Лекции по математической криптографии

Н.К. Верещагин

18 мая 2015 г.

Оглавление

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 $f_n\colon \mathbb{B}^{k(n)} \to \mathbb{B}^{l(n)}$, где k, l некоторые полиномы.

Мы хотим определить понятие "плохо обратимой" функции. Обратить $f(x)$ означает найти 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 алгоритм 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 алгоритма R. Если оказывается, что с ростом времени работы алгоритма R (в пределах полиномиального) обращение становится возможным всё лучше и лучше, и ошибка может быть сделана меньше любого наперёд заданного полинома, то это не годится. Если это разрешить, то получится более слабое определение. А именно, назовём семейство *совсем слабо необратимым*, если для любого полиномиального вероятностного алгоритма R существует некоторый многочлен p с таким свойством: вероятность события

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

не превосходит

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

для всех достаточно больших n. От определения слабой необратимости это требование отличается перестановкой кванторов по многочлену p и алгоритму R.

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 $n \to \infty$ 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 вычислимые семейства функций называются *сильно односторонними*. Аналогично, слабо необратимые семейства, вычислимые 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, $x_1, \ldots, x_n$ и $r_1, \ldots, r_s$, называемых основными и случайными входами. Для такой схемы мы будем обозначать через $C(x_1, \ldots, x_n)$ случайную величину, для которой при каждой последовательности битов $x_1, \ldots, x_n$ на основном входе вероятность события $C(x_1, \ldots, x_n) = y$ равна доле

слов $r_1, \ldots, r_s$, для которых $C(x_1, \ldots, x_n, r_1, \ldots, r_s) = y$.

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) одностороннего 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. Аналогичным образом определяются и совсем слабо необратимые семейства для неравномерного противника.

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 $n \to \infty$ 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. Запаяв 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 совсем слабо необратимым семейством функций даже для равномерного противника.

Задача 3. Докажите, что для равномерного противника сильно необратимые функции существуют (не обязательно полиномиально вычислимые).

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

Задача 5. Пусть $f_n : \{0,1\}^n \to \{0,1\}^n$ выбирается случайно среди всех функций (необязательно 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. Докажите, что существуют слабо необратимые функции, не являющиеся сильно необратимыми. Это верно как для равномерного, так и для неравномерного противника.

Задача 7. Пусть даны две биективные полиномиально вычислимые функции $f_n, g_n : \{0,1\}^n \to \{0,1\}^n$. 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 8. Докажите, что существуют сильно необратимые функции $f_n, g_n : \{0,1\}^n \to \{0,1\}^n$, композиция которых не является даже слабо необратимой (полиномиальная вычислимость функций не требуется). Это верно как для равномерного, так и для неравномерного противника.

Задача 9. Для неравномерного противника: пусть множество значений функции $f_n : \{0,1\}^n \to \{0,1\}^n$ имеет не более $\mathrm{poly}(n)$ элементов. Докажите, что тогда f_n не является даже совсем слабо необратимым семейством функций.

Задача 10. Докажите то же самое для равномерного противника.

Задача 11. Пусть даны сильно [слабо] необратимая функция $f_n : \{0,1\}^n \to \{0,1\}^n$. Докажите, что тогда функция $xy \mapsto f_n(x)y$ определенная на словах длины $2n$ сильно [слабо] необратима. Это верно как для равномерного, так и для неравномерного противника.

Задача 12. Пусть даны два семейства полиномиально вычислимых функций $\{f_n\}$, $\{g_n\}$ с одинаковой областью определения. Рассмотрим новую функцию $h_n(x) = f_n(x)g_n(x)$ (имеется в виду конкатенация значений двух функций). Задача обращения этой функции может как упроститься по сравнению с задачей обращения каждой из исходных функций (за счет добавления новой информации об x), так и усложниться из-за того, что количество прообразов uv относительно новой функции может быть значительно меньше и количества прообразов u относительно f_n, и количества прообразов v относительно g_n. Следующие примеры показывают, что возможно и то, и другое.

(а) Приведите такой пример, когда f_n сильно односторонняя, а h_n не является даже слабо односторонней.

(б) Приведите пример, когда обе функции f_n, g_n сильно односторонние, а h_n не является даже слабо односторонней.

(в) Приведите обратный пример, когда h_n сильно односторонняя, а f_n не является даже слабо односторонней.

(г) Приведите пример, когда h_n сильно односторонняя, и обе функции f_n, g_n не являются даже слабо односторонними.

(д) Докажите, что если длина $g_n(x)$ есть $O(\log n)$, то из сильной [слабой] необратимости функции f_n следует сильная [слабая] необратимость функции h_n.

(е) Приведите пример, когда длина $g_n(x)$ равна 1 и при этом h_n сильно необратима, а f_n не является даже слабо необратимой.

Задача 13. Докажите, что определение сильно необратимой функции для неравномерного противника можно переформулировать следующим образом: для любого полинома q существует последовательность чисел ε_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 ε_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 вычислимое семейство функций f_n не является слабо необратимым 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 Λ 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 срока z длины i, для которой мы знаем, что y имеет прообраз, начинающийся на z. После $k(n)$ шагов мы найдем некоторый прообраз целиком. □

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 $x_1, \ldots, x_n$ в двоичной записи. Последний блок будем понимать как подмножество I множества $\{1, 2, \ldots, n\}$. По определению $f_n(x)$ равно конкатенации $x_1, \ldots, x_n$ 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 6.)

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

$$f^N(x_1x_2\ldots x_N) = f(x_1)f(x_2)\ldots f(x_N).$$

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 $f^N(x_1\ldots x_N)$ для случайного $x_1\ldots x_N$ требует одновременного обращения $f(x_1),\ldots,f(x_N)$ при независимых случайных $x_1,\ldots,x_N$. 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 $f(x_2), \ldots, f(x_N)$ для случайных слов $x_2, \ldots, x_N$. Применим к полученному слову

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

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 $f(x_1)\ldots f(x_N)$ алгоритмом R_{f^N} для случайных $x_1 \ldots x_n$. 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 может ставить обращаемое 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 обращаемое слово y на i-е место и окружим его $N-1$ словами, получив набор

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

Слова $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_N$ 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

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

для случайно выбранных $x_2, \ldots, x_N$. 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 $s_2(x_2), \ldots, s_N(x_N)$.

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

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

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 ε, 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 ε, 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 ε — некоторые положительное число, порядка $1/\mathrm{poly}(n)$. Будем называть слово x "трудным", если $\hat{s}(x) < \varepsilon$. Обозначим через δ 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 $f^N(x_1 \ldots x_N)$ (для случайных независимых $x_1, \ldots, x_N$) меньше

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

При подходящем выборе параметров ε, 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 $f^N(x_1, \ldots, x_N)$ 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 δ, и для каждого из трудных слов x_i условная вероятность успешной работы алгоритма R_{f^N} (при данном значении i-й координаты) меньше ε. В итоге получаем для первой части оценку $N\delta\varepsilon$. □

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 ε 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 δ не меньше $1/2p(n)$. Теперь воспользуемся вторым утверждением леммы для оценки сверху вероятности успеха R_{f^N}. Нам надо выбрать ε, 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 ε 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 14. 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 μ_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 остановится за полиномиальное от n время и выдать некоторую двоичную строку.

Статистическим расстоянием между распределениями вероятностей μ и ν 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 μ_n и ν_n, называются *статистически неотличимыми*, если статистическое расстояние между μ_n и ν_n стремится к нулю быстрее любого обратного многочлена от n. Аналогично определяется статистическая неотличимость случайных величин: случайные величины α_n и β_n, зависящие от натурального параметра n, называются *статистически неотличимыми*, если их распределения $\mu_n(x) = \Pr[\alpha_n = x]$ и $\nu_n(x) = \Pr[\beta_n = x]$ статистически неотличимы.

Будем называть распределение μ_n *доступным*, если μ_n статистически неотличимо от некоторого полиномиально моделируемого распределения ν_n.

Примеры.

1. Пусть μ_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[1]. 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 смоделированным 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 μ_n есть равномерное распределение на множестве всех последовательностей длины n, кроме последовательности из одних единиц. Опять μ_n не является полиномиально моделируемым, но является доступным. В самом деле, статистическое расстояние между равномерным распределением на всех словах длины n и μ_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 μ_n на области определения f_n доступно. Аналогично определяются слабо необратимые семейства частичных функций. Семейство частичных функций f_n называется *сильно односторон-*

[1] Это можно было бы сделать за полиномиальное в среднем время, но такие алгоритмы мы не рассматриваем.

ней 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 xy (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 простого с y. Значение функции на аргументе xy 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 следует 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 мень-

ше $(1-\varepsilon/n)^{n^2} < e^{-\varepsilon n}$ для некоторого положительного ε. (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 вероятность 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 вычетов 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 вполне очевидна из-за того, что надо уметь порождать числа, взаимно простые с $N = (p-1)(q-1)$. Будем это делать обычным образом, то есть, выбирать случайно число, меньшее N до тех пор, пока не найдем, число, взаимно простое с N (но не больше n^2 раз). Вероятность найти такое число с одной попытки равна доле взаимно простых с N чисел среди всех меньших N чисел. Эта доля не меньше $1/(k+1)$, где k есть количество простых делителей N. В самом деле, пусть эти делители суть $p_1, \ldots, p_k$. Тогда эта доля равна $(1-1/p_1)(1-1/p_2)\cdots(1-1/p_k)$ ($(1-1/p_1)$-ая часть всех чисел не делится на p_1, среди них $(1-1/p_2)$-ая часть не делится на p_2 и так далее). Ясно, что $p_i \geq i+1$, поэтому эта доля не меньше $(1-1/2)(1-$

$1/3)\cdots(1-1/(k+1)) = 1/(k+1)$. Осталось заметить, что $k \le \log_2 N \le 2n$, поэтому достаточно провести n^2 попыток, чтобы сделать вероятность неудачи пренебрежимо малой.

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 $x^z \bmod y$:

$$f_n(xyz) = xy(x^z \bmod y).$$

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 $f^n : \{0,1\}^{k(n)} \to \{0,1\}^{l(n)}$ называется *проверяемой*, если по любому 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, 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 $f_n^e : D_n^e \to D_n^e$ 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. Слова e, d будут называться *открытым и закрытым ключами*.

Теперь формальное определение. Пусть задан полиномиальный вероятностный алгоритм K, который получив на вход 1^n генерирует пару слов $\langle e, d\rangle$ или выдает $\perp$ (символ неудачи), причем последнее происходит с пренебрежимо малой вероятностью. Будем через A_n обозначать первые компоненты всех возможных пар $\langle e, d\rangle$, генерируемых алгоритмом K на входе 1^n. Пусть для каждого $e \in A_n$ задана перестановка f_n^e некоторого множества слов D_n^e длины $\mathrm{poly}(n)$. 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 $\mathrm{poly}(n)$ 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 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 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 $0, \ldots, pq - 1$, при этом $f_n^e(x) = x^2 \bmod pq$. Условия (а), (в) и (г) очевидно выполнены. Условие (б) будет выполнено 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, $i, j \in \{1, \ldots, pq - 1\}$ и $ij \equiv 1 \bmod (p - 1)(q - 1)$. (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 γ_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 ξ_n^e статистически неотличимую от $\gamma_n^{e_n}$. При этом требуется

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 $\mathrm{poly}(n)$ 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 15. 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 16. Предположим, существует сильно односторонняя частичная функция. Докажите, что тогда существует и сильно односторонняя всюду определенная функция.

Задача 17. Предположим, существует слабо односторонняя частичная функция. Докажите, что тогда существует и слабо односторонняя всюду определенная функция.

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

Задача 19. 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 20. 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 α_n и β_n, зависящие от натурального параметра n, со значениями в множестве слов некоторой длины $l(n)$ называются вычислительно неотличимыми, если для любой последовательности схем $C_0, C_1, \ldots, C_n, \ldots$ размера $\mathrm{poly}(n)$ (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 α_n проходит тест C_n, если $C_n(\alpha_n) = 1$. Таким образом, мы требуем, чтобы α_n и β_n проходили любые тесты полиномиального размера с приблизительно равной вероятностью.

Для равномерного противника в этом определении надо заменить последовательность схем $C_0, C_1, \ldots, C_n, \ldots$ 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 α_n и β_n статистически неотличимы, то они и вычислительно неотличимыми (для неравномерного противника), поскольку разность вероятностей попадания α_n и β_n в множество $\{x \mid C_n(x) = 1\}$, задаваемое тестом C_n, не превосходит статистического расстояния между α_n и β_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 α_n и β_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 α_n и β_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 α_n, β_n и γ_n имеют совместное распределение. И пусть для любой последовательности значений c_n случайной величины γ_n случайные величины $(\alpha_n|\gamma_n = c_n)$ и $(\beta_n|\gamma_n = c_n)$ вычислительно неотличимы.[1] *Тогда $\alpha_n\gamma_n$ и $\beta_n\gamma_n$ вычислительно неотличимы.*

Для равномерного противника это свойство справедливо только для независимых случайных величин α_n, γ_n, причем случайная величина γ_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 α_n и β_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 α_n проходит тест D_n пренебрежимо мало отличается от вероятности того, что β_n проходит тест D_n. Осталось заметить, что первая вероятность равна вероятности

[1] $(\alpha_n|\gamma_n = c_n)$ обозначает случайную величину с тем же множеством значений, что и у случайной величины α_n, которая принимает значение a с вероятностью $\Pr[\alpha_n = a|\gamma_n = c_n]$.

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 $\varepsilon = 1/\mathrm{poly}(n)$ 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 γ_n). Поэтому для бесконечно многих n найдется $c = c_n$ в множестве значений γ_n, для которого эта разность не меньше ε по абсолютной величине. Запаивая 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. □

Задача 21. Пусть случайные величины α_n и β_n вычислительно неотличимы. Докажите, что в результате отрезания от них начала полиномиальной длины получатся вычислительно неотличимые случайные величины.

Задача 22. Пусть случайные величины α_n и β_n вычислительно неотличимы, а γ_n произвольная случайная величина, независимая от α_n и β_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 α_n^u со значениями в некотором множестве слов полиномиальной от n длины (эта длина зависит только от n). Мы говорим, что случайные величины α_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 α_n^u попарно вычислительно неотличимы тогда и только тогда, когда любой последовательности схем T_n полиномиального размера найдется пренебрежимо малая последовательность ε_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 α_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 ε_n абсолютную величину этой разности для этих u_n, v_n. По условию ε_n пренебрежимо мало. □

Задача 23. 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 $k(n) \to l(n)$ 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 24. Докажите, что второе требование нельзя усилить, потребовав статистической неотличимости $G_n(s)$ и равномерно распределенной случайной величины.

Задача 25. Докажите, что композиция генераторов ПСЧ типа $k(n) \to l(n)$ и $l(n) \to m(n)$ является генератором ПСЧ типа $k(n) \to m(n)$.

Определим также генераторы ПСЧ типа $k(n) \to \infty$, 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 ω_n, ξ_n со значениями в множестве всех бесконечных двоичных последовательностей понимается в следующем смысле: для любой последовательности схем C_n размера $\mathrm{poly}(n)$ 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 ω_n, где k — количество входов C_n. Аналогично понимается $C_n(\xi_n)$. Другими словами, ω_n и ξ_n вычислительно неотличимы, если для любого полинома $p(n)$ вычислительно неотличимы случайные величины, равные и первым $p(n)$ битам случайных величин ω_n и ξ_n.

Имеется следующее соотношение между генераторами типа $k(n) \to l(n)$ и генераторами типа $k(n) \to \infty$. Если G_n — генератор типа $k(n) \to \infty$, 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 $k(n) \to l(n)$ (что очевидно). Обратно, мы докажем, что по любому генератору типа $k(n) \to l(n)$ (какие бы ни были $k(n) < l(n)$) можно построить генератор типа $k(n) \to \infty$.

Задача 26. Докажите, что существует функция G_n (не обязательно вычислимая за полиномиальное время), типа $n \to n+1$ такая, что случайная величина $G_n(s)$ вычислительно неотличима от равномерно распределенной случайной величины.

Задача 27. Докажите, что если функция G_n является генератором типа $k(n) \to l(n)$, а x_n последовательность слов длины $l(n)$, вычислимая за время $\mathrm{poly}(n)$, то функция $s \mapsto G_n(s) \oplus x_n$ является генератором.

3.3 Генераторы ПСЧ и односторонние функции

Любой генератор G_n типа $k(n) \to l(n)$ является слабо необратимой функцией. Действительно, никакая последовательность схем C_n полиномиального размера не может обратить $G_n(s)$ в вероятностью 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 $k(n) \to l(n)$, 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 $n \to \infty$.*

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

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

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 β_n, γ_n последовательность пар совместно распределенных случайных величин, причем случайная величина β_n имеет значения 0,1. Мы говорим, что β_n является трудно вычислимой по γ_n если для любой последовательности схем из функциональных элементов C_n размера $\mathrm{poly}(n)$

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 β_n нельзя вычислить по γ_n с вероятностью существенно лучшей, чем при простом угадывании. Если β_n является трудно вычислимой по γ_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 $h_n : D_n \to \{0,1\}$ является трудным битом для функции $g_n : D_n \to D_n$, если случайная величина $h_n(\alpha_n)$ трудно вычислима по случайной величине $g_n(\alpha_n)$, где α_n — случайная величина, равномерно распределенная в D_n.

Аналогично определяется понятие трудного бита для односторонней перестановки с секретом. Назовем семейство функций $h_n^e : D_n^e \to \{0,1\}$, 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 28. Докажите, что если имеется трудный бит для полиномиально вычислимой перестановки $g_n : D_n \to D_n$, 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 β_n, γ_n, причём β_n принимает значения в множестве 0,1. Пусть случайная величина r_n равномерно распределена в 0,1 и независима от γ_n. Тогда случайная величина β_n трудно вычислима по γ_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 β является трудным битом для γ. (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 $\varepsilon = 1/\mathrm{poly}(n)$. 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 $\mathrm{poly}(n)$ такая, что для бесконечно многих n выполнено

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

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

Пусть нам надо некоторое возможное значение x случайной величины γ. Нам нужно, зная x, выдать по возможности правильный прогноз для значения γ. 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 β, чем случайный бит, значит β скорее равно 1, чем 0). Если $D(0x) = 1$, $D(1x) = 0$, выдаем 0.

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

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

(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 γ. $\square$

Пусть h_n — трудный бит для односторонней перестановки $g_n : D_n \to D_n$, а случайная величина α_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 нефомально, случайная величина $h_n(\alpha_n)g_n(\alpha_n)$ имеет на один бит случайности больше, чем α_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 α_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

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

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

$$r_1, r_2, \ldots, r_{p(n)}, \alpha, \tag{3.3}$$

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

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

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 $\varepsilon_n = 1/\mathrm{poly}(n)$, 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, α 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, α. Случайная величина H_{i+1} получена некоторым преобразованием T строки $r_1 r_2 \dots r_i r_{i+1} \alpha$, вычисляемым схемой размера $\mathrm{poly}(n)$. 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}, α и $h\alpha, g\alpha$. □

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

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

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

и $e, r_1, r_2, \ldots, r_{p(n)}, \gamma^e$ 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 $q(n) \to \infty$.*

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

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

По Лемме 3.4 это отображение является генератором ПСЧ. Действительно, лемма утверждает, что любое начало этой последовательности полиномиальной длины вычислительно неотличимо 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 α_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

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

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 α_n. Поэтому и первые $p(n)$ битов $G_n(s)$ неотличимы от

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

тестами того же размера.[2] По Лемме 3.4 последняя неотличима от равномерно распределенной последовательности тестами размера $\mathrm{poly}(n)$. $\square$

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 любые $p(n)$ 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 $l, l+1, \ldots, l+p(n)-1$, 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 α_n и $g_n(\alpha_n)$. Поэтому в условии леммы требование инъективности g_n можно заменить на более слабое требование: случайные величины α_n и $g_n(\alpha_n)$ вычислительно неотличимы (где α_n равномерно распределена в D_n). Более того, совершенно необязательно, чтобы случайная величина α_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 α_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 α_n доступная случайная величина, вычислительно неотличимая от $g_n(\alpha_n)$. Тогда для любого полинома $p(n)$ случайные величины*

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

[2] В этом доказательстве нам достаточно всего лишь вычислительной неотличимости $K_n(s)$ и α_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 α_n.

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

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 $k(n) \to k(n)+1$, то для некоторого полинома $q(n)$ существует генератор ПСЧ типа $q(n) \to \infty$.*

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 $g_n : \{0,1\}^{k(n)} \to \{0,1\}^{k(n)}$ вместе с трудным битом для нее. Осталось воспользоваться Теоремой 6. □

Задача 29. Докажите, что если для некоторого полинома $q(n)$ существует генератор ПСЧ типа $q(n) \to \infty$, то существует и генератор ПСЧ типа $n \to \infty$.

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 α^n принимает значения в в множестве бесконечных последовательностей нулей и единиц. Мы будем говорить, что α^n неотличима по Яо от равномерно распределенной последовательности,

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

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 α^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 α^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 α^n заменить все биты, начиная с $i+1$-ого, на случайные биты, независимые от α^n. Ясно, что начало $H_{l(n)}$ длины $l(n)$ распределено так же, как начало α^n длины $l(n)$, а H_0 равномерно распределена. Обозначим через i_n тот индекс, для которого $|\Pr[C_n(H_i)=1]-\Pr[C_n(H_{i+1})=1]|$ максимально. Сравним случайные величины H_i и H_{i+1}. У них первые i битов равны i первым битам α^n. Последние $n-i-1$ битов выбираются случайно. Разница только в $i+1$-ом бите, который в H_i выбирается случайным образом, а в H_{i+1} берется из α^n. Поскольку α^n неотличима по Яо от равномерно распределенной последовательности, α_{i_n+1} (верхний индекс

[3] Можно дать определение неотличимости по Яо любых двух случайных величин в множестве слов одной длины, но оно нам не понадобится.

[4] Аналогично дается определение надежности по Яо генераторов вида $k(n) \to l(n)$.

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 α). 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. □

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

$$y_1, \ldots, y_n \mapsto x_1y_1 + \cdots + x_ny_n$$

из n-мерного линейного пространства над полем $\mathbb{B}$ из 2 элементов в $\mathbb{B}$. *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/ε 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 $g\colon \mathbb{B}^n \to \mathbb{B}$, и алгоритм может её "вызывать"). Это важно, поскольку, если бы g подавался

на вход в виде слова, то за полиномиальное от n/ε время алгоритм смог бы прочесть только его начало небольшой длины. В качестве ε 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 ε, получив на вход n и ε, 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/ε 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

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

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

$$(1, 0, \ldots, 0), (0, 1, \ldots, 0), \ldots, (0, 0, \ldots, 1)$$

(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 $\mathbb{B}^n$ выполняется равенство

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

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

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

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.

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 $A_1, \ldots, A_N$ 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 больши́х 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 $\mathbb{B}^n$, 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 $u_1, \ldots, u_s$. 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 $f(u_1), \ldots, f(u_s)$, то есть s битами. Возможных вариантов этих значений 2^s, то есть полиномиальное от n/ε количество. Вспомним, что нам разрешалось отгадывать с нескольких попыток: алгоритм $\mathcal{B}$ выдаёт не один набор коэффициентов, а полиномиальный (от n/ε) 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 $u_1, \ldots, u_s$ и получаем из них попарно независимые векторы $r_1, \ldots, r_{2^s-1}$. 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 $f(u_1), \ldots, f(u_s)$ и проводится 2^s раз, для всех возможных вариантов. Подчеркнём, что выбор случайных векторов $u_1, \ldots, u_s$ 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 $i = 1, 2, \ldots, 2^s - 1$) 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

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

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

Задача 30. Постройте вероятностный алгоритм, который для любого слова x длины n и любых ε, δ, получив на вход n, ε, δ 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 31. 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 $f_n : D_n \to D_n$ 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 $x \odot y = \sum_{i=1}^{l(n)} x_i y_i \bmod 2$. 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 $\mathrm{poly}(n)$ такая, что вероятность события $C_n(f(x)y) = x \odot y$ не меньше $1/2 + \varepsilon_n$, где $\varepsilon_n = 1/\mathrm{poly}(n)$ (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 ε. 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. □

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 односторонняя проверяемая перестановка 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

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 ε_n пренебрежимо мало. □

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 α_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 α_n, а y независимо от x и распределено равномерно среди слов той же длины, что и α_n.*

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

Доказательство. Пусть даны перестановка g_n множества D_n и трудная для нее случайная величина α_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 α_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 неотличимо от $(g_n(x), y)$. Осталось применить Теорему 6. □

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 ε_n пренебрежимо мало. $\square$

Глава 4

Псевдослучайные функции

Пусть фиксированы многочлены $k(n), l(n)$. Будем рассматривать функции из $\{0,1\}^{k(n)}$ или из $\{0,1\}^*$ в $\{0,1\}^{l(n)}$ с равномерным распределением на них. В первом случае 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. Во втором случае функций вообще бесконечно много и конечным числом битов не обойтись. Наша цель — уменьшить количество случайных битов до некоторого многочлена от n за счет небольшого ухудшения "степени случайности". Точнее, мы хотим построить семейство функций $f_s^n : \{0,1\}^{k(n)} \to \{0,1\}^{l(n)}$ или $f_s^n : \{0,1\}^* \to \{0,1\}^{l(n)}$, где $s \in \{0,1\}^{m(n)}$ и $m(n)$ — некоторый многочлен, удовлетворяющий следующим двум требованиям. В первом из них требуется полиномиальная вычислимость $f_s^n(x)$ по n, s, x. Во втором из них требуется вычислительная неотличимость функции f_s^n от случайной функции того же типа (у случайной функции значения на разных словах независимы и равномерно распределены в множестве значений).

Требование полиномиальная вычислимости формулируется так: существует детерминированный алгоритм, который по s, x и n вычисляет $f_s^n(x)$ за полиномиальное от n и длины x время.

4.1 Слабые семейства ПСФ

Требование вычислительной неотличимости существует в двух вариантах, обычном и усиленном.

Обычное требование вычислительной неотличимости от случайной

функции: пусть для некоторого полинома $p(n)$ для каждого n выбраны $p(n)$ различных строк $x_1^n, \ldots, x_{p(n)}^n\}$ из области определения f_s^n; тогда случайная величина

$$f_s^n(x_1^n) \ldots f_s^n(x_{p(n)}^n) \tag{4.1}$$

вычислительно неотличима от равномерно распределенной последовательности битов той же длины. Здесь s выбирается случайно среди всех слов длины $m(n)$ с равномерным распределением.

Псевдослучайные функции из $\{0,1\}^{k(n)}$ в $\{0,1\}^{l(n)}$ с идентификатором длины $m(n)$ можно понимать и как генераторы псевдослучайных чисел вида $m(n) \to 2^{k(n)}l(n)$. При этом тестирующему генератор алгоритму доступны любые символы выходной последовательности в полиномиальном количестве. Поэтому возникает естественная идея построения семейства ПСФ. Пусть нам дан генератор $G_n : \{0,1\}^{m(n)} \to \{0,1\}^{\infty}$, 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. Требование полиномиальной вычислимости 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). Требование неотличимости не выполнено, поскольку определение генератора гарантирует вычислительную неотличимость только начала полиномиальной 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. Как именно, мы расскажем в доказательстве следующей теоремы.

Определение 4. Семейство функций $f_s^n : \{0,1\}^{k(n)} \to \{0,1\}^{l(n)}$ или $f_s^n : \{0,1\}^* \to \{0,1\}^{l(n)}$ называется *слабым семейством ПСФ*, если оно удовлетворяет требованиям полиномиальной вычислимости и обычному требованию вычислительной неотличимости от случайной функции.

Задача 32. Объясните, как из слабого семейства ПСФ типа $\{0,1\}^* \to \{0,1\}^{l(n)}$ можно изготовить слабое семейство ПСФ типа $\{0,1\}^{k(n)} \to \{0,1\}^{l(n)}$. Объясните, как из слабого семейства ПСФ типа $\{0,1\}^* \to \{0,1\}^{l(n)}$ можно изготовить слабое семейство ПСФ типа $\{0,1\}^{k(n)} \to \{0,1\}^{l'(n)}$ для любого полинома $l' \leq l$.

Теорема 13. *Если существует генератор ПСЧ* $G_n : \{0,1\}^n \to \{0,1\}^{2n}$, *то любых полиномов* $k(n), l(n)$ *существует слабое семейство ПСФ* $f_s^n : \{0,1\}^{k(n)} \to \{0,1\}^{l(n)}$ *с идентификатором* s *длины* $\max\{n, l(n)\}$.

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

Идентификатор случайной функции s будет иметь длину $l(n)$. Пусть s произвольная строка длины $l = l(n)$ и $x = x_1 \dots x_k$ — любое слово длины $k = k(n)$. 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.

Условие полиномиальной вычислимости выполнено. Проверим условие неотличимости от случайной функции. 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 $p(n)l(n)$.

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_Λ (где Λ 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 распределен-

ные 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.

Будем называть это преобразование *модификацией относительно* w. При модификации семейства $\tilde{f}_t$ относительно любого слова w случайная величина $\tilde{f}_t(x_1)\dots\tilde{f}_t(x_{p(n)})$ изменяется незначительно: новая случайная величина будет вычислительно неотличима от исходной. Потому применить к исходному семейству ПСФ f_s не более чем полином модификаций, то для результирующего семейства $\tilde{f}_t$ случайная величина $\tilde{f}_t(x_1)\dots\tilde{f}_t(x_{p(n)})$ будет будет вычислительно неотличима от случайной величины $f_s(x_1)\dots f_s(x_{p(n)})$.

Применим модификации относительно всех начал w слова x_1 (в порядке возрастания длины w), затем относительно всех начал слова x_2, которые не являются началами x_1, и так далее. Всего мы произведем не более $k(n)p(n)$ модификаций. 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 любого полинома $l(n) < n$. Для этого построим сначала ПСФ $f^n_s : \{0,1\}^{k(n)} \to \{0,1\}^n$, а затем модифицируем его, обрезав в $f^n_s(x)$ лишние $n - l(n)$ битов. □

Аналогичным образом можно построить слабое семейство псевдослучайных функций типа $\{0,1\}^* \to \{0,1\}^{l(n)}$ (получающих на вход слова любой длины). Для этого 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. Положим $m(n) = l(n)$ и $f^n_s(x) = s_{\hat{x}}$ (отображение $(w,s) \mapsto s_w$ определено в доказательстве Теоремы 13). Поясним, почему мы в определении заменили x на его префиксный код. Это сделано потому, что для любых слов u, v слово s_{uv} является легко вычислимой функцией слова s_u, и поэтому старое определение не годится.

Теорема 14. *Так определённое семейство функций является слабым семейством псевдослучайных функций.*

Доказательство. Условие полиномиальной вычислимости выполнено. Условие неотличимости от случайной функции доказывается точно так же, как в теореме 13. В том доказательстве нам было достаточно попарной несравнимости слов $x_1, \ldots, x_{p(n)}$. Это позволяло нам модифицировать семейство f_t так, чтобы для всех i по очереди делать значения $f_t(x_i)$ случайными и независимыми от $f_t(x_j)$ при $j < i$. (После модификации семейства относительно всех собственных начал слова x_i функции из полученного семейства становятся неопределёнными для всех собственных начал x_i. Поэтому нам нужно, чтобы слова $x_1, \ldots, x_{p(n)}$ были не только попарно различны, но и чтобы, ни одно из них не было собственным префиксом другого.) □

Задача 33. 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 детерминированного 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 $x_1, \ldots, x_m$ полиномиальной длины (от параметра безопасности) строка $B(n,k,x_1)\ldots B(n,k,x_m)$ имеет распределение, неотличимое от некоторого полиномиально моделируемого распределения. Требование возможности проголосовать: для любого полиномиального количества различных строк $x_1, \ldots, x_m$ полиномиальной длины (от параметра безопасности) пренебрежимо мала вероятность совпадения каких-то из двух значений $B(n,k,x_1)\ldots B(n,k,x_m)$. Постройте такие алгоритмы. (Похожая проблема возникла при выборах в Координационный совет Российской оппозиции в 2012 году.)

4.2 Сильные семейства ПСФ

Можно усилить требование неотличимости, разрешив в нём слову x_2 зависеть от $f_s(x_1)$, слову x_3 — от пары $f_s(x_1), f_s(x_2)$ и так далее. Для

этого 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 *схемы с оракулом*. Так называют схемы, в которых есть элементы для вычисления "внешней" функции g, аргументами и значениями которой являются бинарные слова. Такие элементы мы будем называть *внешними*. Результат работы схемы C с оракулом, которому дана внешняя функция g мы будем обозначать через $C[g]$. *Усиленное требование неотличимости семейства f_s^n от случайной функции* звучит так:

> для любой последовательности схем C_n полиномиального размера с оракулом вероятность события $A[f_s] = 1$ приблизительно равна вероятности события $A[g] = 1$. Здесь s выбирается с равномерным распределением среди строк длины $m(n)$, а g выбирается с равномерным распределением среди всех функций того же типа, что и f_s^n. (То есть, каждый внешний элемент g возвращает случайное слово, не зависимое от всех выданных ранее слов, если на его входе новое слово, и то же самое слово, которое он выдал раньше на входе x, если на его вход подано слово x, которое уже было на входе какого-то из предыдущих внешних элементов схемы.)

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 $f_s : \{0,1\}^n \to \{0,1\}^n$ выбирается так. Выберем случайное слово v и положим $f_s(00\ldots0) = v$ и $f_s(v) = 00\ldots0$. Значение на всех остальных словах выберем случайно и независимо. Тогда для всех s выполнено равенство $f_s(f_s(00\ldots0)) = 00\ldots0$. Это равенство можно проверить полиномиальным алгоритмом с оракулом. При этом, для случайной функции g равенство $g(g(00\ldots0)) = 00\ldots0$ выполнено с пренебрежимо малой вероятностью $2^{-n} + (2^n - 1)2^{-2n}$ (первое слагаемое есть вероятность того, что $g(00\ldots0) = 00\ldots0$, а вторая — вероятность того, что $g(00\ldots0) = v$ и $g(v) = 00\ldots0$ для некоторого v, состоящего не из одних нулей). Таким образом, усиленное требование не выполнено.

С другой стороны, для любой последовательности различных слов $u_1, \ldots, u_{\mathrm{poly}(n)}$ длины n слово v с приблизительно единичной вероятно-

стью отлично от всех слов $u_1, \ldots, u_{\mathrm{poly}(n)}$. Поэтому распределение случайной величины $f_s(u_1), \ldots, f_s(u_{\mathrm{poly}(n)})$ статистически неотличимо от равномерного распределения.

Определение 5. Семейство функций $f_s^n : \{0,1\}^{k(n)} \to \{0,1\}^{l(n)}$ или $f_s^n : \{0,1\}^* \to \{0,1\}^{l(n)}$ называется *сильным семейством ПСФ*, если оно удовлетворяет требованиям полиномиальной вычислимости и усиленному требованию вычислительной неотличимости от случайной функции.

Теорема 15. *Построенное для теоремы 13 семейство ПСФ* $f_s^n : \{0,1\}^{k(n)} \to \{0,1\}^{l(n)}$ *удовлетворяет усиленному требованию неотличимости.*

Доказательство. Пусть дана последовательность схем с оракулом C_n полиномиальной длины. Нам нужно доказать, что вероятность события $c[f_s] = 1$ приблизительна равна вероятности события $A[g] = 1$, где s случайная строка, а g — случайная функция.

Для этого упорядочим все элементы схемы так, что входом любого элемента являются только выходы меньших элементов. Пусть количество внешних элементов в схеме равно k. Рассмотрим $k+1$ следующих мысленных экспериментов. В эксперименте номер $i = 0, 1, \ldots, k$ мы запускаем схему C, при этом первым i внешним элементам даём функционировать как случайная функция g (одна и та же для всех элементов). После того, как мы найдём слова $x_1, \ldots, x_i$, которые подаются на вход первых i внешних элементов, мы приступаем ко второму этапу эксперимента. Для этого применим к семейству функций f_s преобразование T из предыдущего доказательства для всех собственных начал w всех слов $x_1, \ldots, x_i$. В результате мы получим некоторое новое семейство функций. Выберем случайную функцию f из нового семейства и продолжим вычисление схемы, считая, что все остальные внешние элементы вычисляют функцию f. Эксперимент считается успешным, если в результате схема выдала 1.

Полезно заметить, что на всех словах $x_1, \ldots, x_i$ функции f и g принимают одно и то же значение. Это следует из того, как происходит преобразование T. Поэтому результат эксперимента равен $C[f]$.

Нам надо доказать, что вероятность успеха в экспериментах с номерами 0 и k приблизительно равны. Поскольку количество экспериментов ограничено многочленом от n, достаточно доказать это для экспериментов с номерами i и $i+1$ (для любого i). Допустим это не так. Тогда можно зафиксировать значения функции g на входах первых i внешних

элементов так, чтобы результаты успеха в i-ом и $i+1$-ом экспериментах сильно отличались. Причем эти значения для обоих экспериментах можно зафиксировать одинаковым образом. После этого слова на входах первых $i+1$ внешних элементов в обоих экспериментах окажутся одинаковыми. Обозначим их через $x_1, \ldots, x_{i+1}$. Как мы отмечали, результат i-ого эксперимента равен $C[f]$, где f есть функция выбранная из некоторого семейства случайных функций. Если слово x_{i+1} совпадает с одним из слов $x_1, \ldots, x_i$, то результат $i+1$-ого эксперимента равен $C[f]$, где f есть функция выбранная из того же самого семейства, и поэтому вероятности успеха двух экспериментов не различаются. Иначе результат $i+1$-ого эксперимента равен $C[f]$, где f есть функция выбранная из семейства, полученного из первого семейства преобразованием T, выполненным для всех собственных начал слова x_{i+1}. Будем выполнять их по очереди, начиная с более коротких начал. Если после применения одного преобразования для какого-то начала вероятность события $C[f] = 1$ сильно изменится, то схему C можно будет использовать для различения выхода генератора ПСЧ от случайной последовательности. Поэтому при каждом применении преобразования, вероятность события $C[f] = 1$ изменяется незначительно. После применения всех преобразований мы получив вероятность успеха в $i+1$-ом эксперименте. □

Задача 34. Докажите, что семейство ПСФ, построенное в доказательстве теоремы 14 удовлетворяет усиленному требованию неотличимости от случайной функции.

Глава 5

Схемы шифрования

Схемы шифрования состоят из трех полиномиальных вероятностных алгоритмов: *алгоритма шифрования* E, *алгоритма расшифровки* D и *алгоритма генерации ключей* K. Алгоритм K получает на вход параметр безопасности (в унарной записи) и выдаёт в качестве результата пару слов (e, d), называемых *ключами*. Алгоритм E получает на вход параметр безопасности и пару слов, *ключ для шифровки* e и *сообщение* m, а на выход выдаёт некоторое слово, называемое *шифрограммой*. Алгоритм D получает на вход параметр безопасности и пару слов, *ключ для расшифровки* d и шифрограмму, а на выход выдаёт некоторое слово (которое в идеале должно быть равно зашифрованному сообщению).

Любая схема шифрования должна удовлетворять следующему требованию.

(а) Правильность расшифровки: для любой последовательности сообщений x_n длины $\mathrm{poly}(n)$ с вероятностью приблизительно равной 1 выполнено $D(d, E(e, x)) = x$. (Для упрощения обозначений мы опускаем индекс n.)

Источником случайности в этом требовании является не только случайная пара ключей (e, d), но и случайные биты алгоритмов E, D. Распределение на парах ключей (e, d) задается алгоритмом генерации K. Кроме того, мы предполагаем, что случайные биты алгоритмов E, D независимы от пары (e, d).

Кроме того, нужно потребовать, чтобы противник из шифрограммы не получал никакой информации о зашифрованном сообщении, кроме его

длины [1]. При этом мы будем рассматривать разные ситуации: противнику изначально известен ключ для расшифровки (схемы шифрования с *открытым ключом*, или *асимметричные схемы*) и противнику изначально ничего не известно (кроме параметра безопасности). Во втором случае говорят о схемах шифрования с *закрытым ключом*, или *симметричных схемах*. Мы начнем с симметричных схем.

5.1 Одноразовые симметричные схемы шифрования

Для одноразовых схем шифрования с закрытым ключом предполагается, что противник получил доступ лишь единственной шифрограмме. Соответствующее требование формулируется так.

(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 требовании состоит в ключе e (распределение на парах ключей (e, d) задается алгоритмом K) и в случайных битах алгоритмов E, D, которые предполагаются независимыми от e.

Определение 6. Тройка полиномиальных вероятностных алгоритмов (K, E, D) называется одноразовой симметричной схемой шифрования, если она удовлетворяет требованиям (а) и (б), приведённым выше.

Условие (б) очевидно эквивалентно своему частному случаю: неотличимости случайных величин $E(e, a_n)$ и $E(e, 00\ldots0)$ (подряд $p(n)$ нулей). Вторую из этих случайных величин можно генерировать, зная только n и длину сообщения. 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

[1] было бы желательно, чтобы он не узнавал и длины, но таких схем в предположении, что сообщения могут иметь любую длины, неизвестно

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 полиномиального 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 $C(E(e,1a)) = 1$ приблизительна 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 $f : \{0,1\}^* \to \{0,1\}^*$ — некоторая функция). Допустим противник применяет к шифрограмме $E(e,x)$ некоторую схему D_n полиномиального от n размера для извлечения этой информации. Докажите, что существует доступная случайная величина α_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.

Одноразовую схему шифрования с закрытым ключом можно построить на основе любого генератора ПСЧ.

Теорема 16. *Если существует генератор $G_n : \{0,1\}^n \to \{0,1\}^\infty$, 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). Ключи для шифровки и расшифровки в схеме гаммирования совпадают и равны случайно выбранному с равномерным распределением слову γ длины $p(n)$. Шифрограмма равна $E(\gamma,x) = \gamma \oplus x$ (побитовое сложение по модулю 2), а алгоритм расшифровки точно такой же $D(\gamma,y) = \gamma \oplus y$. Если

длина сообщения не равна $p(n)$, то положим $E(\gamma, x) = x$, $D(\gamma, y) = y$; условие (б) в этом случае, разумеется, не выполнено, но это и не требуется.) Условие (а) очевидно выполнено. В условии (б) случайные величины $\gamma \oplus a_n$ и $\gamma \oplus b_n$) имеют одно и то же распределение (равномерное), а потому вычислительно неотличимы.

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 γ на псевдослучайную величину, порожденную генератором. Ключи для шифровки и расшифровки опять совпадают и равны некоторому слову s, равномерно распределенному 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 $G(s)_{p(n)}$ вычислительно неотличима от случайной величины $\gamma_{p(n)}$, равномерно распределенной среди слов длины $p(n)$. Поскольку применение схемы полиномиального размера сохраняет вычислительную неотличимость, случайная величина $a_n \oplus G(s)$ неотличима от случайной величины $a_n \oplus \gamma_{p(n)}$. Случайная величина $a_n \oplus \gamma_{p(n)}$ имеет равномерное распределение, значит случайная величина $a_n \oplus G(s)$ вычислительно неотличима от случайной величины $\gamma_{p(n)}$. То же самое справедливо для случайной величины $b_n \oplus G(s)$, значит она вычислительно неотличима от $a_n \oplus G(s)$. $\square$

5.2 Многоразовые симметричные схемы шифрования

Схему, построенную в доказательстве предыдущей 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? Многоразовая схема не должна разглашать информации, если ее применять с одним и тем же ключом для шифрования полиномиального числа произвольных сообщений:

(б) Для любых полиномов $p(n), q(n)$ и для любой пары последовательностей $x_n^1, \ldots, x_n^{q(n)}$ и $y_n^1, \ldots, y_n^{q(n)}$ из $q(n)$ слов длины $p(n)$ последовательность случайных величин

$$E(e, x^1), E(e, x^2), \ldots, E(e, x^{q(n)}) \text{ и } E(e, y^1), E(e, y^2), \ldots, E(e, y^{q(n)})$$

вычислительно неотличимы (для читаемости мы опустили нижний индекс 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. Схема шифрования однобитовых сообщений может быть построена из семейства ПСФ.

Конструкция многоразовой схемы шифрования из семейства ПСФ. Пусть дано семейство ПСФ $f_s^n : \{0,1\}^n \to \{0,1\}$. Требуется построить многоразовую схему шифрования однобитовых сообщений. Ключи для шифровки и расшифровки совпадают и равны случайно выбранному идентификатору s функции f_s из данного семейства ПСФ. Шифрограмма однобитового 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 $c \oplus f_s^n(z)$:

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

Шифрограмма l-битного сообщения m состоит из l выбранных слов $z_1, \ldots, z_l$ длины n и ещё одного слова длины l: $m_1 \oplus f_s(z_1) \ldots m_l \oplus f_s(z_l)$.

Теорема 17. *Если семейство ПСФ удовлетворяет требованию (б) на стр. 54, то построенная схема шифрования удовлетворяет требованию (б).*

Доказательство. Поскольку мы шифруем сообщения, разбивая его на биты, достаточно проверить условие (б) для пары сообщений x_n и y_n одной длины. То есть, нам надо доказать вычислительную неотличимость шифрограмм $E(e, x_n)$ и $E(e, y_n)$. Для этого докажем, что обе они вычислительно неотличимы от случайной величины $z_1, \ldots, z_{p(n)}, r$, состоящей из независимых слов, выбранных с равномерным распределением среди слов длины где $n, \ldots, n, p(n)$ соответственно.

Напомним, что шифрограмма $E(e, x)$ сообщения x выглядит так:

$$z_1, \ldots, z_{p(n)}, x_1 \oplus f_s(z_1) \ldots x_{p(n)} \oplus f_s(z_{p(n)})$$

Сначала заменим в определении этой случайной величины псевдослучайную функцию f_s на случайную функцию g. Мы получим такую случайную величину:

$$z_1, \ldots, z_{p(n)}, x_1 \oplus g(z_1) \ldots x_{p(n)} \oplus g(z_{p(n)}).$$

Эта случайная величина вычислительно неотличима от шифрограммы в силу того, что семейство ПСФ удовлетворяет свойству (б) на стр. 54.

Затем заменим в полученной случайной величине слово $g(z_1) \ldots g(z_{p(n)})$ на последовательность $r_1 \ldots r_{p(n)}$ из $p(n)$ случайных бит. Полученная последовательность

$$z_1, \ldots, z_{p(n)}, x_1 \oplus r_1 \ldots x_{p(n)} \oplus r_{p(n)}$$

статистически неотличима от исходной, поскольку вероятность того, что какие-то два слова из $z_1, \ldots, z_{p(n)}$ совпадут, пренебрежимо мала. А при условии попарного различия слов $z_1, \ldots, z_{p(n)}$ эти случайные величины имеют одинаковое распределение.

Наконец, полученная последовательность имеет одинаковое распределение с последовательностью

$$z_1, \ldots, z_{p(n)}, r_1 \ldots r_{p(n)}.$$

□

Замечание 2. 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 определение, 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 $x_1E(e,x_1)E(e,x_2)$ вычислительно неотличима от случайной величины $x_1E(e,00\ldots0)E(e,00\ldots0)$, то есть мы разгласили только x_1 и длину x_2.

5.3 Экономная симметричная многоразовая схема шифрования

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.

Пусть дано семейство ПСФ $f_s^n : \{0,1\}^n \to \{0,1\}^n$ и генератор ПСЧ $G_n\{0,1\}^n \to \{0,1\}^\infty$. Построим многоразовую схему шифрования сообщений, у которой длина шифрограммы равна сумме параметра безопасности и длины сообщения. Ключом для шифрования и расшифровки является случайно выбранный идентификатор ПСФ: $e = d = s$.

Пусть z — слово длины l. Шифрограмма сообщения $m \in \{0,1\}$ состоит из случайно выбранного слова z длины l и побитовой суммы сообщения и l первых битов выхода генератора на зерне $f_s^n(z)$: $E(d,m) = m \oplus G_n(f_s^n(z))_l$. Алгоритм расшифровки D в применении к ключу s и шифрограмме (z,c) выдает $c \oplus G_n(f_s^n(z))$:

5.4 Защищенность схем шифрования относительно атаки с выбором сообщений

Более общий вид атаки на многоразовую СШЗК выглядит так. Пусть противник с помощью, используя схему C_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 он нам передаёт. Игра продолжается полиномиальное число ходов $p(n)$. Противник узнаёт шифрограммы некоторых секретных битов и все шифрограммы известных ему битов.

На атаку такого вида можно смотреть, как на общение атакующей схемы с интерактивным алгоритмом, который получает на вход последовательность битов $x_1, \ldots, x_{p(n)}$ и ключ e, по просьбе собеседника шифрует либо очередной бит из своего входа, либо бит, присланный противником. СШЗК считается защищенной относительно такой атаки, если для любого полинома p этот интерактивный алгоритм имеет нулевое разглашение. Для $p(n) = 1$ это следует непосредственно из определения.

Для любого полинома $p(n)$ непонятно, следует ли это из определения. Однако можно доказать, что это верно для схемы шифрования из раздела 5.2. В самом деле, допустим, что существует схема из функциональных элементов, которая атакует схему шифрования с не пренебрежимо малой вероятностью. Заменим в алгоритме шифрования использование псевдослучайной функции на случайную. Тогда вероятность успеха атаки изменится незначительно. При этом мы можем предполагать все слова z, выбранные случайно во время атаки попарно различны (это происходит с вероятностью, приблизительно равной 1). Но тогда все биты используемые при гаммировании независимы. Поэтому все шифрограммы представляют независимо выбранные случайные биты.

5.5 Определение схемы шифрования с открытым ключом

Теперь от схемы шифрования требуется, чтобы противник из открытого ключа и шифрограммы любого сообщения не смог получить никакой информации о сообщении, кроме его длины. 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 случайная 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.

Пусть $\alpha_n, \beta_n, \gamma_n$ совместно распределенные случайные величины. Будем говорить, что случайные величины α_n и β_n *вычислительно (статистически) неотличимы при известном* γ_n, если случайные величины $\alpha_n\gamma_n$ и $\beta_n\gamma_n$ вычислительно (статистически) неотличимы.

Требование неразглашения теперь формулируется следующим образом.

(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. Источником случайности здесь является не только пара ключей $\langle e, d\rangle$ (выбираемая алгоритмом генерации ключей), но и случайные биты алгоритма E. Мы предполагаем, что случайные биты алгоритма E независимы от пары $\langle e, d\rangle$.

Определение 7. Тройка вероятностных полиномиальных алгоритмов (K, E, D) называется *схемой шифрования с открытым ключом или асимметричной схемой шифрования*, выполнено это условие (б), а также условие (а) на стр. 61.

Для существования схемы шифрования с открытым ключом требуется более сильная гипотеза, чем существование генератора. Такую схему удается построить при условии существования односторонней перестановки с секретом (определение дано в следующем разделе).

Задача 36. Докажите, что для асимметричных схем шифрования нет различия между одноразовыми и одноразовыми схемами. Это означает из условия (б) следует и такое его усиление: для любых полиномов $p(n), q(n)$ и для любой пары последовательностей $x_n^1, \ldots, x_n^{q(n)}$ и $y_n^1, \ldots, y_n^{q(n)}$ из $q(n)$ слов длины $p(n)$ последовательности случайных величин

$$E(e, x^1), E(e, x^2), \ldots, E(e, x^{q(n)}) \text{ и } E(e, y^1), E(e, y^2), \ldots, E(e, y^{q(n)})$$

вычислительно неотличимы при известном e.

5.6 Построение асимметричной схемы шифрования

Теорема 18. *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 ξ^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, ξ^e) и (e, γ^e) (где γ^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

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

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

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

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

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 6

Протоколы привязки (bit commitment)

Протоколы привязки имитируют сундучок с замком, в который первый игрок (отправитель) может положить некоторое бинарное слово, 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.

6.1 Неинтерактивные протоколы привязки

Неинтерактивный протокол привязки состоит из двух алгоритмов: вероятностного полиномиального алгоритма S и детерминированного полиномиального алгоритма R. Алгоритм S получает на вход параметр безопасности n, сообщение σ 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 σ 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.

Полнота: для любой последовательности слов σ_n полиномиальной от n длины с вероятностью, приблизительно равной 1, выполнено $R(c(\sigma_n), k(\sigma_n)) = \sigma_n$. Это требование означает, что при открытии сундучка настоящим ключом там обнаруживается положенное туда слово. Если вероятность этого события в точности равна 1, то мы будем говорить, что выполнено условие *сильной полноты*

Неразглашение: Для любых последовательностей слов σ_n, τ_n одной и той же полиномиальной от n длины последовательности случайных величин $c_n(\sigma_n)$ и $c_n(\tau_n)$ вычислительно неотличимы. Это свойство протокола формализует непрозрачность сундучка: не имея ключа, получатель не имеет никакой информации о содержимом сундучка.

Недвусмысленность привязки: ни для какого n не существует таких слов c, k_0, k_1, что слова $R(c, k_0)$ и $R(c, k_1)$ различны и оба не равны $\perp$. Это требование выражает невозможность, подбирая ключи, обнаружить в сундучке два разных слова: любая строка c может быть привязкой только к одному слову.

Определение 8. Пара полиномиальных вероятностных алгоритмов (R, S) называется *протоколом привязки*, если она удовлетворяет требованиям полноты, неразглашения и недвусмысленности привязки.

Замечание 3. Заметим, что зная параметр безопасности и длину σ, по привязке $c(\sigma)$ можно найти σ (с почти единичной вероятностью) с помощью перебора. В самом деле, мы можем перебрать все строки k подходящей длины и для каждой из них применить алгоритм R к паре $c(\sigma), k$ и выдать первый результат, отличный от $\perp$. По условию полноты для некоторого k (а именно для $k = k(\sigma)$) алгоритм R выдаст σ, и по условию недвусмысленности привязки все полученные результаты, отличные от $\perp$, равны σ. Поэтому в требовании неразглашения, что неотличимость $c_n(\sigma_n)$ и $c_n(\tau_n)$ вычислительная, а не статистическая — статистическое расстояние между $c_n(\sigma_n)$ и $c_n(\tau_n)$ близко к 1.

Сначала построим *протокол привязки к биту*, то есть пару алгоритмов (S, R), которые удовлетворяют всем трём требованиям для однобуквенных слов σ. Протокол привязки к биту можно построить, имея любую одностороннюю перестановку $f_n : D_n \to D_n$. 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. Односторонние перестановки, для которых такой алгоритм существует, мы будем называть *правильными*.

Теорема 19. *Если существует правильная односторонняя перестановка $f_n : D_n \to D_n$, 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 c 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 полноты очевидно выполнено. Условие недвусмысленности привязки 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 неразглашения сводится к вычислительной неотличимости 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 γ случайный бит, независимый от x. □

Замечание 4. В доказательстве мы не использовали того, что функция f является наложением, поэтому протокол привязки к биту можно построить, исходя из любой правильной односторонней инъективной функции $f_n : D_n \to \{0,1\}^*$.

Еще полезно отметить, что построенный протокол не удовлетворяет свойству сильной полноты, поскольку Посылающему может сильно не повезти и он не найдёт слова x из области определения f_n.

Задача 38. Докажите, что без ограничения общности можно считать, что в неинтерактивных протоколах привязки к биту ключ равен σr (где σ — 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 слове x, сгенерированном с использованием данных ему случайных битов, равно привязке и в этом случае выдает $h_n(x)$ (а иначе $\perp$). Докажите, что этот протокол удовлетворяет требуемым свойствам.

Задача 40. Из любого протокола привязки к биту можно построить и просто протокола привязки.

Оказывается, протокол привязки можно построить и на основе любой сильно односторонней функции. Но такой протокол оказывается интерактивным. Что такое интерактивные протоколы будет рассказано в следующем разделе.

6.2 Интерактивные алгоритмы: определение

Интерактивные алгоритмы — это алгоритмы, которые, кроме обычного входа, имеют еще входной и выходной каналы обмена информацией. Нас будут интересовать только полиномиальные алгоритмы, которые способны посылать, получать только полиномиальное количество сообщений полиномиальной длины.

Определение 9. *Интерактивным детерминированным или вероятностным полиномиальным алгоритмом* называется пара (A, p), где A — это некоторый полиномиальный (детерминированный или вероятностный) алгоритм, с тремя входами, а p — некоторый многочлен. На первый из входов алгоритма подается параметр безопасности n в унарной записи, на второй вход подаётся собственно вход x алгоритма (бинарное слово), а на третий вход подаётся последовательность сделанных сообщений (слово в алфавите $0, 1, \Diamond$). Многочлен $p(n)$ указывает сколько сообщений надо передать и принять алгоритму.

Пусть имеются два интерактивных алгоритма (A, p) и (B, p) с одним и тем же многочленом p. *Сообщения, посылаемые алгоритмами* $A(n, x)$

и $A(n, y)$ определяются следующим образом.

$$c_1 = A(n, x, \Lambda) \text{ (первое сообщение посылается алгоритмом } A),$$
$$c_2 = B(n, y, c_1\Diamond) \text{ (второе сообщение посылается алгоритмом } B,$$
$$c_3 = A(n, x, c_1\Diamond c_2\Diamond),$$
$$c_4 = B(n, y, c_1\Diamond c_2\Diamond c_3\Diamond),$$
$$\dots$$
$$c_{2p(n)} = B(n, y, c_1\Diamond c_2 \dots \Diamond c_{2p(n)-1}\Diamond)$$

Слово c_i называется *сообщением, посланным в* i*-ом раунде*, а *протоколом общения, или диалогом,* называется последовательность $c = c_1\Diamond c_2 \dots \Diamond c_{2p(n,|x|+|y|)}\Diamond$.

После общения алгоритмы A, B применяются еще один раз к параметру безопасности, входу и полученному протоколу, чтобы получить результаты: алгоритм A печатает результат, равный $A(n, x, c)$, алгоритм B печатает результат, равный $B(n, y, c)$.

Мы также будем рассматривать интерактивные алгоритмы *без входа*. Это означает, что алгоритм игнорирует свой вход x. Если мы говорим об интерактивных алгоритмов *без выхода*, это означает, что нам не важно, какой он результат алгоритм напечатает.

Иногда мы будем рассматривать общение интерактивного алгоритма со схемой из функциональных элементов. Пусть дана схема из функциональных элементов C, количество входов и выходов которой четно и равно, скажем $2m$. С помощью такой схемы мы хотим вычислять отображение из множества слов в алфавите $0, 1, \Diamond$ в множество слов длины не более m из нулей и единиц. Для того чтобы определить значение отображения, задаваемого схемой C на некотором слове w в алфавите $0, 1, \Diamond$, припишем к этому слову столько символов $\#$ или отрежем справа столько символов, чтобы получилось слово длины ровно m. Затем применим схему C к полученному слову в алфавите $0, 1, \Diamond, \#$, закодировав каждый символ этого алфавита двумя битами. Затем выбросим у слова длины m в алфавите $0, 1, \Diamond, \#$, полученного на выходе схемы C, все символы, кроме нулей и единиц. Полученное бинарное слово и будет значением этого отображения на слове w и будет обозначаться через $C(w)$.

Протокол общения (диалог) алгоритма $A(n, x)$ со схемой C определяется следующим образом. Очередное сообщение c_{2i}, посылаемое схемой C в $2i$-ом раунде определяется, как $C(x\Diamond c_1\Diamond \dots c_{2i-1}\Diamond)$. Результат, который выдает схема после окончания общения, определяется как $C(x\Diamond c)$, где c есть протокол общения.

6.3 Интерактивные протоколы привязки

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 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 слово σ, 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 σ. 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 σ и являются зависимыми (при любом значении σ), 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 блокировал протокол. Будем эти сообщения обозначать $\perp_S, \perp_R$ ($\perp_S$ означает, что протокол блокирован посылающим, а $\perp_R$ — получателем).

Тройка алгоритмов T, S, R такого вида называется интерактивным протоколом привязки, если выполнены требования Полноты, Неразглашения и Недвусмысленности привязки. В первом и втором условиях выражены интересы посылающего. Они состоят в том, чтобы при приме-

нении правильного ключа в сундучке обнаруживался положенное туда слово (Полнота) и чтобы сундучок был непрозрачным (Неразглашение). Третье условие выражает интересы получателя — невозможно один и тот же сундучок открыть двумя разными ключами так, чтобы там были обнаружены разные слова. 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 требования Полноты и Неразглашения и для любой последовательности схем $\{S_n^*\}$ полиномиального размера выполнено следующее условие Недвусмысленности привязки.

Полнота. С вероятностью, приблизительно равной 1 (при $n \to \infty$), для любого слова σ алгоритм R на входе $c_{T^*,S(\sigma)}$ и $k_{T^*,S(\sigma)}$ выдает либо σ, либо $\perp_R$ (для краткости мы опускаем параметр безопасности n в обозначениях). Если вероятность этого события в точности равна 1, то мы будем говорить, что выполнено условие *сильной полноты*

Неразглашение. Случайные величины $c_{T^*,S(x_n)}$ и $c_{T^*,S(y_n)}$ вычислительно неотличимы для любой последовательности пар слов $\{(x_n, y_n)\}$ одной длины $|x_n| = |y_n| = \mathrm{poly}(n)$.

Недвусмысленность привязки. Назовём слово *с двусмысленной привязкой*, если существуют такие слова k_0, k_1, что $R(c_{T,S^*}, k_0)$ и $R(c_{T,S^*}, k_1)$ есть два различных слова, или существует такое слово k, что $R(c_{T,S^*}, k) = \perp_R$. Требование состоит в том, что вероятность того, что диалог общения алгоритмов S^* и T является двусмысленной привязкой, пренебрежимо мала.

Заметим, что мы не предполагаем, что схема S_n^* получает на вход некоторое слово σ, 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 все три требования выполнены и для любых последовательностей вероятностных схем S_n^* и T_n^* полиномиального размера. В частности, для любых полиномиальных вероятностных алгоритмов S', T' все три требования условия выполнены для стратегий $S_n^* = S'(1^n)$ и $T_n^* = T'(1^n)$. По этой причине из требований Полноты и Недвусмысленности следует, что для всех слов σ полиномиальной от n длины 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 условие недвусмысленности гарантирует, что вероятность результата $\perp_R$ пренебрежимо мала. В то же время по требованию полноты с приблизительно единичной вероятностью результат равен либо σ, либо $\perp_R$, а значит равен σ.

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 ничего не делает.

6.3.1 Интерактивный протокол привязки к биту

Теорема 20. *Если существует генератор ПСЧ, то существует интерактивный протокол привязки к биту, удовлетворяющий условию сильной полноты.*

Доказательство. Пусть G генератор ПСЧ $\{0,1\}^n \to \{0,1\}^{3n}$. Алгоритм T посылает случайную строку 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 σs.

Входом алгоритма R, кроме n, является пара строк r, x и строка k. Если какие-то строки имеют неправильную длину, то алгоритм R считает, что они состоят из одних нулей в нужном количестве. Если $x = G(s) \oplus \sigma \cdot r$, где σ первый бит строки k, а s — остальные ее биты, то алгоритм R выдает σ. Иначе, алгоритм R выдает $\perp_S$.

Ясно, что требование сильной полноты выполнено. Проверим требование недвусмысленности привязки. 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 неразглашения. 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 сообщение, посланное схемой, имеет длину $3n$. 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

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

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 γ_{3n} равномерно распределенная среди слов длины $3n$ случайная величина. □

Замечание 5. Отметим, что условие недвусмысленности привязки 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 σ.

6.3.2 Интерактивные протоколы привязки к строке

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 данной строки x. Другими словами, новая стратегия $\tilde{S}$ есть $|x|$-кратное последовательное повторение стратегии S, новая стратегия $\tilde{T}$ есть $p(n)$-кратное последовательное повторение стратегии T, а новый алгоритм раскрытия сундучка есть $|x|$-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 42. Докажите, что построенный протокол удовлетворяет всем трём требованиям.

Задача 43. 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 $R(c_{T,S^*}, k) \in \{F(c_{T,S^*}), \perp_S\}$.

Глава 7

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 доверить 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 c, 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 $c_{A,B}$ 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 превосходила $1/2$ 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 6. 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 21. *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. 7.1). Сначала алгоритм A подбрасывает монетку один раз. Обозначим через σ 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 τ алгоритму A. Наконец, алгоритм A посылает алгоритму B ключ k.

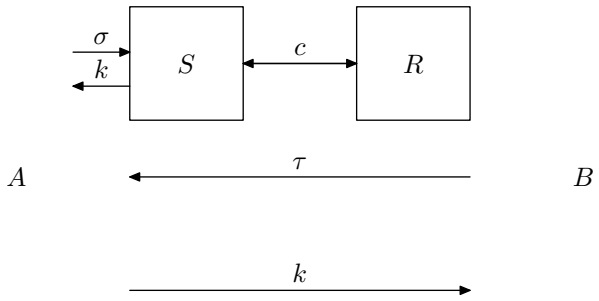

Рис. 7.1: Алгоритмы A и B.

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 τ_n^* полиномиального от n размера, выдающую на выход один бит. Поскольку на обоих стадиях Алиса действует честно, алгоритм S с приблизительно единичной вероятностью выдает бит σ 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 τ^* 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 σ 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 σ 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 τ^*. 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 c, и биту τ, 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 τ. Нам будет важна главным образом зависимость от τ, поэтому мы будем обозначать ключ, выданный схемой k_n^*, через k_τ. Поскольку свой бит τ 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, выдаёт 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. □

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 8

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, ничего не разглашал (алгоритм привязки к слову произвольной длины разглашал только его длину). 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 два слова x, e. Слово x выбирается из данного множества D_n, зависящего от параметра безопасности, а слово слово e выбирается случайно по данному распределению ν_n, также зависящему от параметра безопасности. В примерах, которые у нас будут, вход x будет либо сообщением для шифрования, либо битом для привязки, либо закрытым ключом в протоколах идентификации. Вход e будет открытым ключом в асимметричных схемах шифрования и протоколах идентификации (и будет фиктивным в остальных примерах). Мы будем считать, что для каждого значения параметра безопасности n зафиксировано некоторое полиномиально порождаемое распределение вероятностей на e.

Будем считать, что с алгоритмом $A(x, e)$ (алгоритмом A, получившим на вход x, e и параметр безопасности n, который мы в обозначениях

опускаем) беседует некоторая схема C_n полиномиального от n размера, которая уже имеет некоторую информацию об x, e. (Например, она может знать открытый ключ e, использованный в схеме шифрования.) Эта информация является некоторой функцией g от x, e (и параметра безопасности). [1] Пусть имеется еще одна функция f от x, e (и параметра безопасности). Мы хотим определить, что значит, что алгоритм $A(x, e)$ разглашает только информацию $f(x, e)$, если противнику уже известна информация $g(x, e)$ об паре x, e.

Определение 10 (неразглашения). Мы говорим, что алгоритм A *разглашает только* f *при известном* g *на области* D_n, если существует полиномиальный вероятностный алгоритм M, для которого выполнено следующее. Для любой последовательности слов $\{x_n \in D_n\}$ и любой последовательности схем $\{C_n\}$, для которых длина x_n и размер C_n ограничены некоторым многочленом от n, верно следующее. Пусть алгоритм M получает на вход $f(x_n, e)$ и схему C_n в качестве черного ящика (то есть, алгоритм может только применять схему к любому входу, но не может копаться в её устройстве). Тогда он генерирует случайную величину $\alpha_n(e)$, вычислительно неотличимую от протокола общения $\beta_n(e)$ между алгоритмом $A(x_n, e)$ и схемой $C_n^{g(x_n,e)}$ (схемой C_n, которая до начала общения прочитала g, то есть первые $|g(x, e)| + 1$ входов в схеме зафиксированы равными $g(x, e)\Diamond$) при известном $g(x_n, e)$. То есть, случайные величины $g(x_n, e)\alpha_n(e)$ и $g(x_n, e)\beta_n(e)$ вычислительно неотличимы при стремлении n к бесконечности. Вероятностное пространство в этом определении состоит из e (с заданным распределением ν_n) и случайных битов алгоритмов $A(x_n, e)$ и M. Множество D_n мы будем называть *областью неразглашения алгоритма.*

Алгоритм M, удовлетворяющий этому определению, будем называть *симулятором.* В определении важно, что симулятору разрешается запускать схему C_n ее на любых входах по своему выбору. Смысл этого определения следующий. Противник имеет общеизвестную информацию $g(x, e)$ о входах алгоритма A. Общаясь с алгоритмом $A(x, e)$ с помощью стратегии, реализуемой схемой C, он хочет извлечь некоторую дополнительную информацию о об x, e. Определение гарантирует, что он не может извлечь больше информации, чем содержится в $f(x, e)$: всё, что

[1] На самом деле, все написанное ниже верно в большей общности: g может быть случайной величиной, зависящей от параметров n, x, e. Однако во всех примерах $g(n, x, e)$ будет функцией от n, x, e, поэтому такой общности нам не потребуется.

он узнаёт из общения, но может извлечь самостоятельно с помощью применения симулятора к $f(x, e)$.

Задача 46. Докажите, что любой алгоритм A разглашает только $(e, x, g(x, e))$ при известном $g(x, e)$.

В определении 10 не подразумевается никакого распределения x, требование должно быть выполнено для всех слов $x \in D_n$ полиномиальной длины от n длины. Нам также понадобится ослабленное определение, в котором требуется, чтобы это было выполнено для почти всех x.

Определение 11 (слабого неразглашения). Пусть для каждого n задано некоторое распределение вероятностей μ_n на словах полиномиальной от n длины. Мы говорим, что алгоритм A *разглашает только f при известном g в слабом смысле*, если существует полиномиальный вероятностный алгоритм M, для которого для любой последовательности схем $\{C_n\}$ полиномиального размера требование из определения 10 выполнено с вероятностью приблизительно 1 при случайно выбранном x по распределению μ_n (при стремлении параметра безопасности n к бесконечности).

Замечание 7. В определении неразглашения можно заменить кванторы всеобщности по последовательностям слов и схем полиномиального размера на кванторы всеобщности по словам данной длины и схемам данного размера. Для этого в определениях в соответствующем месте надо сказать так: для любых полиномов p, q, r найдется пренебрежимо малая величина α_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 "$T(g$, диалог $A(x, e)$ и $C(g)) = 1$" и "$T(g, M(C, f))) = 1$" отличаются не более чем на α_n.

Замечание 8. Определение неразглашения тем сильнее, чем меньше информации в f и больше в g. Это означает следующее.

(1) Для любых функций f, g, h, если A разглашает только f при известном g, то A разглашает только пару (f, h) при известном g. В самом деле, новый симулятор M' в применении к паре (f, h) и схеме C применяет старый симулятор M к f и C, игнорируя дополнительную информацию h.

(2) Для любых функций f, g, h, если A разглашает только f при известной паре (g, h), то A разглашает только f и при известном g. В самом деле, новый симулятор M' в применении к f и схеме C применяет старый симулятор M к f и схеме D, которая построена так, что прочтя g, h,

она выбрасывает h и работает так же, как $C(g)$. По условию случайные величины $(g, h$, протокол общения $\mathcal{A}(x, e)$ и $D(g, h))$ и $(g, h, M(D, f))$ вычислительно неотличимы. Значит и случайные величины (g, протокол общения $\mathcal{A}(x, e)$ и $D(g, h))$ и $(g, M(D, f))$, получаемые вычёркиванием h, вычислительно неотличимы. Осталось заметить, что $D(g, h) \equiv C(g)$, и поэтому протокол общения $\mathcal{A}(x, e)$ и $D(g, h))$ совпадает с протоколом общения $\mathcal{A}(x, e)$ и $C(g)$.

Определение 12 (нулевого разглашения). Если A разглашает только g при известном g (то есть f и g — это одна и та же функция), то мы говорим, что A имеет *нулевое разглашение при известном* g. Если g постоянная функция, то мы говорим, что A *разглашает только* f. Если при этом еще и f постоянная функция, то мы говорим, что A имеет *нулевое разглашение*.

Пример 1. Пусть e есть случайное слово длины n, а алгоритм A посылает побитовую сумму (по модулю 2) слова x длины n и e и затем останавливается (если длина одного из слов x, e отлична от n, то просто выдается случайная строка длины n). Этот алгоритм имеет нулевое разглашение. Симулятор выдает на выход случайную строку длины n (схему C_n он не использует, поскольку алгоритм A не интерактивный и поэтому запись общения A и C_n не зависит от C_n).

Этот пример очень похож на схему шифрования с помощью гаммирования (x есть шифруемое сообщение, а e — ключ). Вообще, в любой схеме K, E, D шифрования (с закрытым или открытым ключом) алгоритм $E(e, x)$ разглашает только длину шифруемого сообщения x. Действительно, по условию шифрограмма $E(e, x)$ сообщения x вычислительно неотличима от шифрограммы $E(e, 00\ldots0)$ сообщения той же длины. Поэтому симулятор M, зная длину зашифрованного сообщения l, может выбрать случайный ключ для шифрования e и выдать на выход $E(e, 0^l)$. Схему C он не использует, поскольку алгоритм A опять не интерактивный.

Этот пример показывает, что в определении разглашения важно, что входы x и e трактуются по-разному. А именно, в определении разглашения требуется, чтобы *для всех* (или почти всех) x, случайная величина, равная протоколу общения между A и C при *случайном выборе* e, могла быть порождена по f. В примере 1 невозможно порождать протокол общения *для любых* x, e, зная только длину x.

Различие в трактовке x и e можно увидеть и на более простом при-

мере. Рассмотрим вероятностный алгоритм, игнорирующий свой вход x и печатающий на выход e, которое распределено равномерно независимо от x. Он имеет нулевое разглашение, однако алгоритм, который печатает на выход свой вход x не имеет нулевого разглашения.

Пример 2. Пусть дана любая схема шифрования K, E, D с открытым ключом. Обозначим через e открытый ключ. Тогда алгоритм $x, e \mapsto E(x, e)$ разглашает только пару $(|x|, e)$ при известном e. Действительно, по условию $E(e, x)$ вычислительно неотличима от $E(e, 0^{|x|})$ при известном e. Поэтому симулятор M, зная длину зашифрованного сообщения l и открытый ключ e, выдает на выход $E(e, 0^l)$.

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.

Следующий пример показывает, что нулевое разглашение при известном e немонотонно по e.

Пример 4. Рассмотрим алгоритм A, поучив входной бит x, складывает его со случайным битом e и выдает полученный бит на выход (если на входах не ровно один бит, то просто выдается случайный бит). Этот алгоритм имеет нулевое разглашение, но не имеет нулевого разглашения при известном e, и имеет нулевое разглашение при известной паре (e, x).

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

Определение 13. Пусть B вероятностный интерактивный алгоритм, который получает на вход параметр безопасности и e. Мы говорим, что алгоритм A *разглашает только* f *при известном* g *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, порождает случайную величину, вычислительно неотличимую от записи общения алгоритмов $A(e, x)$ и $B(g, s)$ при известных g, s. Здесь s обозначает случайные биты алгоритма B. В этом контексте случайная величина (запись общения алгоритмов $A(e, x)$ и $B(g, s)$, s, g) обозначается через $\mathrm{view}_B(A(e, x), B(g))$.

Задача 47. 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 g.

8.1 Неразглашение информации при последовательном выполнении алгоритмов (Sequential composition lemma)

Установим одно важное свойство понятия неразглашения. Для этого определим операцию последовательного соединения вероятностных интерактивных алгоритмов. Пусть A, B вероятностные полиномиальные алгоритмы. Выполним сначала алгоритм A на данной паре входов x и e, а затем выполним алгоритм B на паре входов (x, c) и e, где c — протокол общения, полученный в ходе выполнения A. Полученный алгоритм обозначим через (A, B).

Последовательное соединение алгоритмов 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 $x = \langle y, z\rangle$ и открытому ключу e, а алгоритм B — ко второй компоненте и тому же открытому ключу e.

Следующая теорема утверждает, что при последовательном выполнении алгоритмов $A(x, e)$ и $B(x, e)$ разглашение "складывается". Точнее, разглашается только пара, состоящая из слов, разглашаемых каждым алгоритмом. Однако выполнено это не всегда. Достаточными условиями для этого является вхождение e в состав общеизвестной информации, и то, что с почти единичной вероятностью пара (c, x) попадает в область неразглашения алгоритма B.

Теорема 22 (о неразглашении при последовательном выполнении). *Пусть $g(x, e) = (e, k(x, e))$, где k — произвольная функция. Допустим, что алгоритм $A(x, e)$ разглашает только $f(x, e)$ при известном $g(x, e)$ на области D_n, а алгоритм $B(x, e)$ разглашает только $h(x, e)$ при известном $g(x, e)$ на той же области D_n. Пусть также хотя бы одна из функций $f(x, e)$ или $h(x, e)$ полиномиально вычислима. Тогда алгоритм*

(A, B) разглашает только пару $(f(x, e), h(x, e))$ при известном $g(x, e)$ на области D_n.

То же самое верно и для слабого разглашения (для любого распределения вероятностей на входах x).

Доказательство. Зафиксируем симуляторы M, N для алгоритмов A, B и построим симулятор для алгоритма (A, B). Алгоритмы A, B получают на вход некоторое неизвестное x и некоторое неизвестное и случайно выбранное e и после этого по очереди беседуют с некоторой схемой C, которая прочитала $g(x, e)$. В результате получается протокол беседы $c((A, B)(x, e), C(g(x, e))$, который является случайной величиной с параметром x.

Новый симулятор должен, получив схему C и подсказку об x, e в виде пары $f(x, e), h(x, e)$, сгенерировать случайную величину, вычислительно неотличимую от случайной величины $c((A, B)(x, e), C(g(x, e))$ при известном $g(x, e)$.

Будем в дальнейшем, через C^w мы обозначаем схему, полученную из C фиксацией первых $|w| + 1$ входов равными слову $w\Diamond$. Совместно распределённые случайные величины $f(x, e), h(x, e), g(x, e)$ будем кратко обозначать через f, h, g.

Процесс общения алгоритма $(A, B)(x, e)$ со схемой C устроен так. Сначала схема C прочитывает g, превращаясь в схему C^g. Затем эта схема беседует с алгоритмом $A(x, e)$. Обозначим через u протокол этого общения. Затем с алгоритмом B беседует схема, в которой следующие $|u| + 1$ входов (после первых $|g|+1$) зафиксированы равными $u\Diamond$. Будем обозначать эту схему через C^{gu}. Таким образом, протокол общения алгоритма $(A, B)(x, e)$ со схемой C имеет вид $uD(C^{gu})$, где u — протокол общения алгоритма $A(x, e)$ и схемы C^g, а $D(C^{gu})$ — протокол общения алгоритма $B(x, e)$ и схемы C^{gu}. Обе случайные величины $u, D(C^{gu})$, разумеется, зависят от x.

Нам нужно, по f, h и C, сгенерировать некоторую случайную величину, вычислительно неотличимую от $uD(C^{gu})$ при известном g. Чтобы сгенерировать такую случайную величину, запускаем данный нам симулятор M на входах f и C. Он выдаст некоторую последовательность сообщений $\tilde{u}$. После этого запустим симулятор N на входе h и $C^{*\tilde{u}}$ (так обозначена схема, которая на входе $y_1\Diamond y_2\Diamond \dots y_k\Diamond\#\# \dots \#$ выдаёт тот же результат, что и схема C на входе $y_1\Diamond\tilde{u}y_2\Diamond \dots y_k\Diamond\#\# \dots \#$; последовательность $\tilde{u}$ вставляется после первого разделителя). Он нам выдаст

последовательность, которую мы обозначим через $\tilde{D}(C^{g\tilde{u}})$ и про которую известно, что она вычислительно неотличима от протокола общения алгоритма $B(x,e)$ и схемы $C^{g\tilde{u}}$ при известном g. Выдаём в качестве результата $\tilde{u}\tilde{D}(C^{g\tilde{u}})$.

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

$$\alpha = \tilde{u}\tilde{D}(C^{g\tilde{u}}) \quad \text{и} \quad \gamma = uD(C^{gu}),$$

полученных фиксацией первого входа алгоритмов A, B как x_n. В силу транзитивности отношения неотличимости, достаточно доказать, что они обе неотличимы (при известном g) от какой-нибудь третьей случайной величины. Какую величину взять в качестве третьей, зависит от того, какая из двух функций f, h является полиномиально вычислимой функцией от g, x.

Сначала рассмотрим случай, когда таковой является h. Тогда в качестве третьей рассмотрим случайную величину

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

Неотличимость β и α доказывается так. Рассмотрим преобразование

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

Поскольку h есть полиномиально вычислимая функция, x фиксировано, а e входит в состав g, это преобразование можно вычислить вероятностной схемой полиномиального размера: эта схема вычисляет h и применяет симулятор N к h и схеме C^{*v}. В применении к случайной величине gu это преобразование даст случайную величину $g\beta$, а в применении к $g\tilde{u}$ оно даст $g\alpha$. Поскольку исходные случайные величины gu и $g\tilde{u}$ неотличимы, то таковы и $g\alpha$ и $g\beta$.

Осталось доказать неотличимость γ и β. Нам достаточно доказать их неотличимость при любой фиксации случайных битов алгоритма A. После этой фиксации случайная величина u становится функцией от e и g (и, поскольку e входит в состав g, только от g), вычислимой схемой полиномиального размера. Поэтому нам достаточно доказать (вычислительную) неотличимость случайных величин

$$gD(C^{gu}) \text{ и } g\tilde{D}(C^{gu}). \tag{8.1}$$

Поскольку по g можно вычислить u схемой полиномиального размера, существует последовательность схем F_n полиномиального размера, для которой схема F^g реализует ту же функцию, что и схема C^{gu}:

$$C^{gu} \equiv F^g.$$

По условию, для последовательности схем F_n случайные величины

$$gD(F^g) \text{ и } g\tilde{D}(F^g). \tag{8.2}$$

вычислительно неотличимы. Из эквивалентности схем следует, что их диалоги с алгоритмом $B(x, e)$ будут одинаковы и результаты применения к ним симулятора будут одинаковы (здесь нам важно, что симулятор использует схему, как чёрный ящик). Поэтому случайные величины (8.1) совпадают со случайными величинами (8.2). Поэтому из вычислительной неотличимости (8.2) следует вычислительная неотличимость (8.1).

Во втором случае, когда f является полиномиально вычислимой функцией от g, x, в качестве третьей рассмотрим случайную величину

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

Рассуждение в этом случае аналогично. Для неотличимости α и β (при известном g) нам нужно, чтобы f было полиномиально вычислимой функцией от g, x (это позволяет вычислить $\tilde{u}$ по g и случайным битам симулятора M схемой полиномиального размера). А для неотличимости α и β (при известном g) нам опять нужно, чтобы e было полиномиально вычислимой функцией от g, x (это позволяет вычислить $D(C^{gu})$ по g и u вероятностной схемой полиномиального размера).

Для слабого разглашения теорема доказывается точно так же. □

Замечание 9. Утверждение останется верным (и доказательство не изменится), если e является функцией от x и общеизвестной информации $g(x, e)$, причем эта функция полиномиально вычислима, и хотя бы одна из функций $f(x, e)$ или $h(x, e)$ является полиномиально вычислимой функцией от пары $(g(x, e), x)$.[2]

[2]Напомним, что мы не предполагаем, функции $f(x, e)$ или $h(x, e)$ полиномиально вычислимы. Поэтому полиномиальная вычислимость $f(x, e)$ и $h(x, e)$ по $(g(x, e), x)$ не следует из полиномиальной вычислимости e по $(g(x, e), x)$.

Замечание 10. Теорема о неразглашении при последовательном выполнении сохраняется и в ситуации, когда алгоритму B дополнительно даётся на вход протокол общения c алгоритма A и использованные им случайные биты r. В этом случае её можно сформулировать таким образом. *Пусть* $g(x,e) = (e, k(x,e))$, *где* k — *произвольная функция. Допустим, что алгоритм* $A(x,e)$ *разглашает только* $f(x,e)$ *при известном* $g(x,e)$ *на области* D_n, *а алгоритм* $B(c,r,x,e)$ *разглашает только* $h(x,e)$ *при известном* $g(x,e)$ *на области* E_n. *Пусть также хотя бы одна из функций* $f(x,e)$ *или* $h(x,e)$ *полиномиально вычислима. Пусть наконец, для всех* $x \in D_n$ *для протокола общения с алгоритма* A *с любой схемой* C_n *полиномиального размера и его случайных битов* r *с почти единичной вероятностью выполнено* $(c,r,x) \in E_n$. *Тогда алгоритм* (A,B) *разглашает только пару* $(f(x,e), h(x,e))$ *при известном* $g(x,e)$ *на области* D_n. *То же самое верно и для слабого разглашения (для любого распределения вероятностей на входах* x*).*

Доказательство этого утверждения получается повторением доказательства теоремы о неразглашении при последовательном выполнении.

Теорема о неразглашении при последовательном выполнении алгоритмов верна и произвольного полиномиального числа алгоритмов. Для её аккуратной формулировки, будем считать, что имеется один алгоритм A, который применяется $k = \mathrm{poly}(n)$ 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 22. Итак пусть дан вероятностный интерактивный алгоритм A, получающий на вход параметр безопасности n, строку x, число $i \le k = \mathrm{poly}(n)$ и случайную величину e с распределением зависящем от n. Выполним k раз алгоритм A последовательно на входах $(x,1,e)$, $\ldots$, (x,k,e) (строки x, e во всех запусках одни и те же). Будем обозначать полученный алгоритм через $A^{k(n)}$.

Теорема 23 (о неразглашении при последовательном применении полиномиального числа алгоритмов). *Пусть* $g(x,e) = (e, k(x,e))$, *где* k — *произвольная функция, а* $f(x,e)$ *некоторая полиномиально вычислимая функция. Если алгоритм* A *разглашает только* $f(x,e)$ *при известном* $g(x,e)$, *то алгоритм* $A^{k(n)}$ *также разглашает только* $f(x,e)$ *при известном* $g(x,e)$.

Аналогичное утверждение выполнено для слабого разглашения.

Доказательство. Обозначим через M симулятор для алгоритма A и построим симулятор для алгоритма A^k. Новый симулятор должен, получив на вход f и некоторую схему C полиномиального размера, породить случайную величину, при известном g вычислительно неотличимую от протокола общения алгоритма $A^k(x,e)$ и схемы C^g.

Мы будем использовать обозначения C^v и C^{*v}, введённые в доказательстве теоремы 22. Будем обозначать через $D_i(C,e)$ протокол общения алгоритма $A(x,i,e)$ со схемой $C^{g(x,e)}$. Будем также обозначать через $\tilde{D}_i(C,e)$, результат работы симулятора M на входах $f(x,e)$ и C.

Зафиксируем произвольную схему C (точнее последовательность схем C_n полиномиального размера). Протокол общения γ алгоритмов $A^k(x,e)$ и C^g можно получить последовательным применением к пустой строке для $j=1,\ldots,k$ вероятностных операторов

$$u \mapsto uD_j(C^{gu},e). \tag{8.3}$$

Поэтому естественно предположить, что если вместо (8.3) применить к пустой строке "близкие" операторы

$$u \mapsto u\tilde{D}_j(C^{gu},e), \tag{8.4}$$

то полученная случайная величина α будет вычислительно неотличима от γ при известном g. Заметим, что случайная величина α может быть порождена только по $f(x,e)$.

Зафиксируем произвольную последовательность x_n слов полиномиальной длины для алгоритма A^k и докажем, что α в самом деле неотличима от γ при известном $g(x_n,e)$. Для этого рассмотрим гибридные случайные величины $\beta_0,\ldots,\beta_k$. Случайная величина β_i определяется как результат применения к пустой строке последовательно операторов (8.3) для $j=1,\ldots,i$ и затем операторов (8.4) для $j=i+1,\ldots,k$. В частности, β_0 это то же самое, что α, а β_k — то же самое, что γ. Поэтому нам достаточно доказать, что β_{i-1} вычислительно неотличима от β_i при известном g (точнее, это нужно доказать для любой последовательности индексов i_n, в дальнейшем мы считаем, что такая последовательность зафиксирована).

Обозначим через δ случайную величину, полученную применением к пустой строке операторов (8.3) для $j=1,\ldots,i-1$, а через $F(u,e)$ обозначим случайную величину, полученную из строки u путем применения операторов (8.4) для $j=i+1,\ldots,k$. В этих обозначениях

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

Поскольку оператор $F(u,e)$ можно вычислить вероятностной схемой полиномиального размера по $x_n, f(x_n,e)$ и u (напомним, что x_n фиксировано, и его можно запаять в схему), нам достаточно доказать неотличимость $\delta D_i(C^{g\delta},e)$ и $\delta\tilde{D}_i(C^{g\delta},e)$ при известном g. Слово δ зависит от e и случайных битов, использованных алгоритмом A в первых его $i-1$ запусках. Нам достаточно доказать неотличимость $\delta D_i(C^{g\delta},e)$ и $\delta\tilde{D}_i(C^{g\delta},e)$ при известном g для любой фиксации этих битов. После такой фиксации δ становится функцией от e, а значит и от $g(e,x)$, вычислимой схемой полиномиального размера. Поэтому достаточно доказать неотличимость $D_i(C^{g\delta},e)$ и $\tilde{D}_i(C^{g\delta},e)$ (при известном g). 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

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

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

Следующая задача показывает, что в теореме 22 важно, что известная информация включает в себя e.

Задача 48. Пусть e случайный бит, выбранный с равномерным распределением. Пусть алгоритм A выдает на выход $x_1 \oplus e$ (где x_1 — первый бит входа x). Пусть алгоритм B выдаёт на выход $x_2 \oplus e$. Докажите, что оба алгоритма имеют нулевое разглашение, но их последовательное выполнение не имеет нулевого разглашения.

В следующей задаче приводится аналогичный пример для случая, когда вход x отсутствует и количество повторений пропорционально длине входа e.

Задача 49. Пусть $g_n : \{0,1\}^n \to \{0,1\}^n$ сильно односторонняя функция, а e равномерно распределено на её области определения $\{0,1\}^n$. Рассмотрим алгоритм, который (игнорируя вход x), выбирает случайное слово y длины n, сообщает пару $\langle y, y \odot e\rangle$ и останавливается. (а) Докажите, что этот алгоритм имеет нулевое разглашение при известном $g_n(e)$. (б) Докажите, что алгоритм B, который $2n$ раз запускает алгоритм A (с одним и тем же e и независимыми y-ами не имеет нулевого разглашения при известном $g_n(e)$.

Глава 9

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.

Протоколом идентификации (с открытым или 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 пару слов x, y, называемых *ключами*. Ключ x используется при идентификации клиентом в качестве пароля, а ключ y используется проверяющим алгоритмом. Алгоритмы P, V интерактивные и получают на вход, кроме параметра безопасности, ключи x, y, соответственно. Алгоритм $P(x)$ выполняется Доказывающим, а алгоритм $V(y)$ — Проверяющим. Доказывающий, зная x, должен убедить Проверяющего, который знает y, что он в самом деле знает x. Слово x будет называться *ключом для идентификации* а слово y — *ключом для проверки*.

Удобно использовать следующую метафору. Алгоритм $V(y)$ выполняется банкоматом, в который вставлена банковская карточка, микропроцессор которой выполняет алгоритм $P(x)$. Если алгоритм $V(y)$ после общения выдал 1, то он признал карточку подлинной, а иначе фальшивой. Первое требование на протоколы идентификации очевидно: честный клиент должен пройти идентификацию. Формально это выражается следующим условием *полноты:*.

> С вероятностью, приблизительно равной 1, выполнено $V^{P(x)}(y) =$ 1. Вероятность берется по распределению на ключах, порож-

даемому алгоритмом K, и по исходам случайных бросаний, выполняемых P и V.

Второе требование должно выражать то, что нечестный клиент не должен пройти идентификацию. Мы будем рассматривать три вида атаки на протоколы идентификации.

(1) Простая атака: противник, не зная ничего или зная только ключ y в протоколах с открытым ключом, пытается пройти идентификацию Требование *надежности* в этом случае формулируется так:

> *Для протоколов с закрытым ключом:* любой последовательности схем C_n полиномиального размера вероятность события $V^{C_n}(y) = 1$ пренебрежимо мала. Вероятность берется по случайному выбору y и по исходам случайных бросаний, выполняемых V. *Для протоколов с открытым ключом* в этом требовании нужно заменить схему C_n на схему $C_n(y)$ (схема C_n сначала читает y, а потом беседует с алгоритмом V_n).

(2) Атака с подслушиванием. Противник сначала подслушивает полиномиальное число протоколов общения алгоритмов $P(x)$ и $V(y)$, а потом хочет пройти идентификацию. Требование *устойчивости против атаки с подслушиванием* формулируется следующим образом:

> *Для протоколов с закрытым ключом:* пусть даны произвольная последовательность схем C_n полиномиального размера и произвольный полином $k(n)$. Рассмотрим следующую атаку: атакующая схема C_n, получив $k(n)$ протоколов общения алгоритмов $P(x)$ и $V(y)$ (в каждом из этих протоколов алгоритмы используют свежие случайные биты), пытается пройти идентификацию, беседуя с алгоритмом $V(y)$. Атака считается успешной, если алгоритм $V(y)$ 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. *Для протоколов с открытым ключом:* атакующей схеме на вход даётся еще и ключ y (а также $k(n)$ протоколов общения алгоритмов $P(x)$ и $V(y)$).

(3) Атака с фальшивым банкоматом. Представим себе, что противник изготовил фальшивый банкомат, который вместо правильной стратегии

проверяющего 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^*. Вероятность того, что противник успешно пройдет идентификацию должна быть пренебрежимо малой. Требование *устойчивости против атаки с фальшивым банкоматом* формулируется следующим образом:

> *Для протоколов с закрытым ключом:* пусть даны произвольная последовательность схем C_n полиномиального размера и произвольный полином $k(n)$. Рассмотрим следующую атаку: атакующая схема C_n, беседует с алгоритмом $P(x)^{k(n)}$ ($k(n)$-кратное последовательное выполнение независимых копий P с одним и тем же закрытым ключом). После этого схема пытается пройти идентификацию, беседуя с алгоритмом $V(y)$. Атака считается успешной, если алгоритм $V(y)$ 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. *Для протоколов с открытым ключом:* перед началом общения схема получает на вход ключ y.

Задача 50. Докажите, что из условия устойчивости против атаки с фальшивым банкоматом следует условие устойчивости против атаки с подслушиванием.

Определение 14. Протоколом идентификации называется тройка полиномиальных вероятностных алгоритмов (K, P, V), удовлетворяющая требованиям полноты и устойчивости (в зависимости от того, какое из шести рассмотренных нами требований надежности выбрать, получится определение одного из шести типов протоколов идентификации).

Задача 51. Докажите, что в протоколах, устойчивых против простой атаки без ограничения общности можно считать, что алгоритм P посылает алгоритму V ключ для идентификации и после этого останавливается. Поэтому определение протоколов, устойчивых против простой атаки, можно упростить, считая таким протоколом только алгоритм проверки V, который получает на вход оба ключа.

9.1 Протоколы идентификации с закрытым ключом

Протокол идентификации с закрытым ключом, устойчивый относительно простой атаки, можно построить без всяких теоретико-сложностных предположений: ключи x, y совпадают и выбираются случайно с равномерным распределением среди слов длины n. Алгоритм P посылает x, а алгоритм V проверяет, что присланное слово совпадает с y.

Однако этот протокол, очевидно неустойчив относительно атаки с подслушиванием. Протокол, устойчивый относительно атаки с подслушиванием, можно построить на основе семейства псевдослучайных функций. Причем, если это семейство удовлетворяет усиленному требованию надежности, то протокол будет устойчив и относительно атаки с фальшивым банкоматом.

Теорема 24. *Если существует семейство ПСФ $f_t^n : \{0,1\}^n \to \{0,1\}^n$, удовлетворяющее усиленному условию неотличимости (стр. 58), то существует протокол идентификации с закрытым ключом, устойчивый против атаки с фальшивым банкоматом. Если существует семейство ПСФ $f_t^n : \{0,1\}^n \to \{0,1\}^n$, удовлетворяющее обычному условию неотличимости (стр. 54), то существует протокол идентификации с закрытым ключом, устойчивый против атаки с подслушиванием.*

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

Требование полноты очевидно выполнено. Проверим требование надёжности для атаки с фальшивым банкоматом, предполагая, что семейство ПСФ удовлетворяет усиленному требованию неотличимости. Пусть даны 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 $f_t : \{0,1\}^n \to \{0,1\}^n$ как внешнюю процедуру и моделирует атаку схемы C_n. А именно, сначала она моделирует общение схемы C_n с алгоритмом $P^{k(n)}(x)$. При этом

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

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

Теперь проверим требование устойчивости против атаки с подслушиванием, предполагая, что семейство ПСФ удовлетворяет обычному требованию неотличимости. Пусть даны последовательность схем C_n, атакующая схему идентификации и полином $k(n)$. Схема C_n, получив на вход $k(n)$ значений ПСФ f_t на случайно выбранных входах $z_1, \ldots, z_k$, должна вычислить значение на еще одном случайно выбранном входе z. Пусть p обозначает вероятность успеха атаки. Тогда можно зафиксировать различные слова $z_1, \ldots, z_k, z$ так, чтобы вероятность успеха была не меньше $p - k(k+1)2^{-n-1}$. Атаку схемы можно рассматривать, как тестирование на случайность значений $f_t(z_1), \ldots, f_t(z_k), f_t(z)$. Ясно, что случайно выбранные $k+1$ слов пройдут этот тест с вероятностью 2^{-n}. Поэтому и значения $f_t(z_1), \ldots, f_t(z_k), f_t(z)$ пройдут этот тест с вероятностью приблизительно 2^{-n}. Следовательно, p пренебрежимо мало. □

9.2 Протоколы идентификации с открытым ключом

Протоколы, устойчивые против простой атаки

Легко построить такой протокол на основе любой сильно односторонней функции. В самом деле, пусть $f_n\{0,1\}^{p(n)\to\{0,1\}^{q(n)}}$ — сильно односторон-

няя функция. В качестве ключей выбирается пара $(x, f(x))$, алгоритм P посылает x, алгоритм V применяет к присланному слову функцию f и сравнивает полученное значение с $f(x)$.

Задача 52. Проверьте, что этот протокол удовлетворяет требованиям полноты и устойчивости против простой атаки.

Говорят, что именно так устроена идентификация в UNIX системах. Однако этот протокол не устойчив против атаки с подслушиванием: противник, перехвативший сообщение, посланное P, может затем сам пройти идентификацию.

Протоколы, устойчивые против атаки с подслушиванием

Задача 53. Построить протокол идентификации с открытым ключом, удовлетворяющий условиям (а) и (б') на основе любой односторонней перестановки с секретом.

Задача 54. Построить протокол идентификации с открытым ключом, удовлетворяющий условиям (а) и (б') на основе любого протокола шифрования с открытым ключом.

Протоколы из обеих задач могут быть не устойчивы против атаки с фальшивым банкоматом. В следующем разделе мы построим такой протокол на основе любой односторонней функции.

Задача 55. Объясните, почему для протокола, основанного на односторонней функции с секретом, условие (б") может не выполняться.

Протоколы, устойчивые против атаки с фальшивым банкоматом

Сначала мы построим такой протокол идентификации на основе функции Рабина, RSA или дискретной экспоненты (в предположении их необратимости). В этом протоколе условие (б") будет следовать из условия (б) и того, что алгоритм $P(x)$ имеет нулевое разглашение при известном y в слабом смысле (ключ y будет функцией от ключа x — только при таком условие наше определение неразглашения имеет смысл).

Лемма. *Пусть ключ y является функцией от ключа x. Тогда условие (б") следует из условия (б) и того, что, что алгоритм $P(x)$ имеет нулевое разглашение при известном y в слабом смысле.*

Доказательство. Пусть C_n — произвольная последовательность схем полиномиального размера в условии (б"). По условию (в) и теореме 23 алгоритм P^k имеет нулевое разглашение при известном y. Обозначим через M симулятор для алгоритма P^k. Применим его к открытому ключу y с оракулом C^y. Он выдаст случайную величину $\tilde{u}$, вычислительно неотличимую от записи общения u схемы C^g и алгоритма $P^k(x)$ при известном y. Рассмотрим теперь следующую попытку их отличить. Получив на вход $\langle v, y\rangle$, запускаем беседу стратегий $V(y)$ и C^{yv} и выдаем в качестве результата ответ V. Этот отличитель задается схемой полиномиального размера. Вероятность того, что он выдаст 1 на входе $\langle \tilde{u}, y\rangle$ пренебрежимо мала по свойству (б). Из неотличимости $\langle u, y\rangle$ и $\langle \tilde{u}, y\rangle$ следует, что и вероятность того, что он на входе $\langle u, y\rangle$ выдаст 1, также пренебрежимо мала. Это и надо было доказать. □

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 25. *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. В качестве закрытого ключа выдаётся пара $\langle m, x\rangle$, а в качестве открытого ключа — пара $\langle m, x^2 \bmod m\rangle$.

Основная идея протокола в том, чтобы Доказывающий доказал знание x, не разгласив его. Для этого Доказывающий указывет два таких числа z_0, z_1, что зная корень из обоих чисел можно найти x, но при этом корень только из одного из них не сообщает никакой информации об x. Проверяющий просит Доказывающего извлечь корень из одного из них, но только одного. В качестве таких чисел, можно например взять

$z_0 = y^2$ и $z_1 = y^2x^2$, где y выбрано случайно среди обратимых вычетов. Тогда зная корни из обоих этих чисел, можно найти и корень из x^2, как их отношение. При этом корень из каждого из них есть случайный вычет, поэтому не даёт никакой информации об x.

Более конкретно, 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 α.
3) Алгоритм P проверяет, является ли присланное ему одним битом (если это не так, то в качестве α 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.

Нулевое разглашение (в слабом смысле) проверяется следующим образом. По теореме 23 нам достаточно доказать, что в ходе однократного выполнения пунктов 1)–3) разглашается только m, x^2 при известном m, x^2. При этом мы будем предполагать, что пара (m, x) берётся из области определения функции Рабина. Это происходит с приблизительно единичной вероятностью при генерации ключей, поэтому мы докажем неразглашение в слабом смысле.

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

$$\langle y^2, F(y^2), yx^{F(y^2)} \rangle. \tag{9.1}$$

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 неотличимо

от старого, поэтому достаточно сгенерировать случайную величину, вычислительно неотличимую от (9.1) при равномерном распределении на y. На самом деле, мы сгенерируем величину, даже статистически неотличимую от неё.

Для этого заметим, что каждая тройка (9.1) 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 α (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 α независимы: при любом α случайная величина $u^2x^{-2\alpha}$ имеет равномерное распределение среди всех обратимых квадратичных вычетов. Поэтому и случайные величины $F(u^2x^{-2\alpha})$ и α 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 неотличимо от равномерного распределения на всех удачных тройках.

Осталось понять, что в беседе в самом деле появляется равномерное распределение на удачных тройках (точнее, статистически неотличимое от него). Тройка (9.1) 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 $1/\mathrm{poly}(n)$.

Итак, нам известно, что для бесконечно многих n вероятность события $V^{F(m,x^2)}(m, x^2) = 1$ больше $\varepsilon = 1/\mathrm{poly}(n)$. 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 α 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 α. 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 $p_1, p_2, \ldots, p_n$ не меньше $\varepsilon = 1/\mathrm{poly}(n)$. 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 протокол общения F и V и находим последовательность сообщений c, которая появится в беседе в первых $i-1$ раундах. При этом $\alpha_1, \ldots, \alpha_{i-1}$ (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 ε, 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 α 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 запаять i в схему-обратитель. □

Задача 56. Постройте аналогичные протоколы идентификации в предположении необратимости функции RSA и дискретной экспоненты.

В следующем разделе мы построим протокол идентификации с открытым ключом на основе любой сильно односторонней функции f. Закрытым ключом будет случайно выбранное слово x из области определения f, а открытым ключом будет $f(x)$. Идентификация будет заключаться в доказательстве с нулевым разглашением (при известном $f(x)$) знания некоторого прообраза $f(x)$.

9.3 Доказательства 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 *неразглашающим протоколом доказательства наличия свидетелей об* R, если выполнены следующие три условия

Полнота: 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. Если вероятность этого события в точности равна 1, то мы будем говорить, что выполнено условие *сильной полноты*

Корректность: Для любой последовательности слов y_n полиномиальной длины и для любой последовательности схем C_n полиномиального размера обозначим через $p_n(C_n, y_n)$ вероятность того, что $V(y_n)$, общаясь с C_n, выдаст 1. Тогда для всех n, для которых y_n не имеет свидетеля, $p_n(C_n, y_n)$ пренебрежимо мало.

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

Нам понадобится и более сильное условие корректности, в котором требуется, чтобы из схемы C_n можно было извлечь свидетеля всякий раз, когда она убеждает алгоритм V выдать единицу.

Сильная корректность: Пусть дана любая последовательности схем C_n полиномиального размера. Должен существовать полиномиальный вероятностный алгоритм M такой, что для любой последовательности C_n для всех n и всех последовательностей слов y_n полиномиальной длины алгоритм M, получив на вход y_n и схему C_n (которую он может использовать, как чёрный ящик), выдает на вход случайную величину (диалог общения $V(y_n)$ с C_n, x_n), вероятность того, что x_n является свидетелем для y_n при условии, что $V(y_n)^{C_n} = 1$, приблизительно равна 1

Если для данной пары (P, V) выполняются условия полноты, неразглашения и сильной корректности, та такую пару мы будем называть *неразглашающим протоколом доказательства знания свидетеля.*

Теорема 26. *Если существует протокол привязки, то для любого R существует неразглашающий протокол доказательства знания свидетелей об R. При этом если протокол привязки удовлетворяет условию сильной полноты, то и неразглашающий протокол доказательства знания удовлетворяет условию сильной полноты.*

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

Теорема 27. *Если существует сильно односторонняя функция, то существует и протокол идентификации с открытым ключом.*

Доказательство. Сначала заметим, что из условия следует существование протокола привязки (теорема 20), а значит (теорема 26) и существование протокола доказательства знания с нулевым разглашением для любого полиномиально разрешимого отношения R.

Пусть дана произвольная сильно односторонняя функция $f_n : \{0,1\}^{p(n)} \to \{0,1\}^{q(n)}$. 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 26 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 полноты для протокола доказательства знания эквивалентно условию полноты для построенного протокола идентификации.

Неразглашение протокола доказательства знания влечет неразглашение 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 для исходного протокола.

Докажем, что условие корректности для протокола доказательства знания влечет надежность против простой атаки для построенного протокола идентификации. Зафиксируем произвольную последовательность схем C_n полиномиального размера, для которой нужно установить, что вероятность события

$$V^{C(f(x))}(f(x)) = 1 \tag{9.2}$$

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 $p(a)/\mathrm{poly}(n)$. Усредняя по a, получаем, что вероятность события "M^C обращает случайное $f(x)$ " не меньше, чем вероятность события (9.2), делённая на $\mathrm{poly}(n)$. Поскольку f необратима, вероятность первого события пренебрежимо мала, а значит такова же и вероятность события (9.2). □

9.4 Доказательство теоремы 26

Сначала докажем теорему 26 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 $G = (\{1,\ldots,n\}, E)$ 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 28. *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 (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 ϕ корректной).

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

Свойство полноты (сильной полноты, если этим свойством обладал протокол привязки) очевидно выполнено.

Проверим свойство корректности. 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

$$T(c_{S^*,T}, k) \in \{F(c_{S^*,T}), \perp_S\}. \tag{9.3}$$

Поэтому мы можем считать, что в каждом раунде после стадии привязки имеется некоторая функция $F : \{1, \ldots, n\} \to \{1, 2, 3\}$ 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. Надо построить полиномиальный вероятностный алгоритм с оракулом C, который находит некоторую корректную раскраску в вероятностью примерно p. Этот алгоритм моделирует общение V и C. При этом в конце каждого раунда, прежде чем перейти к следующему раунду, он дополнительно просит схему C по очереди раскрыть ключи для концов *всех* ребер. 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. Если раскраска не найдена, то, не меняя i, повторяем то же самое еще один раз, начиная со стадии привязки. Если опять не повезет, повторяем еще раз, всего n раз. Только после этого мы переходим к следующему раунду.

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 $p_1, p_2, \ldots, p_{2n|E|}$, 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 $p_1, p_2, \ldots, p_{2n|E|}$ 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 с точностью до пренебрежимо малой величины. А для этого достаточно доказать, что вероятность этого при условии, что все $i-1$ предыдущих раундов завершились успешно, равна 1 с точностью до пренебрежимо малой величины.

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 графа (имеющих положительную степень, остальные вершины можно раскрасить как угодно). Поскольку мы повторяем для данного i эту процедуру n раз, вероятность условная вероятность невезения не больше 2^{-n}.

Осталось проверить неразглашение. 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 ϕ некорректна, то c_{P,V^*} состоит из пустого слова и может быть порождено по известной нам информации, что $R(\phi, G)$ ложно. Пусть раскраска ϕ корректна. Нам нужно придумать алгоритм с оракулом V^* порождающий случайную величину, вычислительно неотличимую от случайной величины c_{P,V^*}.

Пусть β произвольная, возможно некорректная, раскраска графа. Рассмотрим следующую стратегию Доказывающего, которую мы обозначим через P_β: запускаем алгоритмы S_β 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_α для случайной корректной раскраски α вида $\pi\phi$.

Алгоритм порождения случайной величины, неотличимой от c_{P,V^*}, таков. Выберем случайную раскраску β (все раскраски, корректные и некорректные выбираются с одинаковой вероятностью) и смоделируем общение P_β с V^*. Скажем, что нам *повезло*, если V^* после выполнения алгоритма S_β 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 β. 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_β, 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_β может быть заменен на композицию алгоритмов

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

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

$$V^*(c_{S_{\beta abuv}, V^*}) = (u, v) \tag{9.4}$$

и $\beta(u) = a, \beta(v) = b$ независимы, вероятность их пересечения равна произведению их вероятностей, то есть, одной девятой вероятности события (9.4). Теперь воспользуемся тем, что каждая из стратегий $S_{\beta(1)}$, $\ldots$, S_a, $\ldots$, S_b, $\ldots$, $S_{\beta(n)}$ имеет нулевое разглашение, а значит и их композиция тоже. Следовательно, существует полиномиальный алгоритм с оракулом V^*, генерирующий случайную величину θ, вычислительно неотличимую от протокола общения $S_{\beta abuv}$ и V^*. Поэтому для любых β, a, b, u, v вероятность события (9.4) 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 η. Ее значение равно c_{P_β, V^*} при условии, что в раскраске β концы ребра $V^*(c_{S_\beta, V^*})$ имеют различные цвета. То есть,

$$\Pr[\eta = d] \approx \Pr[c_{P_\beta, V^*} = d \mid (\exists u, v)\ V^*(c_{S_\beta, V^*}) = (u, v),\ \beta(u) \neq \beta(v)] \tag{9.5}$$

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

$$A(c_{P_\alpha, V^*}) = 1 \tag{9.6}$$

примерно равна вероятности события $A(\eta) = 1$. Поскольку вероятность условия в (9.5) приблизительно равна $2/3$, вероятности события $A(\eta) = 1$ примерно равна $3/2$ вероятности события

$$A(c_{P_\beta, V^*}) = 1,\ \beta(u) \neq \beta(v). \tag{9.7}$$

Представим каждое из событий (9.6) и (9.7) как объединение полиномиального числа непересекающихся событий. А именно, для каждого ребра (u, v) и каждой пары различных цветов a, b рассмотрим пересечение события (9.6) и события "Проверяющий послал ребро u, v и концы этого ребра имеют цвета a, b":

$$V^*(c_{P_\alpha, V^*}) = (u, v),\ \alpha(u) = a,\ \alpha(v) = b. \tag{9.8}$$

Аналогичным образом, для каждого ребра (u, v) и каждой пары различных цветов a, b рассмотрим пересечение события (9.7) с аналогичным событием:

$$V^*(c_{P_\beta, V^*}) = (u, v),\ \beta(u) = a,\ \beta(v) = b. \tag{9.9}$$

Нам достаточно доказать, что при любых (u, v) и $a \neq b$ вероятности событий (9.6) $\cap$ (9.8) и (9.7) $\cap$ (9.9) 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 α есть композиция корректной раскраски и некоторой перестановки цветов, а β является случайной раскраской. То есть они относятся как раз, как 3 к 2. А ещё заметим, что при подсчёте вероятности события (9.7) $\cap$ (9.9) 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 (9.6) $\cap$ (9.8) не изменится, если алгоритм P_α заменить на алгоритм $P_{\alpha abuv}$, который выполняет алгоритмы $S_{\alpha(1)}, \ldots, S_a, \ldots, S_b, \ldots, S_{\beta(n)}$ а после получения ребра выдает ключи k_u, k_v (независимо от того, какое ребро пошлет Проверяющий). Аналогичную замену можно произвести в событии (9.7) $\cap$ (9.9). После такой замены условие $\alpha(u) = a, \alpha(v) = b$, имеющееся в событии (9.6) $\cap$ (9.8) становится независимым от остальных двух условий. То же самое верно и для события (9.7) $\cap$ (9.9). Поэтому вероятность (9.6) $\cap$ (9.8) равна одной шестой вероятности события

$$A(c_{P_{\alpha abuv}, V^*}) = 1,\ V^*(c_{S_{\alpha abuv}, V^*}) = (u, v), \tag{9.10}$$

а (9.7) $\cap$ (9.9) — одной девятой вероятности события

$$A(c_{P_{\beta abuv}, V^*}) = 1,\ V^*(c_{S_{\beta abuv}, V^*}) = (u, v). \tag{9.11}$$

Следовательно, нам достаточно доказать, что вероятности событий (9.10) и (9.11) примерно равны.

В чём разница между событиями (9.10) и (9.11)? Разница только в способе выбора раскраски: раскраски α и β выбирается по различным распределениям. Поэтому нам достаточно доказать, что вероятность события (9.11) при фиксированной раскраске β мало зависит от β. Точнее, нам достаточно доказать, что при фиксированной β вероятность события (9.11) равна (с точностью до пренебрежимо малой величины) некоторому числу p_{abuv}, которое не зависит от β. Это следует из того, что алгоритм S имеет нулевое разглашение. Отсюда и из теоремы 23 следует, что алгоритм $S_{\beta abuv}$, получающий на вход β, a, b, u, v, имеет нулевое разглашение. Однако для алгоритма $P_{\beta abuv}$, получающего на вход β, 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 $A_1, \ldots, A_{n+2}$. Все алгоритмы получают на вход β, a, b, u, v и пару 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 его случайных битов, то есть, $k_u = ra$, $k_v = sb$. Таким образом, мы представили $P_{\beta abuv}$, как последовательность независимо работающих алгоритмов, и можем применить к ним теорему 23. Поскольку для всех i 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 верно для любых r, s, поэтому верно и для случайно выбранных r, s.

Итак, рассмотрим тест, который получив на вход протокол 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 β не зависит, то и вероятность события (9.11) 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 $f(y)$. Применив 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 протокола общения $P(g(x,y), f(y))$ и любой схемы полиномиального размера. А если $R(x,y)$ ложно, то просто выдаём пустую последовательность сообщений. □

Замечание 11. Заметим, что использованный в доказательстве теоремы алгоритм извлечения раскраски не просто для каждого графа G находит раскраску G с примерно той же вероятностью, с какой схема C убеждает V выдать 1. Алгоритм моделирует общение алгоритмов Доказывающего и Проверяющего, причем, всякий раз (за исключением пренебрежи-

мо малой вероятности), когда Проверяющий выдаёт 1, алгоритм выдает раскраску графа.

Глава 10

Семейства хэш-функций

Говоря неформально, хэш-функцией называется функция $h : \{0,1\}^* \to \{0,1\}^k$, 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 обнаружимыми коллизиями (СТОК) и слабую — универсальные семейства односторонних хэш-функций (УСОХ).

10.1 Семейства хэш-функций с трудно обнаружимыми коллизиями

Будем рассматривать полиномиально вычислимые семейства функций вида $h^n : \{0,1\}^l \times \{0,1\}^* \to \{0,1\}^k$, 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 каждых фиксированных n и t из $\{0,1\}^l$ получаем функцию $x \mapsto h^n(t, x)$, которую мы будем обозначать через h^n_t. Число n называется параметром безопасности, а строка t — идентификатором хэш-функции. Пусть еще имеется доступная случайная величина α_n, принимающая значения в $\{0,1\}^{l(n)}$ (то есть, распределение вероятности на идентификаторах хэш-функций). Пару, состоящую из h^n и α_n, называют семейством хэш-функций с трудно об-

наружимыми коллизиями (СТОК), если для любой последовательности схем C_n размера $\mathrm{poly}(n)$ вероятность события "$C_n(\alpha_n)$ есть коллизия для $h^n_{\alpha_n}$" 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^0_t, f^1_t одного и того же множества D_t называется семейство с трудно обнаружимыми зацеплениями (СТОЗ), если по t трудно найти зацепление функций f^0_t, f^1_t.

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

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

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

Теорема 29. *Если функция Рабина необратима, то существует СТОЗ.*

Доказательство. Случайная величина α_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 $f^0_{m,x}(y)$ определена как

$xy^2 \bmod m$, а $f^1_{m,x}(y) = y^2 \bmod m$ (вторая функция от x не зависит). Пары 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. □

Задача 57. 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^0_t, f^1_t множества D_t с трудно обнаружимыми зацеплениями, причем распределение на t дается доступной случайной величиной α. Сопоставим каждому слову $x = x_1 x_2 \dots x_m$ его префиксный код $\hat{x} = x_1 x_1 x_2 x_2 \dots x_m x_m 01$. 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

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

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

Теорема 30. *Построенное семейство функций является семейством с трудно обнаружимыми коллизиями.*

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

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

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

$\ldots, \hat{x}_{i-1} = \hat{z}_{i-1}$, мы можем сократить это равенство слева на композицию функций $f_t^{\hat{x}_1}, \ldots, f_t^{\hat{x}_{i-1}}$:

$$f_t^{\hat{x}_i}(f_t^{\hat{x}_{i+1}}(\ldots f_t^{\hat{x}_l}(\beta_t)\ldots)) = f_t^{\hat{z}_i}(f_t^{\hat{z}_{i+1}}(\ldots f_t^{\hat{z}_m}(\beta_t)\ldots)).$$

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

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

10.2 Универсальные семейства односторонних хэш-функций

Полиномиально вычислимое отображение $h^n : \{0,1\}^{q(n)} \times \{0,1\}^{p(n)} \to \{0,1\}^n$ (здесь $p(n), q(n)$ — некоторые многочлены) называется универсальным семейством односторонних хэш-функций (УСОХ), если существует полиномиально моделируемое распределению μ_n на словах длины $q(n)$ со следующим свойством. Для любой последовательности схем C_n размера $\mathrm{poly}(n)$ 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 μ_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 31. *Если существуют частичная сильно односторонняя функция, то для любого полинома $p(n)$ существует полином $q(n)$ и УСОХ $h^n : \{0,1\}^{q(n)} \times \{0,1\}^{p(n)} \to \{0,1\}^n$.*

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

Теорема 32. *Если существуют сильно односторонняя перестановка $f_n : \{0,1\}^n \to \{0,1\}^n$, то для любого полинома $p(n)$ существует полином $q(n)$ и УСОХ $h^n : \{0,1\}^{q(n)} \times \{0,1\}^{p(n)} \to \{0,1\}^n$.*

Доказательство. Сначала построим УСОХ $h_s : \{0,1\}^{n+1} \to \{0,1\}^n$. 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 μ — равномерное на невырожденных матрицах.

Задача 58. Докажите, что равномерное распределение на невырожденных матрицах порядка $n \times (n+1)$ доступно.

Лемма. *Если перестановка $f_n : \{0,1\}^n \to \{0,1\}^n$ 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 59. 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 (количество невырожденных матриц, аннулирующих данный вектор, не зависит от этого вектора). С другой стороны все горизонтальные сечения множества S тоже равномощны (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. □

Пусть теперь дан произвольный многочлен $p(n) > n$ и УСОХ $h_t^n : \{0,1\}^{n+1} \to \{0,1\}^n$. Преобразуем его в семейство хэш-функций $g_{\tilde{t}}^n \{0,1\}^{p(n)} \to \{0,1\}^n$ следующим образом. Идентификатором $\tilde{t}$ новой хэш-функции будет строка из $m = k(n) - n$ идентификаторов $t_{k-1}, \ldots, t_n$ функций из исходного семейства, причем $g_t(x)$ есть композиция функций $g_{t_{k-1}}$, $\ldots$, t_{s_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 $g_{\tilde{t}}$ является УСОХ.*

Доказательство. Утверждение следует из следующего простого наблюдения. Пусть $x \neq y$ есть коллизия $g_{\tilde{t}}$, то есть, $g_{\tilde{t}}(x) = g_{\tilde{t}}(y)$. 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 коллизию одной функции f_{t_j} для некоторого $j = n, \ldots, k-1$.

Теперь допустим, что некоторая схема C_n полиномиального размера для бесконечно многих n атакует семейство $g_{\tilde{t}}$ с вероятностью успеха не менее $\varepsilon_n = 2/\mathrm{poly}(n)$ и при этом первый элемент коллизии равен x_n. Тогда для каждого из этих n можно найдется $j = n, \ldots, p(n)-1$, для которого с вероятностью не менее $\varepsilon_n/(p(n)-n)$ в указанном процессе будет найдена коллизия функции $h^j_{t_j}$. 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 $h_{t_{p(n)-1}}, \ldots, h_{t_{j+1}}$. 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 $h_{t_{p(n)-1}}, \ldots, h_{t_{j+1}}, h_{t_j}$ были выбраны случайно.

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

□

Глава 11

Протоколы электронной подписи

Протоколом электронной подписи называется тройка вероятностных полиномиальных алгоритмов (K, S, V), называемыми *алгоритмом генерации ключей*, *алгоритмом подписи* и *алгоритмом проверки*, соответственно. Все алгоритмы получают на вход параметр безопасности n в унарной записи, а алгоритмы подписи и проверки еще и несколько слов. Алгоритм K, получив на вход параметр безопасности, выдаёт на выход пару слов (x, y), называемых *ключами для подписи и её проверки*. Алгоритм подписи, кроме параметра безопасности, получает на вход еще пару слов, ключ для подписи d и подписываемое *сообщение* m, и выдает на выход слово, называемую *подписью*. Алгоритм V, кроме параметра безопасности, он получает на вход три слова, ключ для проверки y, сообщение m и подпись s, выдаёт на выход 0 или 1 (отвергает или принимает подпись).

Любой протокол электронной подписи должен позволить честному участнику подписывать любые сообщения полиномиальной (от параметра безопасности) длины. Это условие выражается в следующем требовании полноты.

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

Кроме этого, нужно потребовать, чтобы нечестный участник (то есть,

не знающий ключа x) не мог подписать никакого сообщения. Это требование, которое мы будем называть надежностью схемы подписи, существует в нескольких модификациях. Во-первых, протоколы делятся на протоколы с *закрытым ключом* и протоколы с *открытым ключом*. В протоколах с *закрытым ключом* мы считаем, что противник не знает ни одного из двух ключей, а в протоколах с открытым ключом мы считаем, что противник знает ключ для проверки. Во-вторых, протоколы можно делить по виду атаки, которую они способны выдержать. Мы ограничимся рассмотрением только одного вида атаки, называемой *адаптивной атакой с выбором сообщений*: противник (получив ключ для проверки в системах с открытым ключом) выбирает любое сообщение m_1, получает подлинную подпись s_1 под ним, затем выбирает сообщение m_2, получает подлинную подпись s_2 под ним и так далее некоторое полиномиальное от n количество раз. После этого противник должен выдать сообщение, которое отлично от всех выбранных ранее сообщений и подпись под ним. Если эта подпись признана подлинной, то атака считается успешной. Требование надежности состоит в том, что если алгоритм действия противника реализуем схемой, количество функциональных элементов в которой ограничено некоторым полиномом от параметра безопасности, то вероятность успеха атаки пренебрежимо мала.

Вот более детальная формулировка требования *надёжности*. Сначала сформулируем требование надёжности для протоколов с открытым ключом.

Надёжность для протоколов с открытым ключом: Для любой полинома $k(n)$ и любой последовательности схем C_n полиномиальной длины должно быть выполнено следующее. Для каждого значения параметра безопасности n и каждой пары ключей (x, y) определим две последовательности m_i и s_i рекурсивно:

$$\begin{aligned} m_1 &= C(y), \quad & s_1 &= S(x, m_1), \\ m_2 &= C(y \Diamond s_1), \quad & s_2 &= S(x, m_2), \\ & \dots & & \\ m_k &= C(y \Diamond s_1 \Diamond \dots \Diamond s_{k-1}), \quad & s_k &= S(x, m_k), \\ m_{k+1} &= C(y \Diamond s_1 \Diamond \dots \Diamond s_k), \quad & s_{k+1} &= C(x, s_1, \dots, s_k). \end{aligned}$$

Требуется, чтобы при случайном выборе пары ключей (x, y) вероятность события

$$m_{k+1} \notin \{m_1, \dots, m_k\}, \qquad V(y, m_{k+1}, s_{k+1}) = 1$$

была пренебрежимо малой. (В вероятностное пространство также входят случайные биты алгоритма подписи S.)

Требование надёжности для протоколов с закрытым ключом: формулируется совершенно аналогично. Для этого в этом определении везде надо заменить $C(y*)$ на $C(*)$ (схема не получает на вход ключа y).

Определение 15. Протоколом электронной подписи с открытым (закрытым) ключом называется тройка вероятностных полиномиальных алгоритмов (K, S, V), удовлетворяющая требованиям полноты и надёжности для схем с открытым (закрытым) ключом.

В чем отличие схем подписи от протоколов идентификации? 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. Для идентификации доказывающий подписывает случайное выбранное Проверяющим сообщение длины n.

Задача 60. Докажите, что этот протокол идентификации удовлетворяет требованиям полноты и надежности для протоколов идентификации (с открытым ключом, если таковой была исходная схема подписи, и с закрытым ключом, иначе).

11.1 Протоколы электронной подписи с закрытым ключом

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

Теорема 33. *Если существует семейство ПСФ $f_s^n : \{0,1\}^n \to \{0,1\}^n$, удовлетворяющее усиленному условию на стр. 58, то существует протокол идентификации с закрытым ключом.*

Доказательство. Пусть длина идентификатора s ПСФ f_e^n равна $l(n)$. Алгоритм генерации ключей выбирает с равномерным распределением случайную строку s длины $l(n)$ и выдаёт её в качестве обоих ключей:

$x = y = s$. Алгоритм V выбирает случайное слово z длины n и посылает его алгоритму S. Алгоритм S вычисляет $f_t(z)$ и посылает его V. На этом общение заканчивается и алгоритм V выдаёт 1, если присланная ему строка совпадает с $f_t(z)$, и выдаёт 0 иначе.

Условие полноты очевидно выполнено. Проверим условие надёжности. Пусть даны многочлен $k = k(n)$ и последовательность схем C_n, атакующая схему идентификации. Преобразуем схему C_n в схему D_n, тестирующую ПСФ. Схема D_n получает тестируемую функцию $f_t : \{0,1\}^n \to \{0,1\}^n$ в качестве чёрного ящика (= внешней процедуры) и моделирует атаку схемы C_n. А именно, сначала она моделирует общение схемы C_n с алгоритмом $S^k(x)$. При этом всякий раз, когда алгоритм $S(x)$ должен вычислить значение функции f_t на запрошенном аргументе z, схема D_n вызывает внешнюю процедуру для вычисления этого значения. После k повторений схема D_n моделирует общение C_n с $V(y)$. Для этого она выбирает теперь уже сама случайное z' длины n и подаёт его на вход схеме C_n вместе со всей информацией, доступной схеме к этому моменту. Схема C_n возвращает некоторое u'. Наконец, схема D_n в последний раз запрашивает значение внешней функции f_t на z'. Если полученное значение совпадает с u', она выдаёт 1, а иначе 0.

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

11.2 Протоколы электронной подписи с открытым ключом

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 против 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 против атаки, в которой длины всех сообщений $x_1, \ldots, x_k, x_{k+1}$ равны $p(n)$, называются протоколами подписи сообщений длины $p(n)$. Самым простыми будут одноразовые схемы подписи одного бита (то есть протоколы подписи сообщений длины 1).

11.3 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 σ. Затем схема $C = C_n$ еще раз применяется к паре e, s, где $s = S(d, \sigma)$ есть настоящая подпись под битом σ. Атака считается успешной, если выданная фальшивая подпись $s' = C(e, s)$ признана правильной подписью под противоположным битом $\overline{\sigma}$, то есть, $V(e, \overline{\sigma}, s') = 1$. Требуется, чтобы вероятность успеха атаки была пренебрежимо мала (для любой последовательности схем $\{C_n\}$ полиномиального от n размера.

Теорема 34. *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 σ, применяет f_n к s и принимает 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 $\varepsilon = 1/\mathrm{poly}(n)$. 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^σ, он применяет схему C к открытому ключу и x^σ и выдаёт поддельную подпись s'. Атака признаётся успешной, если $y^{\overline{\sigma}} = f(s')$.

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 σ. 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 σ, 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 σ, случайные величины σ, 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 σ не зависимо 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. □

Задача 61. 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 $C(f(x'), y, x')$. Если хотя бы одно из полученных слов будет прообразом y (мы можем это проверить полиномиальной схемой), выдадим это слово. Докажите, что вероятность успеха этого алгоритма также не меньше $\varepsilon/2$, причем в худшем случае она может быть примерно равна

$\varepsilon/2$.

11.4 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

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

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

Теорема 35. *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 $\varepsilon = 1/\mathrm{poly}(n)$ 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. □

Эта система подписи является в самом деле всего лишь одноразовой в следующем смысле. Имея подписи под сообщениями $00\ldots0$ и $11\ldots1$, можно изготовить подпись под любым сообщением.

11.5 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

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

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).

11.6 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 36. *Если существует СТОК* $h^n : \{0,1\}^{l(n)} \times \{0,1\}^* \to \{0,1\}^{k(n)}$ *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 α_n — случайная величина из определения СТОК $h^n : \{0,1\}^{l(n)} \times \{0,1\}^* \to \{0,1\}^{k(n)}$.

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_α. 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_α, 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_α, 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 $\varepsilon = 1/\mathrm{poly}(n)$. 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, α 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 ε. 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 α 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, α), 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 α, 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.

Здесь опять важно, что распределение на парах (e, d, α), 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).

11.7 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 семейство односторонних хэш-функций $h^n : \{0,1\}^{q(n)} \times \{0,1\}^{p(n)} \to \{0,1\}^n$, 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 $x = \sigma_1 \ldots \sigma_l$ устроена так. Подписывающей выбирает случайно и независимо новые пары ключей $\langle e_2, d_2\rangle, \ldots, \langle e_{l+1}, d_{l+1}\rangle$. 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 $s_1, \ldots, s_l$ полученную последовательность подписей. Общей подписью будет последовательность открытых ключей $e_2, \ldots, e_{l+1}$ и последовательность подписей $s_1, \ldots, s_l$. Алгоритм проверки подписи состоит в применении алгоритма V с ключами $e, e_2, \ldots, e_l$ к сообщениям (соответственно) $h_s(e_2)\sigma_1, \ldots, h_s(e_l)\sigma_{l-1}$, $h_s(e_{l+1})\sigma_l$ и подписям $s_1, \ldots, s_l$.

Теорема 37. *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

$$\begin{aligned} s_1 &= S(d, h_s(e_2)\sigma_1), \\ s_2 &= S(d_2, h_s(e_3)\sigma_2), \\ &\dots \\ s_l &= S(d_l, h_s(e_{l+1})\sigma_l). \end{aligned}$$

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

$$\begin{aligned} V(e, h_s(e'_2)\sigma'_1, s'_1) &= 1, \\ V(e_2, h_s(e'_3)\sigma'_2, s'_2) &= 1, \\ &\dots \\ V(e_{l'}, h_s(e_{l'+1})\sigma'_{l'}, s'_{l'}) &= 1. \end{aligned}$$

Допустим для бесконечно многих почти всех n вероятность этого события больше $\varepsilon = 1/\mathrm{poly}(n)$.

Лемма. *Если атака успешна, то найдется такое* $1 < j \le \min\{l, l'\}$, *что* $e_j \ne e'_j$, *но* $h_s(e_j) = h_s(e'_j)$, *или найдется такое* $1 \le j \le \min\{l, l'\}$, *что* $e_j = e'_j$, *но* $\sigma_j h_s(e_{j+1}) \ne \sigma'_j h_s(e_{j+1})$.

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.

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 σ_1 и σ'_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. □

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 $\sigma_1 \ldots \sigma_l$, генерируя новые пары ключей $e_2, d_2, \ldots, e_l, d_l$ (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 подпишем его префиксный код $\hat{y}$ (и при проверке будем проверять, подписано ли $\hat{y}$). Если два слова вида вида $\hat{y}$ различны, то ни одно из них не является началом другого. Поэтому модифицированная система подписи удовлетворяет требованию (б) в его исходном виде. □

11.8 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 $f_s^n : \{0,1\}^* \to \{0,1\}^{l(n)}$ — 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

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_α — для всех бинарных слов α длины n — любой из них использовать для подписи данного сообщения. При этом в состав подписи нужно включить саму последовательность α и ключи e_β для всех непустых начал слова α, а также подписи под ними. Строку α, 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_α (для всех строк α длины n) будет использован не более одного раза. (А можно идентификатор выбирать детерминированным образом, исходя из сообщения, — этот способ изложен в виде задачи, приведённой после доказательства.)

Однако нам нужно еще, чтобы и все остальные ключи d_β, 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 функцию f_s. А именно, запустим алгоритм K с параметром безопасности n, а в качестве случайных битов дадим ему биты слова $f_s(\alpha)$, где α — любая непустая строка длины не более n. Полученную пару ключей и будем использовать в качестве вышеупомянутых ключей (d_α, e_α). Обозначение (e_Λ, d_Λ), где Λ обозначает пустое слово, будем использовать для исходной пары ключей. Если длина α меньше n, то ключ d_α 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_α дважды — для подписи $e_{\alpha 0}$ и для подписи $e_{\alpha 1}$.

Итак, сообщение x подписываем так. Выбираем случайным образом идентификатор α (бинарную строку длины n). Затем алгоритмом S с ключом d_α 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 α и сам идентификатор α.

Проверяется новая подпись так: пусть нам дана последовательность, состоящая из идентификатора α, $n+1$-ой подписи $s_0, \ldots, s_n$ для старой схемы и n пар ключей, которые мы будем обозначать через $e'_{\gamma 0}e'_{\gamma 1}$, где γ пробегает все собственные начала α. Мы используем штрихи, поскольку не предполагаем заранее, что подпись честная. Таким образом, для любого непустого начала β последовательности α у нас имеется ключ e'_β. Также положим e'_Λ равным данному нам открытому ключу. Затем алгоритмом V с ключом e'_α проверяем, что s_n является подлинной подписью под данным нам сообщением, а так же, что для всех собственных начал γ строки α слово $s_{|\gamma|}$ является подлинной подписью под $e'_{\gamma 0}e'_{\gamma 1}$ с ключом e'_γ. Если все проверки закончились успешно, то подпись признается, а иначе отвергается.

Теорема 38. *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 $\varepsilon = 1/\mathrm{poly}(n)$. Заменим в новой схеме ПСФ f_s на случайную функцию $g : \{0,1\}^* \to \{0,1\}^{l(n)}$. 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

считать, что все ключи (e_β, d_β) 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_Λ, d_Λ). Затем передаем e_Λ 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_α, d_α), которая нам понадобилась, генерируется независимо от предыдущих. Но при этом, если при подписывании некоторого сообщения мы сгенерировали один раз пару ключей (e_α, d_α), а затем позже нам понадобилась еще раз пара ключей (e_α, d_α), 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 α *использованной* (в моделировании), если мы в ходе него сгенерировали пару ключей (e_α, d_α).

Допустим, что моделирование закончилось успехом. Добавим фальшивое сообщение x в конец фальшивой подписи $\tilde{s}$. Изменённая таким образом $\tilde{s}$ состоит некоторого идентификатора β (двоичной строки длины n) и $n+1$ пары (подпись, сообщение) для исходной системы подписи. В последней, $n+1$-ая паре 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'_α, где α обозначает соответствующее начало β или его брата. Положим также e'_Λ равным ключу e_Λ. 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 $e'_{\beta[1\ldots j]}$. Тогда идентификатор β, с которого начинается фальшивое сообщение, имеет начало $\alpha = \beta[1\ldots j]$ такое, что

- α является использованным,
- $e'_\alpha = e_\alpha$,
- с помощью ключа d_α 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 α равным пустому слову. Иначе, мы с ключом d_Λ подписали слово e_0e_1. Если слово $e'_0e'_1$ не совпадает со словом e_0e_1, то опять же можно положить α равным пустому слову. Иначе положим α равным первому биту β. 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 α фальшивого идентификатора β, которое использовано и для которого $e_\alpha = e'_\alpha$. Если ни один из использованных нами идентификаторов не начинается на α, то мы с ключом d_α ничего не подписывали и тем самым α годится. Иначе α является собственным началом как слова β, так и некоторого использованного нами идентификатора. Поэтому мы с ключом d_α подписали слово $e_{\alpha0}e_{\alpha1}$, а схема — слово $e'_{\alpha0}e'_{\alpha1}$. Если эти слова различны, то опять же текущее α годится. Иначе приписываем к α еще один бит β, 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 α, ключ e_α 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_α и второй компонент в ней не был подписан нами в ходе моделирования атаки. Это означает, что схема C 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

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 случайным образом число i от 1 до $p(n)$ (с равномерным распределением). 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 α, с вероятностью $1/p(n)$ число i окажется равным номеру особого ключа e_α в порядке генерации пар ключей. Поэтом с вероятностью $\varepsilon/2p(n)$ нам повезет и особый ключ окажется равным данному нам для атаки ключу. $\square$

Задача 62. Докажите, что для теоремы 38 достаточно требования обычной неотличимости семейства ПСФ.

Задача 63. Будем в многоразовой схеме подписи в качестве идентификатора сообщения использовать его префиксный код. Докажите, что теорема 38 останется верной. Докажите, что если в качестве идентификатора использовать сообщения использовать само сообщение, то теорема 38 перестанет быть верной.

Задача 64. 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 одноразовой схемы).

Глава 12

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.

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 A напечатает $g(x, y)$. Если вероятность этого события в точности равна 1, то мы будем говорить, что выполнено условие *сильной полноты*.

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).

Неразглашение информации для полу-честных игроков. Неразглашение информации алгоритмом B: существует полиномиальный вероят-

ностный 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 протокола общения между A с входом x_n и B с входом y_n. Неразглашение информации алгоритмом A: 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 протокол общения.

Неразглашение информации для нечестных игроков. Неразглашение информации алгоритмом B: 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 A_n^* полиномиального размера алгоритм S на входе y_n находит некоторую строку x_n, и после того, как ему сообщат $f(x_n, y_n)$, выдает на выход случайную величину, вычислительно неотличимую от протокола общения между A_n^* и $B(y_n)$, $c(A_n^*, B(y_n))$. Неразглашение информации алгоритмом A формулируется симметричным образом.

Если функция f тривиальна (то есть, всюду равна пустому слову), то мы будем говорить, что о конфиденциальном вычислении функции g.

Замечание 12. Заметим, что из неразглашения для нечестных игроков следует разглашения для полу-честных игроков лишь в ослабленном виде. Для неразглашения алгоритмом B ослабленное условие такое: симулятор, получив на вход случайные биты r алгоритма A и его вход x, может найти некоторое слово x^* и, узнав $f(x^*, y)$, породить случайную величину, вычислительно неотличимую от $\langle r, c(A(x, r), B(y))\rangle$. Для этого симулятор для нечестных игроков надо применить к схеме, моделирующей работу алгоритма $A(x, r)$. Довольно странно, что этому симулятору может понадобиться значение $f(x^*, y)$, отличное от $f(x, y)$. Во всех примерах, которые у нас будут, симулятор со схемой $A(x, r)$ на входе будет выдавать $x^* = x$. Можно было бы включить это ограничение в требова-

ние неразглашения для нечестных игроков, но неясно, зачем это нужно.

12.1 Забывающая передача (oblivios transfer)

Так называется протоколы конфиденциального вычисления функции f, где $f(x,i) = x_i$ (i-ый бит x). Если длина x предполагается фиксированной и равной m, то такие протоколы называются OT_1^m.

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 такая перестановка $f_e(x)$. Применив теорему Левина-Голдрайха, мы 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) при известном e.

Теорема 39. *Если существует улучшенная односторонняя перестановка с секретом, то существует и протокол забывающей передачи для получестных игроков.*

Доказательство. Пусть входом Алисы является строка x длины m, а входом Боба — число i от 1 до m.

> **Протокол.** Алиса выбирает случайно пару e, d и посылает e Бобу. Боб, запустив m раз алгоритм порождения слов из D^e, выбирает m случайных строк $y_1, \ldots, y_m$ из D^e, применяет к i-ой строке функцию f_e (а к остальным строкам не применяет) и посылает полученные m строк Алисе. Алиса, получив кортеж из m строк $z_1, \ldots, z_m$, посылает обратно Бобу кортеж из m 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 m битов $\sigma_1, \ldots, \sigma_m$, 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 полу-честных 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, а не только i-ый. Кроме того, нечестная Алиса могла послать Бобу слово e, для которого f_e вообще не определена. Алгоритм вычисления f_e в этом случае выдаёт неизвестно что, например, пустое слово. Поэтому, может получиться, что изучив слова $z_1, \ldots, z_m$, Алиса найдёт i.

Сначала проверим, что Боб не разглашает информации. Действительно, последнее сообщение делается Алисой, поэтому достаточно проверить что первые два сообщения из протокола общения $A(x)$ и $B(i)$ можно сгенерировать, не зная i. Первое сообщение есть случайно выбранный открытый ключ e, а второе равно $(y_1, \ldots, f_e(y_i), \ldots, y_m)$. Поскольку f_e — перестановка и распределение y_i статистически неотличимо от равномерного распределения на D_e, второе сообщение статистически неотличимо от $(y_1, \ldots, y_i, \ldots, y_m)$ (при любом фиксированном e), а последнее можно сгенерировать, не зная i.

Теперь проверим, что Алиса разглашает только x_i. Для простоты обозначений предположим, что $m = 2$. Пусть $G(e, r)$ обозначает выход алгоритма генерации равномерного распределения на D_e при случайном числе r на его входе. Пусть Боб использовал случайные биты r_1, r_2 для генерации y_1, y_2. Нам нужно научиться порождать случайную величину, вычислительно неотличимую от

$$r_1, r_2, e, z_1, z_2, x_1 \oplus h_e(f_e^{-1}(z_1)), x_2 \oplus h_e(f_e^{-1}(z_2)) \tag{12.1}$$

зная только i и x_i. Здесь

$$(z_1, z_2) = \begin{cases} (f_e(y_1), y_2)), & \text{если } i = 1, \\ (y_1, f_e(y_2)), & \text{если } i = 2, \end{cases}$$

где $y_1 = G(e, r_1)$, $y_2 = G(e, r_2)$. Для этого запускаем алгоритм генерации ключей e, d и выбираем случайно r_1, r_2. После этого выдаем

$$r_1, r_2, e, z_1, z_2, \sigma_1, \sigma_2, \tag{12.2}$$

где z_1, z_2 определяются точно так же, как и раньше, а

$$(\sigma_1, \sigma_2) = \begin{cases} (x_1 \oplus h_e(y_1), b), & \text{если } i = 1, \\ (b, x_2 \oplus h_e(y_2), & \text{если } i = 2, \end{cases}$$

где b — случайный бит. Почему эта случайная величина вычислительно неотличима от (12.1)? Пусть, скажем, $i = 2$. Тогда случайные величины, неотличимость которых нужно доказать, имеют вид

$$r_1, r_2, e, f_e(y_1), y_2, x_1 \oplus h_e(y_1), x_2 \oplus h_e(f_e^{-1}(y_2)) \tag{12.3}$$

и

$$r_1, r_2, e, f_e(y_1), y_2, x_1 \oplus h_e(y_1), b.$$

Разница у них только в последнем компоненте. Ясно, что в последнем компоненте второй случайной величины можно $x_2 \oplus b$ заменить просто на b, не изменив её распределения. Получится случайная величина

$$r_1, r_2, e, f_e(y_1), y_2, x_1 \oplus h_e(y_1), x_2 \oplus b. \tag{12.4}$$

Нам известно, что случайные величины

$$r_2, e, h_e(f_e^{-1}(y_2)) \quad \text{и} \quad r_2, e, b$$

вычислительно неотличимы. Нетрудно заметить, что случайные величины (12.3) и (12.4) могут быть получены из этих случайных величин применением некоторой вероятностной схемы полиномиального размера. Поэтому они также вычислительно неотличимы. □

Теорема 40. *Если существует улучшенная односторонняя перестановка с секретом и для любого полиномиально разрешимого отношения $R_n(x, y)$ существует неразглашающий протокол знания свидетелей со свойством сильной полноты, то существует и протокол забывающей передачи для нечестных игроков.*

Доказательство. Будем обозначать через (e, d) открытый и закрытый ключи данного семейства улучшенных перестановок с секретом f_e. В доказательстве нам понадобится также существование протокола привязки (R, S, T), очевидно, гарантируемое условием теоремы.

> **Протокол.** 1) Алиса выбирает случайно пару e, d с помощью алгоритма генерации ключей K. Если генерация окончилась неудачно, то дальше Алиса и Боб посылают только тривиальные сообщения, скажем, пустые. Фактически это означает, что протокол на этом заканчивается. Иначе Алиса и посылает e Бобу.

2) Алиса и Боб запускают неразглашающий протокол доказательства того, что Алиса знает случайные биты для алгоритма генерации ключей K, для которых алгоритм K выдаст ключ e (тем самым доказывает, что f_e есть перестановка D^e).

Если доказательство закончилось безуспешно (алгоритм проверки выдал 0), то протокол на этом обрывается. Иначе Боб должен, запустив m раз алгоритм порождения слов из D^e, выбрать m случайных строк $y_1, \ldots, y_m$ из D^e, применить к i-ой строке функцию f_e (а к остальным строкам не применять!) и послать полученные слова $z_1, \ldots, z_m$ Алисе. При этом надо убедить Алису, что он действовал честно. В частности, убедить Алису в том, что он использовал случайные биты при выборе $z_1, \ldots, z_m$. При этом Боб не может разглашать эти случайные биты. Это делается следующим образом.

Пусть для честного изготовления $z_1, \ldots, z_m$ нужна случайная строка r известной обоим длины l. При честном образе действий посланные l слов получаются применением к секретному входу i, случайному слову r и слову e некоторой известной обоим полиномиально вычислимой функции F. Сначала Алиса и Боб находят строку r, выполняя протокол бросания монетки по телефону l раз.

3) Боб выбирает случайную последовательность b длины l и посылает её Алисе, используя протокол привязки.

4) Алиса выбирает случайную последовательность a длины l и посылает её Бобу.

5) Боб посылает Алисе $z = (z_1, \ldots, z_m) = F(i, a \oplus b, e)$.

6) Алиса и Боб запускают неразглашающий протокол доказательства того, что Боб знает i, b и ключ k, для которых $z = F(i, a \oplus b, e)$ и алгоритм раскрытия сундучка R, получив на вход привязку из пункта 3 и ключ k, выдаёт b.

Если протокол закончился неудачей (алгоритм проверки выдал 0), то оба алгоритма посылают в дальнейшем только пустые сообщения.

7) Иначе Алиса посылает Бобу слово σ длины m, j-ый бит равен $h_e(f_e^{-1}(z_j)) \oplus x_j$. Здесь z_j обозначает j-ую строку из

присланной в пункте 5 последовательности строк.

8) Наконец, Боб выдает $\sigma_i \oplus h_e(y_i)$.

Полнота построенного протокола очевидна. Проверим неразглашение.

Неразглашение информации Алисой: Пусть Боб применяет некоторый алгоритм, реализуемый схемой полиномиального размера B_n^* и пусть фиксирована последовательность входов x_n полиномиальной длины для Алисы. Единственное место, в протоколе, в котором действия Алисы зависят от секретного входа, это пункт 7. Поэтому до этого места симулятор просто запускает алгоритм Алисы и записывает в протокол посланные ей и схемой Боба сообщения. Точнее, это делается до пункта 6. А в пункте 6 симулятор моделирует доказательство с нулевым разглашением, извлекая из схемы некоторые i, b и k такие, что z, посланное в пункте 6, равно $F(i, a \oplus b, e)$ и алгоритм раскрытия сундучка R, получив на вход привязку из пункта 3 и k выдаёт b, если доказательство закончилось успешно. У нас есть гарантия, что вероятность успешного извлечения при условии, что доказательство закончилось успешно, приблизительно равна 1.

После этого симулятор запрашивает x_i и записывает в протокол последовательность из m случайных битов, в которой он заменяет i-ый бит на $x_i \oplus h_e(y_i)$.

Почему полученный протокол общения вычислительно неотличим от настоящего? Пусть для простоты обозначений $m = 2$. При известных e и случайных битах r, использованных для генерации y, трудный бит $h_e(f_e^{-1}(y(r))$, вычислительно неотличим от случайного бита. По требованию недвусмысленности привязки, существует слово b_1b_2 длины l, для которого алгоритм раскрытия сундучка выдает b_1b_2 или $\perp$. При известном e слово $r_i = a_i \oplus b_i$ имеет равномерное распределение, поэтому $h_e(f_e^{-1}(y_i(r_i)))$ неотличимо от случайных битов при известных e, b_i, a_i. Всё, что Алиса посылает Бобу, есть функция, вычислимая схемой полиномиального размера, от e, a_1, a_2. Поэтому если заменить $h_e(f_e^{-1}(y_i(r_i)))$ на случайный бит, то полученный протокол беседы будет вычислительно неотличим от исходного. Осталось заметить, что после такой замены мы как раз получим протокол, выданный симулятором.

Неразглашение информации Бобом: Пусть Алиса применяет некоторый алгоритм, реализуемый схемой полиномиального размера A_n^* и пусть фиксирована последовательность входов i_n для Боба. Симулятор просто

моделирует общение алгоритма Боба со схемой A_n^*, считая секретный вход Боба равным, скажем, 1.

Почему полученный протокол общения вычислительно неотличим от настоящего? Рассмотрим два случая.

Первый случай. Слово e, посланное Алисой в первом раунде, не могло быть сгенерировано алгоритмом K. Тогда с приблизительно единичной вероятностью доказательство в пункте 2 закончится неудачей, и в дальнейшем сообщения Боба пустые, поэтому не зависят от его входа.

Первый случай. Слово e, посланное Алисой в первом раунде, могло быть сгенерировано алгоритмом K. Тогда f_e является перестановкой D^e.

Давайте изменим образ действия Боба так, чтобы в пункте 3 он привязывался к слову b из одних нулей, а не к тому, который он получил, бросая монетку. Протокол общения, который получится в этому случае, вычислительно неотличим от исходного, поскольку алгоритм Боба T, примененный в пункте 3, не разглашает информации. Поэтому достаточно доказать, что протоколы общения для этого нового алгоритма Боба с A^* вычислительно неотличимы друг от друга при $i = 1, 2$.

Последовательность $a \oplus b$ равномерно распределена, поскольку b выбирается независимо от a (мы можем считать, что Боб бросает монетку после Алисы). Поэтому последовательности z при разных секретных входах Боба i вычислительно неотличимы друг от друга. Поэтому протоколы общения вплоть до шага 1–5 вычислимо неотличимы друг от друга при разных i. В дальнейшем мы выполняем неразглашающий алгоритм, секретным входом которого можно считать i, все предшествующие сообщения и случайные биты, использованные ранее (из них он может восстановить k, a, b). Поэтому мы можем воспользоваться теоремой о неразглашении при последовательном выполнении. □

Замечание 13. Заметим, что условия теоремы гарантируют существование алгоритма привязки, удовлетворяющего требованию сильной полноты, поэтому мы можем предполагать, что использованный протокол привязки имеет это свойство. Допустим, что и использованный протокол доказательства с нулевым разглашением также имеет это свойство. К сожалению, всё это ещё не гарантирует, что построенный в доказательстве теоремы протокол также удовлетворяет требованию сильной полноты. Причина кроется в том, что на первом шаге с ненулевой вероятностью Алиса может не найти ключа e. Однако это единственная причина. Поэтому если Алиса действует честно, а Боб действует честно

за исключением выбора случайных битов (то есть, случайные биты, нужные его алгоритму он выбирает по своему усмотрению, используя свой вход и всю информацию, полученную от Алисы к данному моменту, а в остальном следует своему алгоритму). Тогда алгоритм Боба выдаст x_i с приблизительно единичной вероятностью.

12.2 Конфиденциальное вычисление произвольной полиномиально вычислимых функции

Пусть имеется полиномиально вычислимая последовательность функций $f_n : \{0,1\}^{k(n)} \times \{0,1\}^{l(n)} \to \{0,1\}$. Мы построим алгоритм конфиденциального вычисления пары функций $\Lambda, f_n(x,y)$ (то есть, Алиса не узнает ничего, а Боб узнаёт $f_n(x,y)$, но не больше). В качестве исходных примитива нам понадобится протокол забывающей передачи OT_1^4 для нечестных противников, удовлетворяющий замечанию 13.

Протокол. 1) Сначала Алиса и Боб строят схему C_n полиномиального размера, вычисляющую f_n. Пусть $x_1 \dots x_k$ — входное слово Алисы, а $y_1 \dots y_l$ — входное слово Боба. Обозначим через s обозначает общее число проводников схемы C_n (входы схемы $x_1, \dots, x_k, y_1 \dots y_l$ также считаются её проводникам). Алиса и Боб по очереди вычисляют для каждого проводника z_i пару булевских переменных a_i, b_i таких, что $a_i \oplus b_i = z_i$. Причём делают они это таким образом, что для каждой переменной z_i, за исключением входных переменных, Алиса знает a_i, но не знает b_i, а Боб наоборот знает b_i, но не знает a_i.

2) Для этого производится инициализация: Алиса выбирает k случайных бит $a_1 \dots a_k$ и посылает Бобу $b_1 = a_1 \oplus x_1 \dots a_k \oplus x_k$. Аналогично, Боб выбирает l случайных бит $b_{k+1} \dots b_{k+l}$ и посылает Алисе $a_{k+1} = b_{k+1} \oplus y_1 \dots b_{k+l} \oplus y_l$. Теперь у Алисы и Боба есть переменные a_i и b_i такие, что $a_i \oplus b_i = z_i$ для всех входных переменных z_i. Ни Алиса, ни Боб в дальнейшем не будут использовать сами входные переменные, вместо них Алиса будут использовать a_i, а Боб b_i.

3) После этого для всех $i = k+l+1, \ldots, s$ Алиса и Боб находят a_i, b_i так чтобы $a_i \oplus b_i = z_i$. Для этого они выполняют для каждого такого i следующие. Пусть z_i получается из каких-то проводников схемы z_j, z_m какой-то бинарной операцией, скажем $\wedge$ (случаи других бинарных операций или унарных операций аналогичны). Тогда Алиса выбирает случайным образом a_i и забывающе пересылает Бобу

$$b_i = a_i \oplus [(a_j \oplus b_j) \wedge (a_m \oplus b_m)]. \tag{12.5}$$

Переменные a_j, a_m стали известны Алисе на предыдущих шагах, а переменные b_j, b_m стали известны Бобу. Алиса не знает b_j, b_m, поэтому она формирует четыре бита, по одному биту на каждое из четырех возможных значений пары b_j, b_m. Точнее Алиса и Боб выполняют протокол OT_1^4 в котором входом Алисы является четверка битов, в котором бит с номером $b_j b_m$ определяемая равенством (12.5), а входом Боба является номер $b_j b_m$ в двоичной записи. Заметим, что Боб узнаёт бит b_i, задаваемый равенством (12.5), а этот бит равномерно распределён и не зависит от всех предыдущих сообщений. То есть, по существу, Боб не узнает ничего. И Алиса ничего не узнаёт, но вместе они теперь знают a_i, b_i, для которых $a_i \oplus b_i = z_i$.

4) Наконец Алиса сообщает Бобу a_s; теперь Боб может вычислить $z_s = a_s \oplus b_s = f_n(x, y)$ и выдаёт его в качестве результат протокола.

Теорема 41. *Построенный протокол является протоколом конфиденциального вычисления функции f_n для получестных противников. Более того, алгоритм Боба имеет нулевое разглашение (для нечестного противника).*

Доказательство. Алгоритм Боба состоит из последовательного выполнения неразглашающих алгоритмов. Поэтому последнее утверждение следует из Замечания 10.

Полнота протокола очевидна. Объясним, почему алгоритм Алисы разглашает только $f(x, y)$. Симулятор, получив на вход $f(x, y)$, случайные биты Боба и его вход y, посылает со стороны Алисы случайные биты, а со стороны Боба в точности то же, что должен посылать Боб. Затем

симулятор применяет $s-k-l$ раз симулятор для протокола забывающей передачи. При этом в качестве передаваемого бита, который этот симулятор запросит, ему даётся случайный бит. Поскольку и в реальном протоколе этот бит случаен, полученный на шагах 2,3 симулированный протокол вычислительно неотличим от настоящего. На последнем пятом шаге бит a_s, переданный Алисой, удовлетворяет соотношению $b_s = a_s \oplus z_s$, то есть $a_s = b_s \oplus z_s$, где b_s — случайный бит, который был выбран симулятором в самом конце четвертого шага. Этот бит симулятор не может выбирать случайно, тут ему как раз и понадобится $z_s = f(x,y)$. □

Недостатком этого протокола является то, что при нечестном поведении Боба Алиса может разгласить совсем не то, что разрешено. Напомним, что ей разрешено разгласить $f(x,y)$ для какого-то одного y. Пусть, скажем, $f(x,y)$ не зависит от y и равно $x_1 \wedge x_2$. Допустим, Боб действует честно на втором шаге, а на третьем шаге узнаёт у Алисы $a_3 \oplus [(a_1 \oplus (b_1 \oplus 1)) \wedge (a_2 \oplus b_2)]$. На четвертом шаге Алиса сообщит Бобу a_3 и таким образом Боб узнаёт $(a_1 \oplus (b_1 \oplus 1)) \wedge (a_2 \oplus b_2) = \neg x_1 \wedge x_2$ вместо положенного $x_1 \wedge x_2$.

Исправляется этот недостаток следующим образом: после третьего шага Боб должен доказать Алисе, что действовал честно, то есть во всех применениях протокола забывающей передачи он просил передать именно то, что ожидалось. При этом Бобу не обязательно доказывать, что он использовал честные случайные биты в качестве $b_{k+1} \dots b_{k+l}$ и в протоколе забывающей передачи, если последний удовлетворяет Замечанию 13.

Теорема 42. *Если существует улучшенная односторонняя перестановка с секретом, то существует и протокол конфиденциального вычисления любой полиномиально вычислимой булевой функции для нечестных игроков.*

Доказательство. По теореме 40 существует протокол забывающей передачи OT_1^4, удовлетворяющий Замечанию 13. Кроме того, по теоремам 26 и 20 существует неразглашающий протокол доказательства знания свидетелей с условием сильной полноты. Мы будем использовать эти протоколы в следующем протоколе конфиденциального вычисления полиномиально вычислимой функции $f_n : \{0,1\}^{k(n)} \times \{0,1\}^{l(n)} \to \{0,1\}$.

> **Протокол.** Сначала повторяются шаги 1–3 из протокола для получестных противников, приведенного на стр. 159. Напомним, что на этих шагах они делали следующее.

1) Алиса и Боб строят схему C_n полиномиального размера, вычисляющую f_n.

2) Инициализация: Алиса выбирает k случайных бит $a_1 \dots a_k$ и посылает Бобу $b_1 = a_1 \oplus x_1 \dots a_k \oplus x_k$. Аналогично, Боб выбирает l случайных бит $b_{k+1} \dots b_{k+l}$ и посылает Алисе $a_{k+1} = b_{k+1} \oplus y_1 \dots b_{k+l} \oplus y_l$.

3) Для всех $i = k + l + 1, \dots, s$ Алиса и Боб делают следующее. Пусть z_i получается из каких-то проводников схемы z_j, z_m какой-то бинарной операцией, скажем $\wedge$. Тогда Алиса выбирает случайным образом a_i и забывающе пересылает Бобу

$$b_i = a_i \oplus [(a_j \oplus b_j) \wedge (a_m \oplus b_m)].$$

4) Теперь Боб должен доказать, что он просил Алису передать именно те биты, которые следовало. Для этого Боб доказывает Алисе знание слова y, битов $b_{k+1} \dots b_{k+l}$ и случайных битов r для протокола забывающей передачи таких, что при честном выполнении алгоритма Боба на шагах 2,3 для этих y, b, r посылается именно то, что и было послано.

Поскольку протокол забывающей передачи удовлетворяет Замечанию 13, если это в самом деле верно для каких-то y, b, r, то $a_s \oplus b_s = f(x, y)$ с приблизительно единичной вероятностью.

5) Если доказательство окончилось неудачно, то Алиса посылает Бобу пустое слово, а иначе она сообщает ему a_s.

Условие полноты очевидно выполнено. Алгоритм Боба по-прежнему не разглашает информации, что доказывается так же, как и раньше, поскольку на шаге 4 Боб по-прежнему выполняет алгоритм с нулевым разглашением.

Наиболее сложная часть — убедиться в том, что Алиса разглашает не более $f(x, y)$ для какого-то y. Для этого нам надо построить симулятор, который получает на вход алгоритм Боба B_n^*, реализуемый схемой полиномиального размера, и порождает случайную величину, неотличимую от протокола общения честной Алисы с B_n^*. При этом симулятору разрешается один раз узнать $f(x, y)$ для какого-то y, который он сам выберет.

Для этого симулятор запускает алгоритм Алисы и алгоритм Боба и просто записывает все посланные сообщения на на шаге 2. На шаге 3 он запускает симулятор для протокола забывающей передачи, и когда тот, спросит бит из базы данных, даёт ему случайный бит. При этом каждый раз на вход симулятора дается схема B_n^*, в которую запаяны все до сих пор симулированные сообщения. Полученный протокол вычислительно неотличим от настоящего, что доказывается так же, как и теорема о последовательном выполнении неразглашающего алгоритма.

На шаге 4 опять запускается симулятор для неразглашающего протокола доказательства, который извлекает свидетелей y, b, r. По условию вероятность того, что они извлекутся приблизительно равна 1 при условии, что алгоритм доказательство прошло успешно.

Если доказательство закончилось неудачей, то последнее сообщение Алисы пусто, и симулятор записывает в протокол пустое слово. Иначе симулятор узнаёт значение $f(x, y)$ и записывает в протокол $a_s = b_s \oplus f(x, y)$, где b_s есть случайный бит, выбранный симулятором для протокола забывчивой передачи.

Почему симулированный протокол вычислительно неотличим от настоящего? Для шагов 2–4 это доказывается точно так же, как и в теореме последовательном выполнении неразглашающего алгоритма. Поэтому нам остаётся лишь доказать, что последнее сообщение, записанное симулятором (пустое слово или $a_s = b_s \oplus f(x, y)$) с почти единичной вероятностью равно тому сообщению, которое послала бы Алиса, получив те же сообщения, которые симулятор записал в протокол. Назовем это событием G.

Для этого рассмотрим событие E “доказательство закончилось удачно”. Условная вероятность события G при условии $\bar{E}$ (дополнение до E) равна 1.

Поэтому достаточно доказать, что условная вероятность G при условии E приблизительно единичная. Если доказательство закончилось удачно, то с приблизительно единичной вероятностью симулятор извлечёт такие y, b, r, что при честном выполнении алгоритма Боба на шагах 2,3 для этих y, b, r посылается именно то, что и было послано. Отсюда следует, что на шаге 5 Алиса посылает $b_s = a_s \oplus f(x, y)$, следовательно, событие G выполнено. Таким образом, условная вероятность G при условии E приблизительно единичная, что и требовалось доказать. □

Замечание 14. Аналогичным образом можно построить протокол вычис-

ления любой полиномиально вычислимой функции: на последнем шаге Алиса сообщает Бобу значения всех своих случайных битов, соответствующих выходам схемы. Протокол конфиденциального вычисления любой пары полиномиально вычислимых функций f, g можно построить, выполним последовательно протокол конфиденциально вычисления каждой из них. Неразглашение информации для полученного протокола доказывается так же, как и лемма о последовательном выполнении неразглашающих алгоритмов.

Глава 13

Построение генератора ПСЧ из односторонней функции

Определение 16. Семейство функций $g_z : \{0,1\}^n \to \{0,1\}^m$, где z пробегает некоторое конечное множество индексов, называется универсальным семейством хэш-функций, если для любых двух различных $x_1, x_2 \in \{0,1\}^n$ и любых $y_1, y_2 \in \{0,1\}^m$ вероятность события $[g_Z(x_1) = y_1,\ g_z(x_2) = y_2]$ равна 2^{-2m} (здесь предполагается, что Z выбирается из множества индексов с равномерным распределением). Иными словами, для любого x случайная величина $g_Z(x)$ имеет равномерное распределение в $\{0,1\}^m$ и для различных x_1, x_2 случайные величины $g_Z(x_1)$ и $g_Z(x_2)$ независимы.

Задача 65. Для каждой матрицы A размера $m \times n$ и вектора b длины m из нулей и единиц рассмотрим функцию $x \mapsto Ax + b$ (все операции выполняются в поле из двух элементов). Докажите, что при равномерном распределение на парах A, b получается универсальное семейство хэш-функций.

Лемма (о сглаживании). *Пусть X есть случайная величина со значениями в множестве слов длины n. Пусть g_z является универсальным семейством хэш-функций типа $\{0,1\}^n \to \{0,1\}^m$, где z пробегает множество слов длины k. Обозначим через Z случайную величину, равномерно распределённую среди слов длины k. Тогда L_1-расстояние между распрелелением случайной величиной $Zg_Z(X)$ и равномерным распределением на словах длины $k+m$ не превосходит $2^{(m-H_1(X))/2}$.*

Доказательство. L_1-расстояние между распрелелением случайной величиной $Yg_Z(X)$ и равномерным распределением на словах длины $k+m$

равно

$$\sum_{z\in\{0,1\}^k}\sum_{y\in\{0,1\}^m}|2^{-k}\Pr[g_z(X)=y]-2^{-m-k}|,$$

что в 2^m раз больше среднего значения по z,y величины

$$|\Pr[g_z(X)=y]-2^{-m}|.$$

По неравенству между средним арифметическим и средним квадратическим среднее значение этой случайной величины не превосходит квадратного корня из среднего значения её квадрата. Поэтому нам достаточно оценить сверху величиной $2^{-m-H_1(X)}$ среднее значение по y,z величины

$$(\Pr[g_z(X)=y]-2^{-m})^2.$$

А для этого достаточно доказать, что для любого фиксированного y выполнено

$$\mathbf{E}_z(\Pr[g_z(X)=y]-2^{-m})^2<2^{-m-H_1(X)}.$$

Для этого, во-первых, заменим здесь вероятность $\Pr[g_z(X)=y]$ на среднее значение предиката $[g_z(X)=y]$, равного 1, если выражение в скобках истинно, и равного 0 иначе. Доказываемое неравенство перепишется в виде

$$\mathbf{E}_z(\mathbf{E}_x[g_z(x)=y]-2^{-m})^2<2^{-m-H_1(X)}.$$

Во-вторых, рассмотрим случайную величину X' с тем же распределением, что у X, но независимую от X. С её использованием доказываемое неравенство можно переписать в виде

$$\mathbf{E}_z(\mathbf{E}_x[g_z(x)=y]-2^{-m})(\mathbf{E}_{x'}[g_z(x')=y]-2^{-m})<2^{-m-H_1(X)}.$$

Поскольку для независимых случайных величин среднее значение произведения равно произоведению средних, наше неравенство можно переписать так:

$$\mathbf{E}_z\,\mathbf{E}_x\,\mathbf{E}_{x'}([g_z(x)=y]-2^{-m})([g_z(x')=y]-2^{-m})<2^{-m-H_1(X)}.$$

При вычислении этого среднего мы можем переставить операторы усреднения и считать, что внутренним является $\mathbf{E}_z$. Таким образом, нам достаточно доказать неравенство

$$\mathbf{E}_x\,\mathbf{E}_{x'}\,\mathbf{E}_z([g_z(x)=y]-2^{-m})([g_z(x')=y]-2^{-m})<2^{-m-H_1(X)}.$$

Поскольку семейство функций g_z предполагается универсальным. $\mathbf{E}_z[g_z(x) = y] = 2^{-2m}$ (и аналогично для среднего по x'). Кроме того, если $x \neq x'$, то события $[g_z(x) = y]$ и $[g_z(x') = y]$ независимы. Таким образом, для $x \neq x'$ среднее по z величины $([g_z(x) = y] - 2^{-m})([g_z(x') = y] - 2^{-m})$ равно произведению средних, то есть, нулю. Для $x = x'$ среднее по z величины $([g_z(x) = y] - 2^{-m})([g_z(x') = y] - 2^{-m})$ меньше 2^{-m}, что вычисляется прямым подсчетом:

$$\begin{aligned}\mathbf{E}_z([g_z(x) = y] - 2^{-m})^2 &= \mathbf{E}_z[g_z(x) = y] - 2 \cdot \mathbf{E}_z[g_z(x) = y]2^{-m} + 2^{-2m} \\ &= 2^{-m} - 2 \cdot 2^{-2m} + 2^{-2m} < 2^{-m}.\end{aligned}$$

По условию

$$\sum_x \Pr[X = x]\Pr[X' = x]2^{-m} = 2^{-m}\sum_x (\Pr[X = x])^2 = 2^{-m-H_1(X)},$$

откуда и следует доказываемое неравенство. □

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.