает  $y$ . Например, они хотят узнать, кто из них богаче, не разглашая своего состояния.

В более общей формулировке имеются две функции  $f(x, y)$  и  $g(x, y)$ . После выполнения протокола Алиса должна узнать только  $f(x, y)$ , а Боб только  $g(x, y)$ .

Будем называть протоколом конфиденциального вычисления пары  $f, g$  пару полиномиальных вероятностных алгоритмов  $(A, B)$ , получающих на вход параметр безопасности в единичной кодировке и слова  $x, y$  (алгоритм  $A$  получает на вход  $x$ , а  $B$  —  $y$ ) удовлетворяющую следующим свойствам.

(а) (Полнота) Для любых последовательностей слов  $x_n, y_n$  полиномиальной длины с приблизительно единичной вероятностью после общения  $A(x)$  и  $B(y)$  алгоритм  $A$  напечатает  $f(x, y)$ , а алгоритм  $B$  напечатает  $g(x, y)$ .

Вторым условием будет неразглашение информации. Его можно сформулировать в двух вариантах: первый вариант предполагает, что во время общения Алиса и Боб не отклоняются от протокола (так называемые “полу-честные”, semi-honest, игроки), а во втором варианте они могут отклоняться от протокола (“нечестные”, malicious, игроки).

В первом варианте условие неразглашения выглядит так.

(б) (Неразглашение информации для полу-честных игроков). Существует полиномиальный вероятностный алгоритм  $S$ , который получает

на вход параметр безопасности и две строки и такой, что для любой последовательности строк  $x_n, y_n$  полиномиальной длины случайная величина  $S(x_n, f(x_n, y_n))$  вычислительно неотличима от пары  $\langle r, c(A(x_n, r), B(y_n)) \rangle$ , состоящей из случайных битов  $r$  алгоритма  $A$  и диалога между  $A$  с входом  $x_n$  и  $B$  с входом  $y_n$ . Также должно быть выполнено симметричное условие: Существует полиномиальный вероятностный алгоритм  $T$ , который получает на вход параметр безопасности и две строки и такой, что для любой последовательности строк  $x_n, y_n$  полиномиальной длины случайная величина  $T(y_n, g(x_n, y_n))$  вычислительно неотличима от  $\langle r, c(A(x_n), B(y_n, r)) \rangle$ .

Иными словами, если Алиса и Боб во время общения ведут себя честно, но после общения становятся не в меру любопытными, они не смогут извлечь никакой дополнительной информации, вспоминая проведенный диалог.

Во втором варианте неразглашение информации формулируется так.

(б) (Неразглашение информации для нечестных игроков). Существует полиномиальный вероятностный алгоритм  $S$ , который получает на вход параметр безопасности, строку и схему и такой, что для любой последовательности строк  $x_n$  полиномиальной длины и любой последовательности схем  $B_n^*$  полиномиального размера алгоритм  $S$  на входе  $x_n$  находит некоторую строку  $y_n$ , и после того, как ему сообщат  $f(x_n, y_n)$ , выдает на выход случайную величину, вычислительно неотличимую от диалога между  $A$  с входом  $x_n$  и  $B_n^*$ ,  $c(A(x_n), B_n^*)$ . Также должно быть выполнено симметричное условие.

## 16.1 Забывающая передача (oblivious transfer)

Так называются протоколы конфиденциального вычисления пары  $f, g$ , где  $f(x, i) = \Lambda$ ,  $g(x, i) = x[i]$  ( $i$ -ый бит  $x$ ). Если длина  $x$  предполагается фиксированной и равной  $k$ , то такие протоколы называются  $OT_1^k$ .

Сначала предположим, что требование неразглашения информации должно быть выполнено только для полу-честных игроков. Протокол забывающей передачи можно изготовить из улучшенной проверяемой перестановки с секретом.

Пусть дана такая перестановка  $f_e(x)$ . Применяв теорему Левина-Голдрайха, мы без ограничения общности можем считать, что у нас есть еще и трудный бит  $h_e(x)$  для этой функции. Причём мы можем считать,

что этот бит трудно вычислить даже по случайным битам, использованным при генерации  $f_e(x)$  по  $e$ .

По лемме Яо (Лемма 3.4) пара, состоящая из  $h_e(f_e^{-1}(y))$  и случайных битов  $r$ , использованных при генерации  $y$  вычислительно неотличима от пары (случайный бит,  $r$ ).

### Протокол забывающей передачи $OT_1^k$

Пусть входом Алисы является строка  $x$  длины  $k$ , а входом Боба — число  $i$  от 1 до  $k$ . Алиса выбирает случайно пару  $e, d$  и посылает  $e$  Бобу. Боб, запустив  $k$  раз алгоритм  $S$  на входе  $(1^n, e)$  выбирает  $k$  случайных строк  $y_1, \dots, y_k$  из  $D^e$ , применяет к  $i$ -ой строке функцию  $f_e$  (а к остальным строкам не применяет) и посылает полученные  $k$  строк Алисе. Алиса, получив кортеж из  $k$  строк  $z_1, \dots, z_k$ , посылает обратно Бобу кортеж из  $k$  строк, у которого  $j$ -ая строка равна  $h_e(f_e^{-1}(z_j)) \oplus x[j]$ . Алиса может обратить  $f_e$ , поскольку ей известно  $d$ . Наконец, Боб получив кортеж из  $k$  битов  $\sigma_1, \dots, \sigma_k$ , выдает  $\sigma_i \oplus h_e(y_i)$ .

По построению,  $z_i = f_e(y_i)$ , поэтому  $\sigma_i = h_e(y_i) \oplus x[i]$ , а значит Боб в самом деле выдает  $x[i]$ . Поэтому этот протокол удовлетворяет условию полноты. Проверим условие неразглашения информации. Сразу отметим, что для этого протокола оно выполнено только для полуприемных игроков. Действительно, нечестный Боб мог бы применить функцию  $f_e$  не только к  $y_i$ , но и ко всем остальным строкам. Тогда бы он мог узнать аналогичным образом и все остальные биты  $x$  целиком, а не только  $i$ -ый.

Сначала проверим, что Боб не разглашает информации. Причем это верно даже если Алиса действует нечестно, используя некоторую схему  $A_n^*$  полиномиального размера. Действительно, последнее сообщение делается Алисой, поэтому достаточно проверить что первые два сообщения из диалога  $A^*$  и  $B(i)$  можно сгенерировать, не зная  $i$ . Первое сообщение есть  $e = A^*(\Lambda)$  а второе равно  $(y_1, \dots, f_e(y_i), \dots, y_k)$ . Поскольку  $f_e$  — перестановка и распределение  $y_i$  статистически неотлично от равномерного распределения на  $D_e$ , второе сообщение статистически неотлично от  $(y_1, \dots, y_i, \dots, y_k)$  (при любом фиксированном  $e$ ), а последнее можно сгенерировать, не зная  $i$ .

Теперь проверим, что Алиса разглашает только  $x[i]$ . Пусть Боб использовал случайные биты  $r_1, \dots, r_k$  для генерации  $y_1, \dots, y_k$ . Нам нужно научиться порождать случайную величину, вычислительно неотличима

мую от

$$r_1, \dots, r_k, e, (S(r_1), \dots, f_e(S(r_i)), \dots, S(r_k)), (x[1] \oplus h_e(f_e^{-1}S(r_1)), \dots, x[i] \oplus h_e(S(r_i)), \dots, x[k] \oplus h_e(S(r_k))) \quad (16.1)$$

зная только  $i$  и  $x[i]$ . Для этого запускаем алгоритм генерации ключей  $e, d$  и выбираем случайно  $r_1, \dots, r_k$ . После этого выдаем

$$r_1, \dots, r_k, e, (S(r_1), \dots, f_e(S(r_i)), \dots, S(r_k)), (\sigma_1, \dots, x[i] \oplus h_e(S(r_i)), \dots, \sigma_k), \quad (16.2)$$

где  $\sigma_1, \dots, \sigma_k$  — случайные биты. Почему эта случайная величина вычислительно неотличима от (16.1)? Ясно, что она имеет то же распределение, что и случайная величина

$$r_1, \dots, r_k, e, (S(r_1), \dots, f_e(S(r_i)), \dots, S(r_k)), (x[1] \oplus \sigma_1, \dots, x[i] \oplus h_e(S(r_i)), \dots, x[k] \oplus \sigma_k). \quad (16.3)$$

При  $k = 2$  мы можем воспользоваться леммой Яо. Пусть, скажем  $i = 1$ . Поскольку по  $r_2, e$  невозможно вычислить  $h_e(f_e^{-1}S(r_2))$ , случайные величины (16.3) и (16.1) вычислительно неотличимы. При  $k > 2$  надо воспользоваться гибридными случайными величинами.



# Глава 17

## Интерактивные алгоритмы

### 17.1 Определение

Интерактивные алгоритмы — это алгоритмы, которые, кроме обычного входа, имеют еще входной и выходной каналы обмена информацией. Чтобы дать определение интерактивному алгоритму, начнем с понятия стратегии. Стратегией называется функция, отображающая конечные последовательности слов в бинарном алфавите в слова в бинарном алфавите. Пусть имеются стратегии  $F, G$ . Последовательность сообщений  $c_1, c_2, \dots$  между стратегиями определяется по правилу

$$\begin{aligned}c_1 &= F(\Lambda), \\c_2 &= G(c_1), \\c_3 &= F(c_1, c_2), \\c_4 &= G(c_1, c_2, c_3), \\&\dots\end{aligned}$$

Общение длится до тех пор, пока некоторое  $c_i$  не станет пустым; посылка пустого слова означает желание прекратить общение. Если пустое слово не послано, то последовательность сообщений бесконечна. Последовательность сообщений  $c_1, c_2, c_3, \dots$  будем обозначать через  $c(F, G)$  и называть диалогом между  $F$  и  $G$ . Диалог между  $F$  и  $G$  это не одно и то же, что диалог между  $G$  и  $F$ : в последнем случае общение начинается стратегией  $G$ , а не  $F$ .

Стратегией с параметром из некоторого множества  $M$  называется отображение, сопоставляющее каждому элементу  $x \in M$  некоторую стра-

тегию  $F_x$ . Интерактивным алгоритмом со входами из  $\{0, 1\}^*$  называется вычислимая стратегия с параметром из  $\{0, 1\}^*$ . Чтобы запустить интерактивный алгоритм  $A$ , задаваемый стратегией с параметром  $F_x$ , нужен вход и собеседник, то есть, некоторая стратегия. Пусть  $x$  бинарное слово, а  $G$  — некоторая стратегия. Результат  $A^G(x)$  работы алгоритма  $A$  на входе  $x$  с собеседником  $G$  определяется следующим образом. Запустим стратегии  $F_x$  и  $G$ . Пусть  $s(F_x, G)$  полученный диалог. Если диалог бесконечен, то результат не определен. Иначе применим к  $s(F_x, G)$  стратегию  $F_x$ , результатом будет полученное слово. Пусть  $A, B$  два интерактивных алгоритма, а  $x, y$  двоичные слова. Запустим алгоритм  $A$  на входе  $x$ , а  $B$  на входе  $y$  так, чтобы они общались друг с другом. Результат, выданный алгоритмом  $A$  будем обозначать через  $A^{B(y)}(x)$ .

Обозначение  $A^G(x)$  напоминает обозначения, связанные с машинами с оракулом. Подчеркнем важное отличие машин с оракулом от интерактивных алгоритмов. Ответ оракула на очередной вопрос зависит только от этого вопроса. Очередное сообщение стратегии зависит от всех предыдущих сообщений полученных от алгоритма.

Теперь определим понятие полиномиального интерактивного алгоритма. Полиномиальной стратегией  $F_x$  с параметром из некоторого множества двоичных слов называется стратегия, вычислимая за полиномиальное от  $n = |x|$  время. Такая стратегия может сделать только полиномиальное от  $n$  количество ходов, полиномиальной от  $n$  длины (точнее, сделанные ей ходы могут зависеть только от полинома первых символов последовательности сделанных ходов). Интерактивный алгоритм  $A$  называется полиномиальным, если стратегия с параметром, его задающая, полиномиальна.

Полиномиальные вероятностные интерактивные алгоритмы (и стратегии с параметром) можно определить через полиномиальные детерминированные интерактивные алгоритмы. Пусть  $A$  полиномиальный детерминированный интерактивный алгоритм, а  $p$  некоторый полином. Будем понимать результат работы  $A$  на входе вида  $xr$  с собеседником  $G$ , где  $r$  двоичное слово длины  $p(|x|)$ , выбираемое случайно по равномерному распределению, как результат работы вероятностного интерактивного алгоритма на входе вида  $x$  с собеседником  $G$ .

Последовательность стратегий  $\{F_n\}$  называется неравномерно полиномиальной, если существуют последовательность схем  $C_n$  такая, что размер схемы  $C_n$  ограничен полиномом от  $n$ , и функция  $F_n$  вычисляется схемой  $C_n$ . Последнее надо понимать следующим образом: слово  $y$  не

фиксированной длины подается на вход схемы в кодировке  $y10^i$ , где  $i$  подобрано так, что длина слова  $y10^i$  равна количеству входных проводников схемы.

## 17.2 Повторение интерактивного протокола

Пусть  $V$  — интерактивный алгоритм с выходом  $0,1$ . Пусть интерактивная стратегия  $P$ , которая общается с  $V$ , стремится максимизировать вероятность результата  $1$ . Обозначим через  $p(x)$  максимально возможную вероятности результата  $1$ , то есть,  $p(x) = \max_P \Pr[V^P(x) = 1]$ .

Пусть дано произвольное натуральное число  $k$ . Рассмотрим новый интерактивный алгоритм  $\tilde{V}$ , состоящий в следующем. Будем повторять  $V(x)$  последовательно  $k$  раз, используя при каждом повторении новые случайные биты. При этом стратегия  $P$ , с которой беседует  $\tilde{V}$ , не обязана действовать одинаково при каждом повторении: ее действия при очередном повторении могут зависеть от сообщений, полученных в ходе предыдущих повторений. В результате беседы у алгоритма  $\tilde{V}(x)$  появится  $k$  результатов  $c_1, \dots, c_k$ . В качестве окончательного результата он выдает  $1$ , если  $c_1 = \dots = c_k = 1$ , и выдает ноль иначе.

Пусть стратегия  $P$  стремится максимизировать вероятность того, что  $\tilde{V}(x)$  выдает  $1$ . Утверждается, что при любом  $k$  оптимальная стратегия  $P$  состоит в независимом применении стратегии, оптимальной для  $k = 1$ , то есть,

$$\max_P \Pr[\tilde{V}^P(x) = 1] = p(x)^k.$$

Неравенство

$$\max_P \Pr[\tilde{V}^P(x) = 1] \geq p(x)^k$$

очевидно — пусть  $P$  применяет при каждом повторении оптимальную стратегию для  $k = 1$ . Нам нужно доказать обратное неравенство Любая стратегия  $P$  может быть разложена в композицию стратегий

$$P_1, P_2(c_1), \dots, P_k(c_1, \dots, c_{k-1}).$$

Здесь  $P_1$  есть исходная стратегия  $P$ , оборванная после первого после первого повторения,  $P_2(c_1)$  есть стратегия  $P$ , полученная из  $P$  фиксированием сообщений  $c_1$ , полученных в ходе первого повторения, и оборванная после второго повторения, и так далее. Тогда  $\Pr[\tilde{V}^P = 1]$  есть

произведение вероятностей  $\Pr[V^{P_1} = 1]$ ,  $\Pr[V^{P_2(c_1)} = 1|V^{P_1} = 1]$ ,  $\dots$ ,  $\Pr[V^{P_k(c_1, \dots, c_{k-1})} = 1|V^{P_1} = 1, \dots, V^{P_{k-1}(c_1, \dots, c_{k-2})} = 1]$ . Докажем, что каждый сомножитель здесь не превосходит  $p$ . Для этого фиксируем  $i$ . Нам надо установить, что  $\Pr[V^{P_i(c_1, \dots, c_{i-1})} = 1|V^{P_1} = 1, \dots, V^{P_{i-1}(c_1, \dots, c_{i-2})} = 1]$ . Истинность условия зависит только от  $c_1, \dots, c_{i-1}$ . Поэтому достаточно доказать, что при любых фиксированных  $c_1, \dots, c_{i-1}$  Вероятность  $V^{P_i(c_1, \dots, c_{i-1})} = 1$  не превосходит  $p$ . Это следует из условия, поскольку стратегия  $P_i(c_1, \dots, c_{i-1})$  не может достигнуть вероятности результата 1 большей, чем максимальная.

Если исходная стратегия  $P$  задается схемой размера  $s$ , то и все рассматриваемые стратегии  $P_i(c_1, \dots, c_{i-1})$  задаются схемами не большего размера, что доказывает неравенство

$$\max_{P: \text{size}(P) \leq s} \Pr[\tilde{V}^P = 1] \leq \left( \max_{P: \text{size}(P) \leq s} \Pr[V^P = 1] \right)^k.$$

Аналогичное утверждение на самом деле верно в более общей ситуации. Будем рассматривать “игральные автоматы” следующего вида. В каждый момент функционирование автомата полностью определяется парой  $\langle s, q \rangle$ , где  $s$  — элемент некоторого конечного множества  $S$ , и  $q$  вершина в некотором конечном дереве  $Q$ , с корнем  $q_0$ . Элемент  $s$  называется секретом, выбирается случайно в начале игры (по некоторому распределению), хранится в секрете (от игрока) и не меняется в ходе игры. Вершина дерева  $q$  называется позицией, известна игроку изменяется в ходе игры. Начальная позиция равна корню дерева. Ходы делаются по очереди, начиная с игрока (таким образом очередь хода в позиции определяется четностью расстояния этой позиции от корня). В свою очередь автомат выбирает один из сыновей текущей вершины дерева  $q$  по некоторому распределению, зависящему от  $q$  и секрета  $s$ . То есть, для каждой тройки  $s, q, q'$ , где  $q'$  сын вершины  $q$  неотрицательное число  $p(s, q, q')$  так, что  $\sum_{q'} p(s, q, q') = 1$ . В свою очередь игрок выбирает также одного из сыновей текущей вершины. Игра продолжается до тех пор, пока  $q$  не станет равным некоторому листу, после чего автомат определяет, выиграл игрок или проиграл, применяя к  $q$  и секрету некоторую вероятностную функцию со значениями 0,1. То есть, для каждой пары (секрет, лист) задана вероятность  $p(s, q)$  того, что игрок выиграл в соответствующей позиции.

Для каждого автомата  $A$  рассмотрим  $p_A$  — максимум по всем стратегиям игрока вероятности выиграть, пользуясь этой стратегией. Каж-

дая пара (интерактивный алгоритм  $V$ , его вход  $x$ ) очевидным образом определяет игральный автомат  $A$  такой, что  $p_A = p_V(x)$ . Секрет — это случайные биты автомата, а позиция состоит из слова  $x$  и последовательности сообщений, сделанных Доказывающим и Проверяющим.

Рассмотрим на автоматах операцию возведения в степень. Неформально, автомат  $A^k$  это  $k$  автоматов типа  $A$ , поставленных рядом и действующих независимо друг от друга. Игрок может в свою очередь сделать один ход в игре с любым из  $k$  автоматов. Исходом игры является  $k$  битов. Кроме того, после окончания всех  $k$  игр, к их результатам применяется некоторая булева функция  $f$ , чтобы определить, выиграл ли игрок или проиграл. Так определенный игровой автомат будет обозначаться  $A_f^k$ .

Теперь формально. Пусть  $A$  — игральный автомат,  $k$  — произвольное натуральное число, а  $f$  — булева функция от  $k$  аргументов. Сначала построим игральный автомат  $A^k$ . Он функционирует следующим образом. Секретом является последовательность  $\langle s_1, \dots, s_k \rangle$  из  $k$  секретов исходного автомата, выбираемых независимо в соответствии с распределением, входящим в определение  $A$ . Позиция в каждый момент является кортежем  $\langle a_1, \dots, a_k \rangle$ , где  $a_1, \dots, a_k$  — позиции исходного автомата, в которых либо во всех позициях ход игрока, либо во всех позициях, кроме одной, ход игрока. В позициях первого типа ходить должен автомат, а в позициях второго типа — игрок. Начальная позиция состоит из  $k$  начальных позиций  $A$  (и в ней очередь хода за игроком). Если в позиции  $\langle a_1, \dots, a_k \rangle$  должен ходить игрок, он может выбрать любое число  $i = 1, \dots, k$  и сделать разрешенный ход в игре и  $i$ -ым автоматом. То есть, новое состояние должно иметь вид  $\langle a_1, \dots, a'_i, \dots, a_k \rangle$ , где  $a'_i$  — сын  $a_i$  в дереве автомата  $A$ . Если в позиции  $\langle a_1, \dots, a_k \rangle$  должен ходить автомат, то он делает ход, в соответствии со своей стратегией, в той из позиций  $a_i$ , где очередь хода за ним.

Листьями в новом дереве являются являются те кортежи, в которых все компоненты являются листьями. После того, как игра закончена, возникает  $k$  заключительных позиций и, соответственно,  $k$  результатов  $c_1, \dots, c_k$ . Это будет результатом игры автомата  $A^k$ . Результат игры автомата  $A_f^k$  получается применением к  $c_1, \dots, c_k$  функции  $f$ , то есть, игрок объявляется выигравшим, если  $f(c_1, \dots, c_k) = 1$

Оказывается, что если функция  $f$  монотонна, то

$$p_{A^k} = f(p, \dots, p),$$

где  $f(p_1, \dots, p_k)$  обозначает мультилинейное расширение  $f$ . Так называется функция

$$f(p_1, \dots, p_k) = \sum_{x_1, \dots, x_k \in \{0,1\}} f(x_1, \dots, x_k) \prod_i p_i^{(x_i)},$$

где  $p^{(1)} = p, p^{(0)} = 1 - p$ . Значение  $f(p_1, \dots, p_k)$  равно вероятности того, что  $f(x_1, \dots, x_k) = 1$ , если каждое из  $x_i$ , независимо от других, принимает значение 1 с вероятностью  $p$ , и 0 с вероятностью  $1 - p$ . Нетрудно доказать, что  $f$  не убывает по каждому аргументу. Из определения непосредственно следует, что  $f$  линейна по каждой переменной.

Сначала докажем это для прозрачных автоматов. Автомат  $A$  называется прозрачным, если у него отсутствует секрет (что означает, что множество  $S$  одноэлементно). Прозрачные автоматы соответствуют играм с полной информацией, которых фиксирована вероятностная стратегия одного из игроков (самого автомата).

Ценой позиции  $\langle a_1, \dots, a_k \rangle$  назовем максимум по стратегиям игрока среднего выигрыша при применении этой стратегии (среднее берется по случайным выборам, сделанным  $A^k$ ). Для игр с полной информацией цену позиции можно вычислять рекурсивно: если в данной позиции очередь хода за игроком, то цена позиции есть максимум по всем возможным его ходам цены позиции, полученной в результате этого хода, а если в данной позиции очередь хода за автоматом, то цена позиции есть среднее по всем возможным его ходам цены позиции, полученной в результате этого хода.

Докажем индукцией по максимальному возможному числу оставшихся ходов, что цена позиции  $c(a_1, \dots, a_k)$  равна  $f(c(a_1), \dots, c(a_k))$ , где  $c(a)$  — цена позиции  $a$  в исходной игре (задаваемой  $V$ ).

Разберем два случая:

1) Очередь хода за автоматом. Тогда цена позиции  $\langle a_1, \dots, a_k \rangle$  есть среднее по всем  $a'_1, \dots, a'_k$  цены позиции  $\langle a'_1, \dots, a'_k \rangle$ . Здесь  $a'_i$  обозначает случайную позицию, выбранную автоматом  $A$  по распределению которое входит в его определение. По индуктивному предположению цена позиции  $\langle a'_1, \dots, a'_k \rangle$  равна  $f(c(a'_1), \dots, c(a'_k))$ . Итак,  $c(a_1, \dots, a_k)$  есть среднее значение  $f(c(a'_1), \dots, c(a'_k))$ . Поскольку функция  $f$  линейна по всем аргументам, среднее значение  $f(c(a'_1), \dots, c(a'_k))$  совпадает со значением  $f$  на средних значениях  $c(a'_1), \dots, c(a'_k)$ . Поскольку среднее значение  $c(a'_i)$  равно цене позиции  $a_i$ , цена позиции  $\langle a_1, \dots, a_k \rangle$  равна  $f(c(a_1), \dots, c(a_k))$ , что и требовалось доказать.

2) Очередь хода за автоматом. Тогда цена позиции  $\langle a_1, \dots, a_k \rangle$  есть максимум по всем  $i$  максимума по всем  $a'_i$  цены позиции  $\langle a_1, \dots, a'_i, \dots, a_k \rangle$ . По индуктивному предположению цена такой позиции есть  $f(c(a_1), \dots, c(a'_i), \dots, c(a_k))$ . При любом фиксированном  $i$  в силу монотонности  $f$  максимум будет достигнут при том  $a'_i$ , которое максимизирует  $c(a'_i)$ . То есть при любом фиксированном  $i$  максимум будет равен  $f(c(a_1), \dots, c(a_i), \dots, c(a_k))$ , что не зависит от  $i$ . Так что игроку неважно, какое  $i$  выбрать, в любом случае при оптимальном выборе ходов он в среднем выиграет  $f(c(a_1), \dots, c(a_k))$ .

Чтобы доказать это для непрозрачных автоматов, мы переделаем любой непрозрачный автомат в эквивалентный прозрачный. Эквивалентность двух автоматов определяется так. Эквивалентные автоматы  $A_1, A_2$  должны иметь все одинаковое, кроме множества секретов и распределения на нем, и способа выбора очередного хода автоматом (то есть вероятностных стратегий, применяемых ими). При этом для любой стратегии  $P$  игрока средний выигрыш в игре с  $A_1$  должен быть равен среднему выигрышу в игре с  $A_2$ .

Пусть дан непрозрачный автомат  $A$ . Новый автомат  $A'$  делает ход  $q \rightarrow q'$  с вероятностью, равной условной вероятности того, что автомат  $A$  сделал бы ход  $q \rightarrow q'$  при условии, что автомат  $A$  окажется в позиции  $q$ . Вероятность здесь берется по распределению на секретах и распределению, входящему в определение переходов автомата  $A$ . Аккуратно вероятность перехода хода  $q \rightarrow q'$  определяется так. Зафиксируем  $s$  и определим  $t(s, q)$  как вероятность того, что игрок попадет в позицию  $q$ , если будет стремиться к этому, при условии, что секрет равен  $s$ . То есть,  $t(s, q)$  есть вероятность попасть в позицию  $q$ , если игрок в любой позиции, являющейся предком  $q$  выбирает ход, ведущий в  $q$ . Формально, для вершины  $q_{2i}$  четной высоты  $2i$ ,

$$t(s, q_{2i}) = p(s, q_1, q_2) \cdots p(s, q_{2i-1}, q_{2i}),$$

где  $q_0, q_1, \dots, q_{2i}$  — путь в дереве из корня в вершину  $q_{2i}$ . Для вершин  $q_{2i+1}$  нечетной высоты,  $t(s, q_{2i+1}) = t(s, q_{2i})$ , где  $q_{2i}$  обозначает отца  $q_{2i+1}$ .

Через  $t(q)$  обозначим среднее значение вероятности попасть в вершину  $q$ , то есть  $t(q) = \sum_s p(s)t(s, q)$ , где  $p(s)$  обозначает вероятность выбрать секрет  $s$ . Тогда вероятность перехода  $q \rightarrow q'$  равна  $t(q')/t(q)$ . Вероятность выиграть в данном листе  $q$  определяется как условная вероятность того, что  $A$  объявил бы выигравшим игрока, если бы игра

пришла в позицию  $q$ , то есть,

$$\sum_s p(s)t(s, q)p(s, q)/t(q),$$

где  $p(s, q)$  обозначает вероятность того, что  $A$  объявил игрока выигравшим в позиции  $q$ . Вероятности переходов в новом автомате подобраны таким образом, что для него вероятность попасть в позицию  $q$ , если игрок стремится к этому, равна  $t(q)$ .

Пусть зафиксирована некоторая стратегия игрока. Иными словами в дереве позиций для всех вершин четной высоты выбран только один сын, а все остальные сыновья вместе с их потомками уничтожены. Тогда вероятность выигрыша в исходной игре определяется как сумма по всем  $s$  и всем листьям  $q$  урезанного дерева величины  $p(s)t(s, q)p(s, q)$ . А в новой игре — как сумма по всем листьям  $q$  урезанного дерева величины

$$t(q) \frac{\sum_s p(s)t(s, q)p(s, q)}{t(q)}.$$

Если произвести сокращение, то станет ясно, что эти вероятности равны.

Вспомним, что мы собирались доказать равенство

$$p_{A_f^k} = f(p_A, \dots, p_A),$$

для непрозрачных автоматов  $A$ . Пусть  $\tilde{A}$  обозначает эквивалентный  $A$  прозрачный автомат. Нетрудно проверить, что автоматы  $\tilde{A}^k$  и  $\tilde{A}^k$  совпадают (имеют одинаковые вероятности всех переходов). Поэтому автомат  $\tilde{A}^k$  эквивалентен автомату  $A^k$ . А следовательно

$$p_{A_f^k} = p_{\tilde{A}_f^k} = f(p_{\tilde{A}}, \dots, p_{\tilde{A}}) = f(p_A, \dots, p_A).$$

Заметим, что если исходный игровой автомат задавался полиномиальным вероятностным алгоритмом  $V$ , то новый уже не будет, поскольку для него вероятности переходов  $q \rightarrow q'$  уже не вычислимы за полиномиальное время. Однако он задается алгоритмом, работающим на полиномиальной зоне, поскольку вероятность каждого перехода  $q \rightarrow q'$  можно найти перебором, и то же самое относится к вероятности выигрыша в данной позиции. Поскольку цену позиции в игре с полной информацией можно находить рекурсивно на полиномиальной зоне, это доказывает включение IP в PSPACE.



Автомат  $\tilde{V}$  имеет  $k$  каналов общения с игроком и для каждого канала применяет стратегию  $V$ . Для разных каналов стратегия  $V$  применяется независимо. Можно себе представлять  $\tilde{V}$  как вероятностную стратегию, играющую параллельно и независимо друг от друга  $k$  партий в игру, соответствующую  $V$ . В каждый момент состояние автомата является кортежем из  $k$  состояний исходного автомата  $V$  и еще одним битом, определяющим очередь хода. Начальная позиция — это кортеж, состоящий из  $k$  начальных состояний  $V$  и очередь хода принадлежит игроку. В свою очередь стратегия  $\tilde{V}$  в каждой из партий, в которой ее ход, делает выборы, предписываемые стратегией  $V$  и сообщает игроку соответствующую информацию. Для разных партий случайные эти случайные величины независимы. То есть ходы стратегии  $\tilde{V}$  в разных партиях не зависят друг от друга.

Игроку в его очередь, разрешается сделать по одному ходу в любых партиях, в которых его ход, но не менее чем в одной (иначе игра может никогда не закончиться).

Если Доказывающий обязан сделать ход во всех позициях, то такая игра называется параллельным повторением  $V$ . Если сначала нужно сыграть первую игру, затем вторую и так далее, то такая игра называется последовательным повторением  $V$ .

Аналогичным образом можно определить понятие протокола привязки к одному из чисел  $\{1, \dots, m_n\}$  где  $m_n$  зависит от параметра безопасности и растет не более чем с экспоненциальной скоростью:  $m_n \leq 2^{\text{poly}(n)}$ . А именно, как и раньше протоколом привязки называется тройка алгоритмов  $\langle R, S, T \rangle$ , для которых выполнены условия

(а) Для любой неравномерно полиномиально последовательности стратегий  $\{R_n^*\}$  с вероятностью, приблизительно равной 1 (при  $n \rightarrow \infty$ ), для любого  $i \in \{1, \dots, m\}$  выполнено

$$T(c(S_i, R^*), k(R^*, S_\sigma)) \in \{i, \perp_R\}.$$

(для краткости мы опускаем параметр безопасности  $n$  в обозначениях).

(б) Для любой неравномерно полиномиальной последовательности стратегий  $\{S_n^*\}$  для всех  $n$  вероятность того, что существуют  $k_0, k_1$ , для которых

$$T(c(R, S^*), k_0) \text{ и } T(c(R, S^*), k_1)$$

различные элементы множества  $\{1, \dots, m\}$ , пренебрежимо мала, а также

вероятность события

$$\exists k T(c(R, S^*), k) = \perp_R$$

пренебрежимо мала.

(в) Для любой неравномерно полиномиальной последовательности стратегий  $\{R_n^*\}$  случайные величины  $c(S_i, R_n^*)$  для  $i = 1, \dots, m$  вычислительно неотличимы.

Как и раньше, протоколы, удовлетворяющие условиям (а), (б) и (в), называются протоколами привязки к одному из чисел  $\{1, \dots, m_n\}$ .

*Задача 63.* Докажите, что при этих условиях существует функция  $K(c)$  со значениями в  $\{1, \dots, m\}$  такая, что для любой последовательности стратегий  $\{S_n^*\}$  с приблизительно единичной вероятностью для всех  $k$  выполнено  $T(c(R, S^*), k) \in \{K(c(R, S^*)), \perp_S\}$ .

Протокол привязки к элементам множества  $\{1, \dots, m\}$  можно построить из любого протокола  $(S, R, T)$  привязки к одному биту. Сделаем это для случая  $m = 2^{p(n)}$ , где  $p$  — некоторый полином. Числа из  $\{1, \dots, m\}$  отождествим с битовыми строками длины  $p(n)$ . Посылающий и получающий привязываются по очереди к каждому из битов этой строки. Другими словами, новая стратегия  $\tilde{S}$  есть  $p(n)$ -кратное последовательное повторение стратегии  $S$ , новая стратегия  $\tilde{R}$  есть  $p(n)$ -кратное последовательное повторение стратегии  $R$ , а новый алгоритм раскрытия сундучка есть  $p(n)$ -кратное применение старого алгоритма  $T$ . Если хотя бы в одном из применений  $T$  выдал  $\perp_S$  или  $\perp_R$ , то новый алгоритм выдает, соответственно,  $\perp_S$  или  $\perp_R$  (если в одном из применений было выдано  $\perp_S$ , а в другом  $\perp_R$ , то неважно, что выдать). Иначе надо выдать полученную битовую строку.

Истинность условий (а), (б), (в) следуют из свойств последовательного повторения интерактивного протокола.

# Литература

- [1] А. Китаев, А. Шень, М. Вялый. Классические и квантовые вычисления. М.: МЦНМО, 1999, 192 с.
- [2] Eric Bach: How to Generate Factored Random Numbers. SIAM J. Comput. 17(2): 179-193 (1988)
- [3] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, Michael Luby: A Pseudorandom Generator from any One-way Function. SIAM J. Comput. (SIAMCOMP) 28(4):1364-1396 (1999)
- [4] John Rompel. One-Way Functions are Necessary and Sufficient for Secure Signatures. STOC 1990, pages 387–394.
- [5] Zermelo, Ernst. Über eine Anwendung der Mengenlehre auf die Theorie des Schachspiels. Proc. Fifth Congress Mathematicians, (Cambridge 1912), Cambridge University Press 1913, pp. 501–504.