

# „Теоретико-сложностные основы криптографии“.

## Заметки к курсу в СПбАУ

А.В. Смаль

12 июня 2018 г.

### Аннотация

Курс посвящён изучению теоретических оснований, на которых строится надёжность криптографических протоколов.

### Содержание

<b>1. Совершенная надёжность</b>	<b>3</b>
<b>2. Односторонние функции</b>	<b>3</b>
2.1. Односторонние функции с худшем случае . . . . .	4
2.2. Односторонние функции для алгоритмов . . . . .	4
2.3. Односторонние функции для неравномерного противника . . . . .	4
2.4. Примеры односторонних функций . . . . .	5
2.5. Построение сильно односторонних функций из слабо односторонних . . . . .	6
2.6. Частичные односторонние функции . . . . .	8
<b>3. Генераторы псевдослучайных чисел</b>	<b>9</b>
3.1. Вычислительно неотличимые случайные величины . . . . .	9
3.2. Генераторы псевдослучайных чисел . . . . .	11
3.3. Трудный бит . . . . .	12
<b>4. Протоколы шифрования с секретным ключом</b>	<b>17</b>
4.1. Эффективная схема шифрования с закрытым ключом . . . . .	21
<b>5. Протоколы шифрования с открытым ключом</b>	<b>21</b>
<b>6. Протокол привязки (схема обязательства, commitment protocol)</b>	<b>23</b>
6.1. Неинтерактивный протокол привязки к строке . . . . .	23
<b>7. Интерактивные протоколы</b>	<b>24</b>
7.1. Интерактивные протоколы привязки к строке . . . . .	24
7.2. Протокол подбрасывания монетки . . . . .	25
<b>8. Протоколы с нулевым разглашением</b>	<b>26</b>
8.1. Понятие разглашения . . . . .	26
8.2. Языки с нулевым разглашением . . . . .	30

<b>9. Протоколы цифровой подписи</b>	<b>33</b>
9.1. Одноразовый протокол подписи одного бита . . . . .	34
9.2. Одноразовый протокол подписи $p(n)$ битов . . . . .	34
9.3. Одноразовый протокол подписи полиномиального числа битов . . . . .	35

## Введение

Мы будем предполагать, что алгоритмы шифрования/дешифрования всем известны (т.е. по security by obscurity).

### 1. Совершенная надёжность

**Определение 1.1.** Система шифрования с закрытым ключом — это пара алгоритмов  $E(k, m)$  и  $D(k, c)$ , такая, что для любых  $k$  и  $m$  выполняется  $D(k, E(k, m)) = m$ . Система называется *совершенно надёжной*, если для любых двух сообщений  $m_1$  и  $m_2$  случайные величины  $E(k, m_1)$  и  $E(k, m_2)$  при  $k \leftarrow \mathcal{U}(K)$  распределены одинаково ( $K$  — пространство ключей).

*Замечание 1.1.* Система шифрования с одноразовым шифроблокнотом является совершенно надёжной.

*Замечание 1.2.* Для совершенной надёжности необходимо, чтобы длина ключа была не менее длины сообщения.

**Теорема 1.1.** Пусть  $P = NP$ . Тогда для любой системы шифрования с закрытым ключом  $(E, D)$  с полиномиальным алгоритмом  $E$ , в которой  $|m| > |k|$ , существуют сообщения  $m_0$  и  $m_1$  и полиномиальный алгоритм  $A$ , для которого

$$\left| \Pr_k[A(E(k, m_0)) = 1] - \Pr_k[A(E(k, m_1)) = 1] \right| \geq \frac{1}{2}.$$

*Доказательство.* Не уменьшая общности предположим, что  $K = \{0, 1\}^{n-1}$ . Возьмём в качестве  $m_0 = 0^n$ . Пусть  $S = \{E(k, 0^n) \mid k \in K\}$ . Легко видеть, что  $S \in NP$  и  $|S| \leq 2^{n-1}$ . Возьмём в качестве алгоритма  $A$  полиномиальный разрешающий алгоритм для  $S$ , т.е.  $A(y) := [y \in S]$  (он существует по предположению  $P = NP$ ).

Для каждого сообщения  $m$  рассмотрим  $t_m = |\{k \mid E(k, m) \in S\}|$ . Если существует сообщение  $m^*$ , для которого  $t_{m^*} \leq 2^{n-2}$ , то  $m_1 = m^*$  удовлетворяет требованиям.

Предположим теперь, что  $t_m > 2^{n-2}$  для любого  $m$ . Это значит, что существуют более  $2^{n-2} \cdot 2^n = 2^{2n-2}$  пар ключ-сообщение  $(k, m)$ , для которых  $E(k, m) \in S$ . Следовательно, для некоторого  $y \in S$  существует более  $2^{2n-2}/|S| \geq 2^{n-1}$  пар  $(k, m) : E(k, m) = y$ , т.е. существуют ключ  $k$  и два различных сообщения  $m'$  и  $m''$ :  $E(k, m') = E(k, m'')$ . Это противоречит корректности системы шифрования.  $\square$

### 2. Односторонние функции

Доказывать надёжность криптографических протоколов без каких-либо предположений, к сожалению, не получается — из такого доказательства следовало бы  $P \neq NP$ . Было бы здорово показать, что криптография возможна, если  $P \neq NP$ , но это тоже не получается сделать. Поэтому в дальнейшем мы будем отталкиваться от более сильного предположение — предположения о существовании *односторонней функции*.

В дальнейшем мы будем рассматривать семейства функций  $f_n : \{0, 1\}^{k(n)} \rightarrow \{0, 1\}^{l(n)}$ , где  $k(n)$  и  $l(n)$  будут некоторыми полиномами. Кроме того, нас почти всегда будут интересовать функции, которые можно вычислить за полиномиальное время.

**Определение 2.1.** Семейство функций  $f_n : \{0, 1\}^{k(n)} \rightarrow \{0, 1\}^{l(n)}$  называется *полиномиально вычислимым*, если имеется алгоритм, который получая на вход число  $n$  и  $x$  длины  $k(n)$  вычисляет  $f_n(x)$  за полиномиальное от  $n$  время.

## 2.1. Односторонние функции с худшем случае

**Определение 2.2.** Полиномиально вычислимое семейство функций  $f_n : \{0, 1\}^{k(n)} \rightarrow \{0, 1\}^{l(n)}$  называется *односторонним в худшем случае*, если не существует полиномиально вычислимой функции  $g_n$ , что для любого  $x \in \{0, 1\}^{k(n)}$  верно  $f_n(g_n(f_n(x))) = f_n(x)$ .

**Теорема 2.1.** *Односторонние функции с худшем случае существуют  $\iff P \neq NP$ .*

*Доказательство.*

$\Rightarrow$  Пусть  $P = NP$ . Определим язык  $L = \{(1^n, y, z) \mid \exists x, |x| = k(n), z \sqsubset x, f_n(x) = y\}$ ,  $L \in NP$ . По предположению для  $L$  существует полиномиальный разрешающий алгоритм. Для нахождения прообраза  $y$  запустим этот алгоритм сначала на слове  $(1^n, y, \lambda)$ , где  $\lambda$  — пустая строка. Если это слово не принадлежит  $L$ , то  $y$  не имеет прообраза. В противном случае восстановим прообраз  $y$  по битам: сначала запустим алгоритм для слова  $(1^n, y, 0)$  и проверим, есть ли у  $y$  прообраз начинающийся с нуля. Далее аналогично восстановим второй и все последующие биты. Нам потребуется  $k(n) + 1$  запуск полиномиального алгоритма, т.е. прообраз можно найти алгоритмически за полиномиальное время.

$\Leftarrow$  Если  $P \neq NP$ , то можно построить одностороннюю в худшем на основе любой  $NP$ -трудной задачи. Пусть  $R(x, y)$  — это отношение, задающее  $NP$ -трудную задачу  $S$  (например, для  $S = SAT$ :  $R(\phi, a) = 1 \iff \phi(a) = 1$ ). Пусть  $f_n(x, y) = (x, R(x, y))$ . Если  $f_n^{-1}$  вычисляется за полиномиальное время, то и задачу  $S$  можно решить за полиномиальное время, вычислив  $f^{-1}(x, 1)$ .

□

## 2.2. Односторонние функции для алгоритмов

Мы будем определять *односторонние функции* (one-way function, owf) для противника, который является вероятностным полиномиальным алгоритмом, т.е. для *равномерного противника*.

**Определение 2.3.** Полиномиально вычислимое семейство  $f_n$  называется *слабо односторонним для равномерного противника*, если **существует** такой полином  $p$ , что для любого полиномиального вероятностного алгоритма  $R$  при всех достаточно больших  $n$

$$\Pr_{x,R}[f_n(R(1^n, f_n(x))) = f_n(x)] < 1 - \frac{1}{p(n)}.$$

**Определение 2.4.** Полиномиально вычислимое семейство  $f_n$  называется *сильно односторонним для равномерного противника*, если **для любого** полинома  $q$ , что для любого полиномиального вероятностного алгоритма  $R$  при всех достаточно больших  $n$

$$\Pr_{x,R}[f_n(R(1^n, f_n(x))) = f_n(x)] < \frac{1}{q(n)}.$$

## 2.3. Односторонние функции для неравномерного противника

Аналогичным образом можно определить односторонние функции для противника, являющегося последовательностью схем, т.е. для *неравномерного противника*.

**Определение 2.5.** Полиномиально вычислимое семейство  $f_n$  называется *слабо односторонним для неравномерного противника*, если **существует** такой полином  $p$ , что для любой последовательности схем  $C_n$  полиномиального размера при всех достаточно больших  $n$

$$\Pr_x[f_n(x) = f_n(C_n(f_n(x)))] < 1 - \frac{1}{p(n)}.$$

**Определение 2.6.** Полиномиально вычислимое семейство  $f_n$  называется *сильно односторонним для неравномерного противника*, если **для любого** полинома  $q$ , что для любой последовательности схем  $C_n$  полиномиального размера при всех достаточно больших  $n$

$$\Pr_x[f_n(x) = f_n(C_n(f_n(x)))] < \frac{1}{q(n)}.$$

*Замечание 2.1.* Односторонние функции для неравномерного противника можно было бы определять для *вероятностных* схем, т.е. для схем, которым на вход подают не только  $f_n(x)$ , но и некоторую строку со случайными битами  $r$ . Однако, легко показать, что от случайных битов в таких определениях можно избавиться: для этого нужно для каждого  $n$  выбрать одну “самую лучшую” строку  $r_n$ , на которой достигается максимальная вероятность обращения  $f_n$  и “зашить” её в схему. Нетрудно увидеть, что вероятность обращения при  $r = r_n$  будет не меньше, чем по всем  $r$  в среднем.

В дальнейшем мы часто будем говорить про односторонние *функции*, подразумевая под этим *семейства* односторонних функций. Когда говорят про *одностороннюю функцию*, то имеется в виду сильно односторонняя функция.

**Определение 2.7.** Если в определении односторонней функции убрать требование полиномиальной вычислимости, то получится определение *необратимой* функции.

## 2.4. Примеры односторонних функций

Неизвестно, существуют ли односторонние или хотя бы слабо односторонние функции (даже для равномерного противника). Доказательство их существования повлечёт за собой  $P \neq NP$ .

**Теорема 2.2.** Если  $P = NP$ , то любое полиномиально вычислимое семейство функций  $f_n$  не является слабо обратимым даже для равномерного противника. Более того, существует детерминированный алгоритм, который для всех  $x$  по  $n$  и  $f_n(x)$  за полиномиальное от  $n$  время находит некоторый прообраз  $f_n(x)$  длины  $k(n)$ .

*Доказательство.* См. пункт “ $\Rightarrow$ ” доказательства теоремы 2.1. □

*Пример 2.1* (произведение натуральных чисел). Семейство  $f_n$  устроено следующим образом:  $k(n) = l(n) = 2n$ , вход  $x$  делится пополам, каждая половинка представляет собой  $n$ -битовое число, результат  $f_n$  — произведение этих чисел (получится не более чем  $2n$ -битовое число).

*Пример 2.2* (SUBSET-SUM). Семейство  $f_n$  для SUBSET-SUM устроено следующим образом:  $k(n) = n^2 + n$ ,  $l(n) = n^2 + 2n + \lceil \log n \rceil$ , вход  $x$  разбивается на  $n + 1$  блок длины  $n$ , первые  $n$  блоков интерпретируются как  $n$ -битовые числа  $x_1, \dots, x_n$ , а последний блок интерпретируется как подмножество  $[n]$ . Тогда  $f_n(x) = \langle x_1, x_2, \dots, x_n, \sum_{i \in I} x_i \rangle$ .

## 2.5. Построение сильно односторонних функций из слабо односторонних

**Теорема 2.3.** *Если существуют слабо односторонние функции, то существуют и сильно односторонние функции (это верно для любых противников).*

*Доказательство.* Будем доказывать для равномерного противника — для неравномерного доказательство будет аналогичным. Пусть  $f$  — слабо односторонняя функция, т.е. существует такой полином  $p$ , что для любого полиномиального вероятностного алгоритма  $R$  при всех достаточно больших  $n$

$$\Pr_{x,r}[f_n(x) = f_n(R(1^n, f_n(x), r))] < 1 - \frac{1}{p(n)}.$$

Определим функцию  $F$ , которая определяется следующим соотношением:

$$F_n(x_1, x_2, \dots, x_N) = \langle f_n(x_1), f_n(x_2), \dots, f_n(x_N) \rangle,$$

т.е.  $F_n : \{0, 1\}^{N \cdot k(n)} \rightarrow \{0, 1\}^{N \cdot l(n)}$ . Для того, чтобы обратить  $F_n$  нам нужно  $N$  раз обратить функцию  $f_n$ . В каждом случае вероятность ошибки не меньше  $1/p(n)$ , поэтому общая вероятность успеха не более  $(1 - 1/p(n))^N$ . Для  $N = n \cdot p(n)$  эта вероятность близка к  $e^{-n}$ , что убывает быстрее любого обратного полинома. Таким образом функция  $F$  — сильно односторонняя.

В предыдущем рассуждении кроется ошибка. Дело в том, что мы предполагаем, что обращающий алгоритм будет обязательно устроен следующим образом: он будет пытаться  $N$  раз обратить  $f_n$ , т.е. найти прообразы для  $f_n(x_1), f_n(x_2), \dots, f_n(x_N)$ . Это не обязательно так — мы не можем предполагать, что этот алгоритм будет устроен каким-то конкретным образом. Поэтому предыдущее доказательство ошибочное, хотя получившаяся функция  $F$  действительно сильно односторонняя.

Корректное доказательство этого факта будет устроено другим образом. Мы предположим, что функция  $F$  не является сильно односторонней и из этого покажем, что в свою очередь функция  $f$  не является слабо односторонней. Для этого мы воспользуемся алгоритмом, который обращает  $F$ , для построения алгоритма обращения  $f$ . Предположим, что алгоритм  $R_F$  умеет обращать  $F$  с вероятностью успеха более  $1/q(n)$  для некоторого полинома  $q$ . Тогда мы покажем, что существует алгоритм  $R_f$ , который обращает  $f$  с вероятностью успеха более  $1 - 1/p(n)$ .

Алгоритм  $R_f$  мог бы быть устроен так: для обращения  $y$  мы выберем случайные  $x_2, x_3, \dots, x_N$  и запустим алгоритм  $R_F$  на входе  $\langle y, f_n(x_2), f_n(x_3), \dots, f_n(x_N) \rangle$ . Действительно, если  $R_F$  найдёт прообраз для этого входа, то он в т.ч. найдёт и прообраз для  $y$ . Вероятность успеха  $R_F$  на таком входе для случайного  $y$  не менее вероятности успеха  $R_F$  для случайных  $x_1, \dots, x_N$ . Однако нам этого недостаточно — мы хотели бы получить вероятность успеха близкую к единице.

Для этого будем использовать два дополнительных приёма:

- будем пытаться подставить  $y$  не только на место  $f_n(x_1)$ , а для каждого  $i$  будем запускать алгоритм  $R_F$  на входе  $\langle f_n(x_1), \dots, f_n(x_{i-1}), y, f_n(x_{i+1}), \dots, f_n(x_N) \rangle$  для случайных  $x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_N$ <sup>1</sup>;
- будем повторять каждую итерацию  $M$  раз для различных случайных независимых наборов  $x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_N$ .

<sup>1</sup>Этот приём необходим, т.к. в противном случае у нас не получится увеличивать вероятность успеха. Действительно, если представить, что алгоритм  $R_F$  не работает на строках, у которых первый бит нулевой, то вероятность успеха такого алгоритма вполне может быть  $1/2$ . Но тогда и вероятность успеха  $R_f$  не может быть выше  $1/2$ .

Давайте выделим один *раунд* алгоритма  $R_f$ : для каждого  $i$  выбирается независимый случайный набор входов  $x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_N$  и вызывается алгоритм  $R_F$  на входе  $\langle f_n(x_1), \dots, f_n(x_{i-1}), y, f_n(x_{i+1}), \dots, f_n(x_N) \rangle$ . Этот этап будет повторён  $M$  раз. Значение  $M$  мы выберем позже, это будут некоторый полином от  $n$ .

Через  $s_i(x)$  мы будем обозначать вероятность того, что алгоритм  $R_F$  найдёт прообраз  $\langle f_n(x_1), \dots, f_n(x_{i-1}), f_n(x), f_n(x_{i+1}), \dots, f_n(x_N) \rangle$  для случайных  $x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_N$ . Через  $\hat{s}(x)$  обозначим вероятность успеха одного раунда алгоритма  $R_f$ . Эта вероятность заведомо не меньше максимальной вероятности среди всех  $s_i(x)$ , т.е.  $\hat{s}(x) \geq \max_i s_i(x)$ .

Рассмотрим отдельно входы  $x$ , которые наш алгоритм  $R_f$  обращает с маленькой вероятностью, т.е. это “трудные” для обращения входы. Будем говорить, что  $x$  — трудный, если  $\hat{s}(x) < \epsilon$  для некоторого обратного полинома  $\epsilon$ , который мы выберем дальше. Долю трудных “трудных” слов среди всех слов длины  $n$  мы обозначим через  $\delta$ , т.е.  $\delta = \Pr_{x \leftarrow U_n}[\hat{s}(x) < \epsilon]$ .

Дальнейшее доказательство будет построено так: мы предположим, что получившийся алгоритм  $R_f$  не обращает функцию  $f$  с нужной вероятностью, т.е. вероятность его ошибки больше  $1/p(n)$ . Из этого будет следовать, что доля трудных слов  $\delta$  довольно большая (больше некоторого обратного полинома). А раз трудных слов много, то и вероятность успеха  $R_F$  не может быть больше  $1/q(n)$ . Таким образом мы придём к противоречию. Для реализации этого плана потребуются следующие две леммы.

**Лемма 2.1.** *Вероятность ошибки  $R_f$  при обращении слова  $f_n(x)$  для случайного  $x$  не больше  $\delta + (1 - \epsilon)^M$ .*

*Доказательство.* По формуле полной вероятности вероятность ошибки  $R_f$  при обращении слова  $f_n(x)$  для случайного  $x$  можно расписать как вероятность ошибки на “трудных” входах и на простых входах.

$$\begin{aligned} \Pr_{x,r}[\text{ошибка } R_f] &= \Pr_r[\text{ошибка } R_f \mid \hat{s}(x) < \epsilon] \cdot \Pr_x[\hat{s}(x) < \epsilon] \\ &\quad + \Pr_r[\text{ошибка } R_f \mid \hat{s}(x) \geq \epsilon] \cdot \Pr_x[\hat{s}(x) \geq \epsilon] \\ &\leq 1 \cdot \delta + (1 - \epsilon)^M \cdot 1. \end{aligned}$$

□

**Лемма 2.2.** *Вероятность успеха алгоритма  $R_F$  при обращении слова  $F(\bar{x})$  для случайного  $\bar{x} = x_1, \dots, x_N$  не больше  $N\delta\epsilon + (1 - \delta)^N$ .*

*Доказательство.* Оценим вероятность успеха сверху по формуле полной вероятности:

$$\begin{aligned} \Pr_{\bar{x},r}[\text{успех } R_F] &= \Pr_r[\text{успех } R_F \mid \exists i, \hat{s}(x_i) < \epsilon] \cdot \Pr_{\bar{x}}[\exists i, \hat{s}(x_i) < \epsilon] \\ &\quad + \Pr_r[\text{успех } R_F \mid \forall i, \hat{s}(x_i) \geq \epsilon] \cdot \Pr_{\bar{x}}[\forall i, \hat{s}(x_i) \geq \epsilon] \\ &\leq \epsilon \cdot N\delta + 1 \cdot (1 - \delta)^N. \end{aligned}$$

□

Предположим теперь, что у получившегося алгоритма  $R_f$  вероятность ошибки больше, чем  $1/p(n)$ , т.е.  $R_f$  обращает  $f$  с вероятностью успеха меньше  $1 - \frac{1}{p(n)}$ . Положим  $M = n/\epsilon$ . Тогда второе слагаемое в лемме 2.1 будет порядка  $e^{-n}$ , что при достаточно больших  $n$  меньше, чем  $\frac{1}{2p(n)}$ . Таким образом  $\delta > \frac{1}{2p(n)}$ .

Теперь мы хотим определить  $N$  и  $\epsilon$  так, чтобы вероятность успеха  $R_F$  оказалась меньше, чем  $1/q(n)$ . Выберем  $N = n \cdot p(n)$ , тогда при  $\delta > \frac{1}{2p(n)}$  мы получаем, что второе слагаемое в лемме 2.2 будет порядка  $e^{-n/2}$ :

$$(1 - \delta)^N < \left(1 - \frac{1}{2p(n)}\right)^{n \cdot p(n)} = \left[\left(1 - \frac{1}{2p(n)}\right)^{2p(n)}\right]^{n/2} \approx e^{-n/2}.$$

При достаточно больших  $n$  это меньше, чем  $\frac{1}{2q(n)}$ . Осталось определить  $\epsilon$  так, чтобы и первое слагаемое лемме 2.2 было меньше  $\frac{1}{2q(n)}$ . Например, это достигается при

$$\epsilon = \frac{1}{2N \cdot q(n)} = \frac{1}{2n \cdot p(n) \cdot q(n)}.$$

При таким  $M$ ,  $N$  и  $\epsilon$  получается, что алгоритм  $R_f$  вызовет полиномиальный алгоритм  $R_F$  не более  $M \cdot N = 2n^3 \cdot p^2(n) \cdot q(n)$  раз, т.е.  $R_f$  сам по себе будет полиномиальным.  $\square$

## 2.6. Частичные односторонние функции

Односторонние функции, которые мы определили выше, определены для всех слов длины  $k(n)$ . Можно обобщить это определение на случай частичных функций, которые определены на некотором  $D_n \subset \{0, 1\}^{k(n)}$ . Для того, чтобы такие функции можно было применять для построения криптографических протоколов, мы дополнительно потребуем возможности генерировать *почти* равномерное распределение на  $D_n$ .

**Определение 2.8.** Последовательность распределений вероятностей  $\mu_n$  на множестве двоичных слов называется *полиномиально моделируемой*, если существует полиномиальный вероятностный алгоритм  $K$ , такой, что для всех  $x \in \{0, 1\}^*$

$$\Pr_r[K(1^n, r) = x] = \mu_n(x).$$

**Определение 2.9.** *Статистическим расстоянием* между распределениями вероятностей  $\mu$  и  $\nu$  называется

$$\delta(\mu, \nu) = \max_{A \subset \{0, 1\}^*} |\mu(A) - \nu(A)|.$$

Не сложно показать, что максимум достигается при  $A$  равном  $\{x \mid \mu(x) > \nu(x)\}$  и его дополнению. Таким образом

$$\delta(\mu, \nu) = \frac{1}{2} \sum_x |\mu(x) - \nu(x)|.$$

**Определение 2.10.** Последовательности распределений  $\mu_n$  и  $\nu_n$  называются *статистически неотличимыми*, если статистическое расстояние между ними стремится к нулю быстрее любого обратного полинома от  $n$  при  $n \rightarrow \infty$ .

**Определение 2.11.** Случайные величины  $\alpha_n$  и  $\beta_n$  называются *статистически неотличимыми*, если их распределения  $\mu_n(x) = \Pr[\alpha_n = x]$  и  $\nu_n(x) = \Pr[\beta_n = x]$  статистически неотличимы.

**Определение 2.12.** Распределение  $\mu_n$  называется *доступным*, если оно статистически неотлично от некоторого полиномиально моделируемого распределения  $\nu_n$ .

**Определение 2.13.** Семейство частичных функций  $f_n$  с областями определения  $D_n$  называется *сильно односторонним*, если  $f_n$  полиномиально вычислимо, равномерное распределение на  $D_n$  доступно, и для любого полинома  $q$  и для любого полиномиального вероятностного алгоритма  $R$ ,

$$\Pr_{x \leftarrow D_n, r} [f_n(x) = f_n(R(1^n, f_n(x), r))] < \frac{1}{q(n)}$$

при всех достаточно больших  $n$ . Сильно одностороннее семейство частичных функций  $f_n$  называется *сильно односторонней перестановкой* (one-way permutation, owp), если для всех  $n$  оно является перестановкой своей области определения  $D_n$ .



Аналогичным образом определяются *слабо односторонние функции* и *односторонние функции* для неравномерного противника.

**Пример 2.3** (Предположительно сильно односторонние частичные функции).

1. *Функция Рабина*. Функция  $f_n$  определена на словах вида  $xy$  длины  $4n$ , где  $|x| = |y| = 2n$ . При этом  $x$  и  $y$  интерпретируются как  $2n$ -битовые числа, удовлетворяющих следующим требованиям:

- (a)  $y = p \cdot q$ , где  $p$  и  $q$  простые  $n$ -битовые числа вида  $4k + 3$ ;
- (b)  $x = z^2 \bmod y$  для некоторого  $z$ , взаимно простого с  $y$ .

Значение функции на  $xy$  равно конкатенации слов  $x^2 \bmod y$  и  $y$ .

2. *Функция RSA*. Функция RSA является обобщением функции Рабина. Она определена на словах вида  $xuz$ , где  $x$ ,  $y$  и  $z$  имеют длину  $2n$  и интерпретируются как двоичные записи чисел, удовлетворяющие следующим требованиям:

- (a)  $y = p \cdot q$ , где  $p$  и  $q$  простые  $n$ -битовые числа;
- (b)  $x \in [1, pq - 1]$  и взаимно просто с  $y$ ;
- (c)  $z$  взаимно просто  $\phi(pq) = (p - 1) \cdot (q - 1)$ .

Значение функции на  $xuz$  равно конкатенации слов  $(x^z \bmod y)$ ,  $y$  и  $z$ .

3. *Дискретная экспонента*. Функция определена на словах вида  $xuz$ , где  $x$ ,  $y$  и  $z$  имеют длину  $n$  и соответствующие числа удовлетворяют следующим требованиям:

- (a)  $y$  —  $n$ -битовое простое число,
- (b)  $x \in [2, y - 1]$ , порождает всю мультипликативную группу вычетов по модулю  $y$  (т.е. любой ненулевой вычет является степенью  $x$ ),
- (c)  $z \in [1, y - 1]$ .

Значение функции на  $xuz$  равно конкатенации слов  $x$ ,  $y$  и  $(x^z \bmod y)$ . Обращение этой функции — является дискретным логарифмированием.

Более подробно об этих примерах см. [1].

**Определение 2.14.** Частичная функция  $f_n : D_n \rightarrow \{0, 1\}^{l(n)}$ , где  $D_n \subset \{0, 1\}^{k(n)}$  называется *проверяемой*, если по  $n$ , любому слову  $x$  длины  $k(n)$  и любому слову  $y$  из множества значений  $f_n$  можно за полиномиальное время проверить, верно ли, что  $f_n$  определена на  $x$  и её значение на  $x$  равно  $y$ .

*Замечание 2.2.* Неизвестно, является ли функция Рабина проверяемой, т.к. неясно, как за полиномиальное время проверить, является ли данное число квадратичным вычетом по составному модулю.

### 3. Генераторы псевдослучайных чисел

#### 3.1. Вычислительно неотличимые случайные величины

**Определение 3.1** (Для неравномерного противника). Случайные величины  $\alpha_n$  и  $\beta_n$ , зависящие от натурального параметра  $n$ , со значениями в множестве слов некоторой длины  $l(n)$  называются *вычислительно неотличимыми*, если для любой последовательности схем

$C_0, C_1, \dots, C_n, \dots$  размера  $\text{poly}(n)$  (с  $l(n)$  входами и одним выходом) вероятность событий  $C_n(\alpha_n) = 1$  и  $C_n(\beta_n) = 1$  отличаются на пренебрежимо малую величину, т.е.

$$|\Pr[C_n(\alpha_n) = 1] - \Pr[C_n(\beta_n) = 1]| < \epsilon_n,$$

где  $\epsilon_n$  убывает быстрее любого обратного полинома. Схема  $C_n$  в этом контексте называется *тестом* и мы говорим, что случайная величина  $\alpha_n$  проходит тест  $C_n$ , если  $C_n(\alpha_n) = 1$ . Таким образом, мы требуем, чтобы  $\alpha_n$  и  $\beta_n$  проходили любые тесты полиномиального размера с приблизительно равной вероятностью.

**Определение 3.2** (Для равномерного противника). Случайные величины  $\alpha_n$  и  $\beta_n$ , зависящие от натурального параметра  $n$ , со значениями в множестве слов некоторой длины  $l(n)$  называются *вычислительно неотличимыми*, если для любого вероятностного полиномиального алгоритма  $T$  вероятность событий  $T(1^n, \alpha_n) = 1$  и  $T(1^n, \beta_n) = 1$  отличаются на пренебрежимо малую величину, соответственно

$$|\Pr[T(1^n, \alpha_n) = 1] - \Pr[T(1^n, \beta_n) = 1]| < \epsilon_n,$$

где  $\epsilon_n$  убывает быстрее любого обратного полинома. Алгоритм  $T$  в этом контексте называется *тестом* и мы говорим, что случайная величина  $\alpha_n$  проходит тест  $T$ , если  $T(1^n, \alpha_n) = 1$ .

*Замечание 3.1.* Если  $\alpha_n$  и  $\beta_n$  статистически неотличимы, то они и вычислительно неотличимы (например, для неравномерного противника), поскольку разность вероятностей  $\alpha_n$  и  $\beta_n$  в множество  $\{x \mid C_n(x)\}$ , задаваемое тестом  $C_n$ , не превосходит статистического расстояния между  $\alpha_n$  и  $\beta_n$ .

**Лемма 3.1** (Свойства вычислительной неотличимости).

1. Отношение вычислительной неотличимости рефлексивно, симметрично и транзитивно.
2. Для неравномерного противника: вычислительно неотличимые последовательности случайных величин  $\alpha_n$  и  $\beta_n$  вычислительно неотличимы и вероятностными тестами полиномиального размера. Это означает, что для любой последовательности  $T_n$  вероятностных схем полиномиального от  $n$  размера с  $l(n)$  входами, вероятности событий  $T_n(\alpha_n) = 1$  и  $T_n(\beta_n) = 1$  приблизительно равны.
3. Если  $\alpha_n$  и  $\beta_n$  вычислительно неотличимы, а  $C_n$  — последовательность вероятностных схем полиномиального размера с  $l(n)$  входами, то и случайные величины  $C_n(\alpha_n)$  и  $C_n(\beta_n)$  вычислительно неотличимы. Аналогично для равномерного противника.
4. Пусть случайные величины  $\alpha_n$ ,  $\beta_n$  и  $\gamma_n$  имеют совместное распределение. И пусть для любой последовательности значений  $c_n$  случайной величины  $\gamma_n$  случайные величины  $(\alpha_n \mid \gamma_n = c_n)$  и  $(\beta_n \mid \gamma_n = c_n)$  вычислительно неотличимы. Тогда и  $\alpha_n \gamma_n$  и  $\beta_n \gamma_n$  вычислительно неотличимы.

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

*Доказательство.*

1-2. Очевидно.

3. Пусть дана последовательность тестов  $T_n$  позволяет отличать  $C_n(\alpha_n)$  и  $C_n(\beta_n)$ . Тогда последовательность схем  $D_n(x) = T_n(C_n(x))$  будет вероятностным тестом полиномиального размера для случайных величин  $\alpha_n$  и  $\beta_n$ .

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

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

по случайно выбранному  $c$  (в соответствии с распределением случайной величины  $\gamma_n$ ). Поэтому для бесконечно многих  $n$  найдётся  $c = c_n$  в множестве значений  $\gamma_n$ , для которой разность вероятности не меньше  $\epsilon$ . Если “зашить”  $c_n$  в схему  $T_n$ , то получится полиномиальный тест, различающий  $(\alpha_n \mid \gamma_n = c_n)$  и  $(\beta_n \mid \gamma_n = c_n)$ .

□

### 3.2. Генераторы псевдослучайных чисел

**Определение 3.3.** Пусть даны многочлены  $k(n)$  и  $l(n)$  такие, что  $l(n) > k(n)$  для всех  $n$ . Генератором псевдослучайных чисел типа  $k(n) \rightarrow l(n)$  будем называть семейство функций  $G_n : \{0, 1\}^{k(n)} \rightarrow \{0, 1\}^{l(n)}$ , удовлетворяющее следующим условиям.

1. Семейство  $G_n$  вычислимо за полиномиальное от  $n$  время.
2. (Надежность генератора ПСЧ.) Случайная величина  $G_n(s)$  для равномерного случайного  $s$  вычислительно неотличима от случайной величины равномерно распределённой на всех словах длины  $l(n)$ .

**Определение 3.4.** Генератор псевдослучайных чисел типа  $k(n) \rightarrow \infty$  будем называть семейство отображений  $G_n : \{0, 1\}^{k(n)} \rightarrow \{0, 1\}^*$ , удовлетворяющее следующим требованиям.

1. Существует алгоритм, который по слову  $s$  и натуральному числу  $l$  вычисляет элемент последовательности  $G_n(s)$  с номером  $l$  за время, полиномиальное от  $|s| + l$ .
2. Случайная величина  $G_n(s)$  вычислительно неотличима от равномерно распределённой бесконечной последовательности нулей и единиц. (Это означает, что для любого полинома  $p(n)$  первые  $p(n)$  битов  $G_n(s)$  вычислительно неотличимы от случайной величины равномерно распределённой на всех словах длины  $p(n)$ .)

*Замечание 3.2.* По генератору ПСЧ типа  $k(n) \rightarrow \infty$  можно построить генератор ПСЧ типа  $k(n) \rightarrow l(n)$  для любого полинома  $l(n)$  (нужно взять первый  $l(n)$  битов).

**Теорема 3.1.** Если существует генератор ПСЧ  $G_n$  типа  $k(n) \rightarrow l(n)$ , то существуют и односторонние функции.

*Доказательство.* Действительно, можно рассмотреть сам генератор  $G_n$  как слабо одностороннюю функцию. Давайте покажем, что никакая последовательность схем полиномиального размера  $C_n$  не обращает  $G_n$  с вероятностью успеха более  $3/4$  для бесконечно многих  $n$ . Предположим, что такая последовательность существует. Тогда можно рассмотреть следующий полиномиальный тест для последовательностей длины  $l(n)$ : для входа  $y$  проверяем, что  $G_n(C_n(y)) = y$ . Если это верно (т.е. обращение произошло удачно), то выдаём 1, а иначе 0. Вероятность того, что  $G_n$  пройдёт такой тест не менее  $3/4$ . При этом равномерно распределённая на  $l(n)$  случайная величина пройдёт этот тест с вероятностью не более  $1/2$  (т.к. размер образа  $G_n$  не более  $1/2$ ). □

Обратное утверждение тоже верно.

**Теорема 3.2 ([4]).** Если существует односторонняя функция, то существует и генератор псевдослучайных чисел типа  $n \rightarrow \infty$ .

Мы же докажем более простое утверждение.

**Теорема 3.3.** Если существует односторонняя перестановка, то существует и генератор псевдослучайных чисел типа  $n \rightarrow \infty$ .

В дальнейшем мы будем говорить про *полиномиального противника* подразумевая под этим две возможных формулировки: полиномиальный вероятностный алгоритм и семейство схем полиномиального размера.

### 3.3. Трудный бит

**Определение 3.5.** Для пары совместно распределённых случайных величин  $(\beta_n, \gamma_n)$ , где  $\beta_n$  распределена на  $\{0, 1\}$  будем говорить, что  $\beta_n$  является *вычислительно трудной* относительно  $\gamma_n$ , если для любого полиномиального противника  $B$

$$|\Pr[B(\gamma_n) = \beta_n] - \frac{1}{2}| < \frac{1}{p(n)}$$

для любого полинома  $p$  для достаточно больших  $n$ .

**Теорема 3.4.** Случайная величина  $\beta_n$  вычислительно трудна относительно  $\gamma_n$  тогда и только тогда, когда случайная величина  $\beta_n \gamma_n$  вычислительно неотличима от  $r_n \gamma_n$ , где  $r_n$  — это равномерно распределённая на  $\{0, 1\}$  случайная величина.

*Доказательство.*

$\Leftarrow$  Пусть существует противник  $B$ , который для бесконечного числа  $n$  предсказывает  $\beta_n$  по  $\gamma_n$  с хорошей вероятностью, т.е.

$$\Pr[B(\gamma_n) = \beta_n] \geq \frac{1}{2} + \epsilon_n,$$

где  $\epsilon_n$  — обратный полином. Используя противника  $B$  построим противника  $A$ , который различает  $\beta_n \gamma_n$  и  $r_n \gamma_n$ .

$$A(x, y) = \begin{cases} 1, & B(y) = x, \\ 0, & B(y) \neq x. \end{cases}$$

Тогда  $\Pr[A(\beta_n \gamma_n) = 1] \geq \frac{1}{2} + \epsilon_n$ , а  $\Pr[A(r_n \gamma_n) = 1] = \frac{1}{2}$ .

$\Rightarrow$  Пусть существует противник  $A$ , который для бесконечного числа  $n$  различает  $\beta_n \gamma_n$  и  $r_n \gamma_n$  с хорошей вероятностью, т.е.

$$\Pr[A(\beta_n \gamma_n) = 1] - \Pr[A(r_n \gamma_n) = 1] \geq \epsilon_n,$$

где  $\epsilon_n$  — обратный полином. Построим противника  $B$ , который предсказывает  $\beta_n$  по  $\gamma_n$ .

$$B(x) = \begin{cases} r, & A(0x) = 0, A(1x) = 0, \\ 1, & A(0x) = 0, A(1x) = 1, \\ 0, & A(0x) = 1, A(1x) = 0, \\ r, & A(0x) = 1, A(1x) = 1, \end{cases}$$

где  $r$  означает случайный бит.

Покажем, что

$$\Pr[B(\gamma_n) = \beta_n] = \frac{1}{2} + \Pr[A(\beta_n \gamma_n) = 1] - \Pr[A(r_n \gamma_n) = 1] \geq \frac{1}{2} + \epsilon_n.$$

Давайте докажем это равенство для фиксированного  $\gamma_n = x$ :

$$\Pr[B(\gamma_n) = \beta_n \mid \gamma_n = x] = \frac{1}{2} + \Pr[A(\beta_n \gamma_n) = 1 \mid \gamma_n = x] - \Pr[A(r_n \gamma_n) = 1 \mid \gamma_n = x].$$

Отсюда по формуле полной вероятности получается требуемое равенство. Для того, чтобы убедиться, что это равенство верно, нужно подставить все четыре возможные варианта из определения  $B$  и проверить, что равенство выполняется. Например, в первом случае вероятность слева будет равна  $1/2$ , т.к.  $B$  возвращает случайный бит, а справа обе вероятности равны нулю. Для второго случая вероятность слева будет равна первой вероятности справа, а последняя  $= 1/2$ .

*Замечание 3.3.* В случае неравномерного противника данная конструкция даёт вероятностную схему, которую, как описано выше можно переделать в детерминированную. В случае равномерного противника нам потребовалось бы также фиксировать внутренние биты вероятностного алгоритма.

□

**Определение 3.6.** Для односторонней перестановки  $f_n : D_n \rightarrow D_n$ , где  $D_n \subset \{0, 1\}^{k(n)}$ , будем называть *трудным битом* такую полиномиально вычислимую функцию  $h_n : D_n \rightarrow \{0, 1\}$ , для которой случайная величина  $h_n(U(D_n))$  трудна для  $f_n(U(D_n))$ .

**Утверждение 3.1.** Пусть  $f_n : D_n \rightarrow D_n$  — односторонняя перестановка с трудным битом  $h_n : D_n \rightarrow \{0, 1\}$ . Тогда случайная величина  $h_n(x)f_n(x)$  вычислительно неотличима от  $rx$ , где  $x \leftarrow U(D_n)$ , а  $r \leftarrow U_1$ .

*Доказательство.* Заметим, что  $f_n(x)$  распределено так же, как  $x$  (важно, что  $f_n$  — перестановка). Поэтому  $rf_n(x)$  распределено так же, как  $rx$ . Осталось применить теорему 3.4 для  $rf_n(x)$  и  $h_n(x)f_n(x)$ . □

Это конструкцию можно итерировать. Например,  $h_n(x)h_n(f_n(x))f_n(f_n(x))$  позволяет получить два бита. Заметим, что если случайная величина  $h_n(x)h_n(f_n(x))f_n(f_n(x))$  отличима от  $rh_n(f_n(x))f_n(f_n(x))$ , то отличимы  $h_n(x)f_n(x)$  и  $rf_n(x)$  (первые можно получить из вторых). Продолжая рассуждение получаем, что  $h_n(x)h_n(f_n(x))f_n(f_n(x))$  неотличима от  $rr'x$ , где  $r, r' \leftarrow U_1$ .

**Теорема 3.5.** Пусть  $f_n : D_n \rightarrow D_n$  — односторонняя перестановка с трудным битом  $h_n : D_n \rightarrow \{0, 1\}$ . Тогда случайная величина

$$h_n(x)h_n(f_n(x)) \dots h_n(f^{(p(n))}(x))f_n^{(p(n)+1)}(x)$$

вычислительно неотличима от случайной величины

$$r_1 r_2 \dots r_{p(n)} x,$$

где  $r_1, \dots, r_{p(n)} \leftarrow U_1$ ,  $x \leftarrow U(D_n)$ .

**Следствие 3.1.**  $G_n(x) = h_n(x)h_n(f_n(x)) \dots h_n(f^{(p(n))}(x))f_n^{(p(n)+1)}(x)$  является надёжным генератором псевдослучайных чисел.

*Доказательство.* Доказательство „гибридным“ методом.

$$\begin{array}{rcccccc}
T_0 = & h_n(x) & h_n(f_n(x)) & \dots & h_n(f^{(p(n)-1)}_n) & f^{(p(n))}_n(x) \\
T_1 = & r_1 & h_n(x) & \dots & h_n(f^{(p(n)-2)}_n) & f^{(p(n)-1)}_n(x) \\
T_2 = & r_1 & r_2 & \dots & h_n(f^{(p(n)-3)}_n) & f^{(p(n)-2)}_n(x) \\
\vdots & \vdots & \vdots & \dots & \vdots & \vdots \\
T_{p(n)} = & r_1 & r_2 & \dots & r_{p(n)} & x
\end{array}$$

Пусть взломщик  $B$  отличает  $T_0$  от  $T_{p(n)}$ , т.е.  $\Pr[B(T_{p(n)}) = 1] - \Pr[B(T_0) = 1] \geq \epsilon_n$  для бесконечного числа  $n$  и обратного полинома  $\epsilon_n$ . Тогда существует такое  $i$ , что

$$\Pr[B(T_i) = 1] - \Pr[B(T_{i+1}) = 1] \geq \epsilon_n/p(n).$$

Т.е. мы научились отличать

$$\begin{array}{cccccc}
r_1 & \dots & r_i & h_n(x) & h_n(f_n(x)) & \dots & h_n(f^{(p(n)-i-1)}_n) & f^{(p(n)-i)}_n(x) \\
r_1 & \dots & r_i & r_{i+1} & h_n(x) & \dots & h_n(f^{(p(n)-i-2)}_n) & f^{(p(n)-i-1)}_n(x)
\end{array}$$

Используя противника, который отличает эти две случайные величины мы можем отличать  $h_n(x)f_n(x)$  от  $rx$  — по этим случайным величинам можно детерминированным алгоритмом построить соответствующие значения  $T_i$  и  $T_{i+1}$ , что противоречит утверждению 3.1.

*Замечание 3.4.* В этом доказательстве мы воспользовались тем, что наш у нас неравномерный противник, т.е. для схема, т.к. мы в эту схему зашили число  $i$ . В случае с алгоритмами можно взять случайное  $i$  и доказать, что с хорошей вероятностью оно подойдёт.

□

**Теорема 3.6** (Голдрейх, Левин). Пусть  $f_n : D_n \rightarrow D_n$  — односторонняя перестановка,  $D_n \subseteq \{0, 1\}^{k(n)}$ . Рассмотрим две функции:  $g_n : D_n \times \{0, 1\}^{k(n)} \rightarrow D_n \times \{0, 1\}^{k(n)}$  и  $h_n : D_n \times \{0, 1\}^{k(n)} \rightarrow \{0, 1\}$ , такие что

$$g_n(xy) = f_n(x)y, \quad h_n(x, y) = x \odot y = \bigoplus_{i=1}^{k(n)} x_i y_i = \sum_{i=1}^{k(n)} x_i y_i \pmod{2}.$$

Тогда  $g_n$  — односторонняя перестановка с трудным битом  $h_n$ .

**Определение 3.7.** Код Уолша-Адамара — это код исправляющий ошибки  $WH : \{0, 1\}^k \rightarrow \{0, 1\}^{2^k}$ , определяющий следующим соотношением  $WH(x) = (x \odot y)_{y \in \{0, 1\}^k}$ .

Код Уолша-Адамара не очень удобен в практических применениях, т.к. он удлиняет строки в экспоненту раз. Однако он обладает одним очень хорошим свойством.

**Утверждение 3.2.** Код Уолша-Адамара имеет расстояние  $2^{k-1}$ .

*Доказательство.* Пусть  $x_i \neq y_i$ . Рассмотрим  $r$  и  $r^{\oplus i}$ , где  $r, r^{\oplus i} \in \{0, 1\}^k$  и отличаются только в бите  $i$ . Тогда либо  $x \odot r$  отличается от  $x \odot r^{\oplus i}$ , либо  $y \odot r$  отличается от  $y \odot r^{\oplus i}$ , а значит коды  $x$  и  $y$  отличаются либо в координате  $r$ , либо в координате  $r^{\oplus i}$ . Т.к. все  $\{0, 1\}^k$  можно разбить на пары, различающиеся в одном бите, то все пары строк имеют коды, отличающиеся ровно в половине всех битов. □

**Лемма 3.2.** Пусть  $s \in \{0, 1\}^{2^m}$  и  $\Pr_i[s_i \neq WH(x)] \leq 1/2 - \epsilon$  (в терминах расстояния Хемминга  $\Delta(WH(x), s) \leq (1/2 - \epsilon)2^m$ ). Существует вероятностный алгоритм  $A^s$  со временем работы  $\text{poly}(m, \frac{1}{\epsilon})$ , который выдаёт список  $L$  слов длины  $m$  такой, что  $x \in L$  с вероятностью не менее  $1/2$  (алгоритм получает оракульный доступ к строке  $s$ ).

*Доказательство теоремы Голдрейха-Левина.* Функция  $g_n$  является перестановкой. Легко показать, что если противник взламывает  $g_n$ , то он взламывает и  $f_n$ , т.е.  $g_n$  является односторонней перестановкой.

Теперь нужно показать, что  $h_n$  является трудным битом  $g_n$ . Пусть существует неравномерный противник  $B$  (в дальнейшем будем предполагать, что противник — это всегда семейство схем), который предсказывает  $h_n$ , т.е.

$$\Pr_{x \leftarrow U(D_n), y \leftarrow U_{k(n)}} [B(f(x)y) = x \odot y] \geq 1/2 + \epsilon_n$$

для бесконечного числа  $n$  и обратного полинома  $\epsilon_n$ . Построим противника  $C$ , который обращает  $f_n$ , т.е.

$$\Pr_{x \leftarrow U(D_n)} [C(f_n(x)) = x] \geq \epsilon_n/4.$$

Противник  $C$  будет использовать алгоритм декодирования списком кода Уолша-Адамара из леммы 3.2 и при обращении к биту  $y$  выдаём значение  $B(f(x)y)$ . В списке  $L$ , который возвращает алгоритм  $A$  ищем  $z$  такое, что  $f_n(z) = f_n(x)$ . Если такое  $z$  нашлось, то выдаём его, иначе выдаём любую строку.

Пусть  $M \subseteq D_n$  и  $x \in M \iff \Pr_{y \leftarrow U_{k(n)}} [B(f_n(x)y) = x \odot y] \geq 1/2 + \epsilon_n/2$ . Давайте покажем, что  $\Pr_{x \leftarrow U(D_n)} [x \in M] \geq \epsilon_n/2$ . Пусть это не так, тогда

$$\begin{aligned} \Pr_{x,y} [B(f_n(x)y) = x \odot y] &= \Pr_{x,y} [B(f_n(x)y) = x \odot y \mid x \in M] \cdot \Pr_x [x \in M] \\ &\quad + \Pr_{x,y} [B(f_n(x)y) = x \odot y \mid x \notin M] \cdot \Pr_x [x \notin M] \\ &< 1 \cdot \epsilon_n/2 + (1/2 + \epsilon_n/2) \cdot 1 < 1/2 + \epsilon_n. \end{aligned}$$

Заметим, что  $\Pr_x [C(f_n(x)) = x \mid x \in M] \geq 1/2$ , т.к. с вероятностью не менее  $1/2$  в списке будет искомый элемент. Поэтому получаем

$$\Pr_x [C(f_n(x)) = x] \geq \Pr_x [C(f_n(x)) = x \mid x \in M] \cdot \Pr_x [x \in M] \geq \frac{1}{2} \cdot \frac{\epsilon_n}{2} = \frac{\epsilon_n}{4}.$$

□

*Доказательство леммы 3.2.* Будем рассматривать код Уолша-Адамара как таблицу истинности некоторой функции  $f : \{0, 1\}^m \rightarrow \{0, 1\}$ . Пусть  $WH(x)$  соответствует  $f$ , а кодовое слово  $s$  — некоторой функции  $\tilde{f}$ . Таким образом для восстановления  $x$  нам нужно вычислить  $f$  на входах  $100 \dots 0, 010 \dots 0, 001 \dots 0, \dots$ . Действительно,

$$\begin{aligned} x_1 &= f(100 \dots 0) \\ x_2 &= f(010 \dots 0) \\ &\vdots \\ x_n &= f(000 \dots 1) \end{aligned}$$

Используя линейность  $f$  мы можем вычислять  $f(z)$  следующим образом: выберем случайный  $r$  и вычислим  $f(z) = f(r) \oplus f(z \oplus r)$ . Проблема в том, что доступа к  $f$  у нас нет, а есть доступ к  $\tilde{f}$ , про которую известно  $\Pr_y [f(y) \neq \tilde{f}(y)] \leq 1/2 - \epsilon$ . Используя идею с линейностью мы можем попробовать вычислять  $f(z) = f(r) \oplus \tilde{f}(z \oplus r)$  (добавление  $r$  позволяет вычислять  $\tilde{f}$  в случайной точке, в то время как значение  $\tilde{f}$  на строках вида  $00 \dots 010 \dots 00$  может быть неправильным). Если бы ошибка в  $\tilde{f}$  случалась с вероятностью менее  $1/4 - \epsilon$ , то суммарная ошибка при таком вычислении  $f(z)$  была бы не более  $1/2 - 2\epsilon$ . Тогда мы могли бы её амплифицировать и получить  $f(z)$  с хорошей вероятностью. Однако, ошибка в  $\tilde{f}$  случается с большей вероятностью.

**Идея.** Если бы у нас был доступ к истинному значению, то тогда бы тоже всё сработало  $f(z) = f(r) \oplus \tilde{f}(z \oplus r)$ . Предположим, что мы всё же умеем вычислять  $f(r)$  вместо  $\tilde{f}(r)$ . Тогда мы можем вычислить  $f(z)$  следующим образом: выберем случайные  $r_1, r_2, \dots, r_N$  и вычислим  $\text{maj}_i\{f(r_i) \oplus \tilde{f}(z \oplus r_i)\}$ . С хорошей вероятностью полученное значение будет совпадать с  $f(z)$ . Чтобы оценить это давайте вспомним следующий факт из теории вероятностей.

*Замечание 3.5.* Неравенство Чебышёва (закон больших чисел для 2-независимых случайных величин). Пусть  $X_1, X_2, \dots, X_N$  — одинаково распределённые попарно-независимые Бернулиевские случайные величины,  $\forall i, \Pr[X_i = 1] = p$ , тогда

$$\Pr\left[\left|\frac{\sum_i X_i}{N} - p\right| \geq \delta\right] \leq \frac{1}{\delta^2 N}.$$

Для применения этого неравенства нам нужны попарно независимые случайные величины. Пусть  $N = 2^k - 1$ , конкретное значение для  $N$  мы определим позже. Выберем  $t_1, t_2, \dots, t_k \leftarrow U_m$  — независимые случайные строки. Из этих случайных строк можно сгенерировать  $N$  попарно независимых следующим образом:

$$\forall J \subseteq \{1, \dots, k\}, J \neq \emptyset, r_J = \bigoplus_{i \in J} t_i.$$

**Утверждение 3.3.** Если  $J_1 \neq J_2$ , то  $r_{J_1}$  и  $r_{J_2}$  будут независимыми.

**Следствие 3.2.**  $\{r_J\}_J$  — 2-независимые случайные величины.

Теперь опишем алгоритм декодирования кода Уолша-Адамара используя нашу идею с вычислением  $f(r)$ . Сгенерируем независимые  $t_1, t_2, \dots, t_k \leftarrow U_m$  и по ним определим  $r_J = \bigoplus_{i \in J} t_i$ ,  $J \subset [n], J \neq \emptyset$ . Так как доступа к  $f$  у нас нет, то вычислить  $f(r)$  мы не можем. Вместо этого мы переберём все значения для  $f(r)$ . Если бы все  $r$  были независимы, то нам пришлось бы перебрать  $2^N$  различных значений. Однако, наши  $r$  получаются из  $k$  строк  $t_i$ . Поэтому достаточно перебрать значения  $f$  на  $t_i$ . Пусть  $\forall i, a_i = f(t_i)$ , тогда

$$f(r^J) = f\left(\bigoplus_{j \in J} t_j\right) = \bigoplus_{j \in J} f(t_j) = \bigoplus_{j \in J} a_j.$$

Таким образом, алгоритм декодирования для всех возможных  $a_1, \dots, a_k \in \{0, 1\}$  добавит в список  $L$  строку  $x_1, \dots, x_m$ , полученную по следующей процедуре:

$$x_i = \text{maj}_J \left\{ \bigoplus_{j \in J} a_j \oplus \tilde{f}((0 \cdots 1 \cdots 0) \oplus r^J) \right\},$$

Для успеха  $x_i$  должен быть верен с вероятностью не менее  $1 - \frac{1}{10^m}$  (тогда весь  $x$  мы угадаем с вероятностью  $\geq 9/10$ ). Определим случайные величины  $\{X_i^J\}$  такие, что

$$X_i^J = 1 \iff \tilde{f}((0 \cdots 1 \cdots 0) \oplus r^J) = f((0 \cdots 1 \cdots 0) \oplus r^J).$$

Пусть  $p = \Pr[X_i^J = 1] \geq 1/2 + \epsilon$ . По закону больших чисел вероятность того, что при вычислении  $\text{maj}$  мы получим более половины неправильных значений  $\tilde{f}$  не более  $\frac{1}{\epsilon^2 N}$ .



$$\begin{aligned}
\Pr[x_i \text{ ошибочный}] &= \Pr\left[\sum_J \frac{X_i^J}{N} \leq \frac{1}{2}\right] \\
&= \Pr\left[\sum_J \frac{X_i^J}{N} - \left(\frac{1}{2} + \epsilon\right) \leq -\epsilon\right] \\
&\leq \Pr\left[\left|\sum_J \frac{X_i^J}{N} - \left(\frac{1}{2} + \epsilon\right)\right| \geq \epsilon\right] \\
&\leq \Pr\left[\left|\sum_J \frac{X_i^J}{N} - p\right| \geq \epsilon\right] \leq \frac{1}{\epsilon^2 N}.
\end{aligned}$$

Таким образом, мы требуем, чтобы  $\frac{1}{\epsilon^2 N} \leq \frac{1}{10m}$ , следовательно  $N \geq \frac{10m}{\epsilon^2}$ .  $\square$

## 4. Протоколы шифрования с секретным ключом

**Определение 4.1.** Пусть дана доступная случайная величина  $d_n$ . *Одноразовый протокол с секретным ключом* задаётся двумя вероятностными полиномиальными алгоритмами  $E(d, x)$  и  $D(d, m)$ , такими, что  $D(d, E(d, x)) = x$ . Будем говорить, что этот протокол *надёжный*, если для некоторого полинома  $l$  и любых последовательностей строк  $x_n$  и  $y_n$  длины  $l(n)$  случайные величины  $E(d_n, x_n)$  и  $E(d_n, y_n)$  вычислительно неотличимы.

Такой протокол мы уже можем построить. Возьмём генератор псевдослучайных чисел  $G : \{0, 1\}^n \rightarrow \{0, 1\}^{l(n)}$ . Возьмём  $d_n = U_n$  и для  $x \in \{0, 1\}^{l(n)}$  определим

$$\begin{aligned}
E(d, x) &= G(d) \oplus x, \\
D(d, m) &= G(d) \oplus m.
\end{aligned}$$

Если бы противник научился отличать случайные величины  $E(d_n, x_n)$  и  $E(d_n, y_n)$ , то он также научился бы отличать одну из этих случайных величин от равномерного распределения  $U_{l(n)}$ . Пусть он умеет отличать  $E(d_n, x_n)$  от  $U_{l(n)}$ . Тогда мы можем зашифровать  $x_n$  в схему и таким образом научиться отличать образ  $G(d_n)$  от  $U_{l(n)}$ .

**Определение 4.2.** Пусть  $S_n \subseteq \{0, 1\}^{l(n)}$  и для любого  $s \in S_n$  определена функция  $f_n^s : \{0, 1\}^n \rightarrow \{0, 1\}^n$ . Множество функций  $\{f_n^s\}_{s \in S_n}$  называется *семейством псевдослучайных функций*, если выполняются следующие свойства.

1. Существует полиномиальный алгоритм, который по  $(s, x)$  вычисляет  $f_n^s(x)$ .
2. Распределение  $U(S_n)$  доступно.
3. (*слабая надёжность*) Для любого полинома  $p$  и для любого набора различных строк  $t_1, \dots, t_{p(n)} \in \{0, 1\}^n$  случайная величина  $f_n^s(t_1)f_n^s(t_2) \cdots f_n^s(t_{p(n)})$  вычислительно неотличима от  $U_{np(n)}$ , где  $s \leftarrow U(S_n)$ .
- 3'. (*сильная надёжность*) Для любого полинома  $p$ , любого семейства схем полиномиального размера  $\{C_i\}_{i=1}^{p(n)-1}$  и любого  $t_1 \in \{0, 1\}^n$  определим случайные величины  $\{y_i\}_{i=1}^{p(n)}$

следующим образом:

$$\begin{aligned}
s &\leftarrow U(S_n), \\
y_1 &= f_n^s(t_1), \\
t_2 &= C_1(t_1, y_1), \\
y_2 &= f_n^s(t_2), \\
t_3 &= C_2(t_1, y_1, y_2), \\
y_3 &= f_n^s(t_3), \\
&\vdots
\end{aligned}$$

Тогда случайная величина  $y_1 y_2 \cdots y_{p(n)}$  вычислительно неотличима от  $U_{np(n)}$  (при условии, что все  $t_i$  различны).

**Теорема 4.1.** *Если существует генератор псевдослучайных чисел, то существует и семейство псевдослучайных функций.*

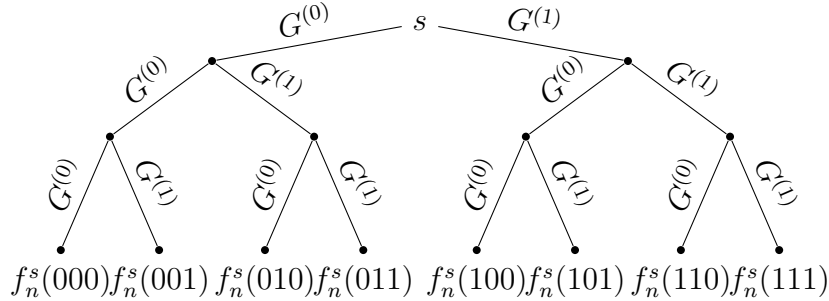
*Доказательство.* Пусть  $G_n : \{0, 1\}^n \rightarrow \{0, 1\}^{2n}$ . Мы построим семейство псевдослучайных функций  $\{f_n^s\}_{s \in S_n}$ , где  $S_n = \{0, 1\}^n$ ,  $f_n^s : \{0, 1\}^n \rightarrow \{0, 1\}^n$ . Для этого определим функции  $G_n^{(0)}, G_n^{(1)} : \{0, 1\}^n \rightarrow \{0, 1\}^n$  исходя из следующего соотношения:

$$G_n(r) = G_n^{(0)}(r) G_n^{(1)}(r).$$

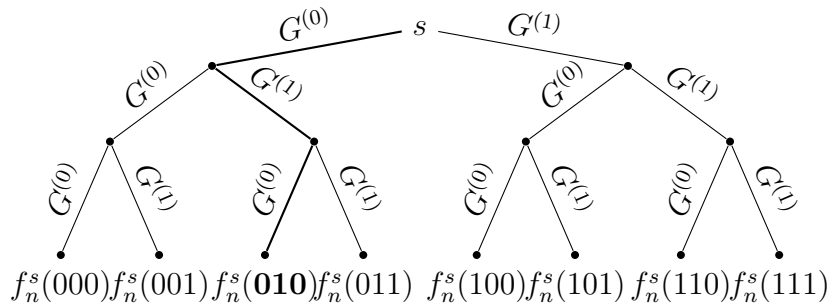
Тогда

$$f_n^s(x) = G_n^{(x_n)}(\cdots G_n^{(x_2)}(G_n^{(x_1)}(s)) \cdots).$$

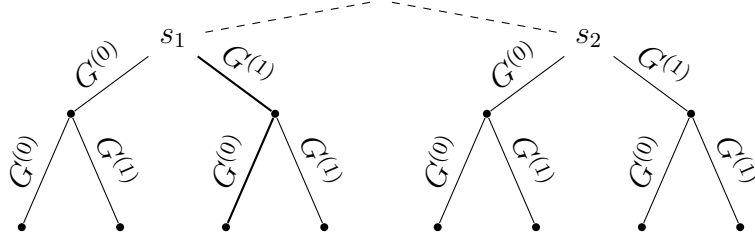
Можно представить вычисление  $f_n^s$  на всевозможных  $x$  в виде бинарного дерева: в корне записано число  $s$ ; если в вершине записано число  $z$ , то в его наследниках будут записаны  $G_n^{(0)}(z)$  и  $G_n^{(1)}(z)$ ; в листьях дерева будут все возможные  $f_n^s(x)$ .



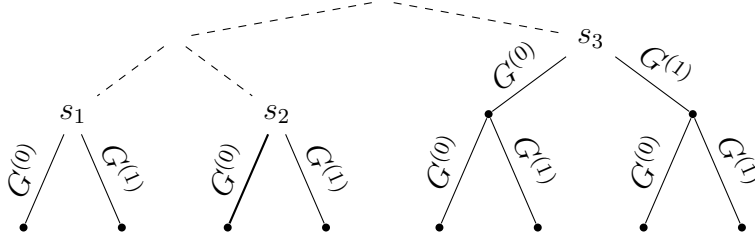
Будем доказывать гибридным методом. Для этого нам нужно построить последовательность распределений, которые постепенно сходятся к равномерному. Изначальное распределение задаёт дерево  $T_0$ , в корне которого выбирается  $s$ , а дальше вычисление происходит детерминировано. Каждое  $t_i$  задаёт некоторый путь от корня к листьям. Мы будем модифицировать исходное дерево следующим образом: для каждой вершины на пути соответствующему  $t_1$  мы будем удалять эту вершину из дерева и заменять её сыновей на случайные строки из  $U(S_n)$ .



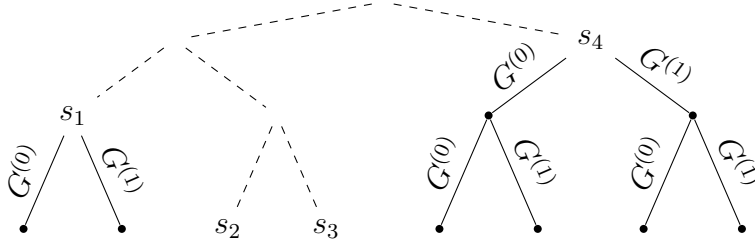
Для примера  $t_1 = 010$ .



Удаляем корневую вершину и заменяем её сыновей на  $s_1, s_2 \leftarrow U(S_n)$



Удаляем следующую вершину и заменяем сыновей на случайные элементы из  $U(S_n)$ .



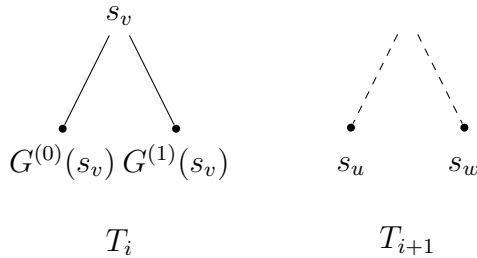
Удаляем последнюю вершину и заменяем сыновей на случайные элементы из  $U(S_n)$ .

В конце этого процесса в вершине, которая раньше соответствовала  $t_1$ , будет некоторый  $s_j \leftarrow U(S_n)$ . Мы повторим этот процесс для всех  $t_i$ . Таким образом получится последовательность из  $n \cdot p(n) + 1$  дерева, которые задают распределения  $\{T_i\}_{i=0}^{np(n)}$ . В последнем распределении  $T_{np(n)}$  значение для каждого  $f_n^s(t_i)$  выбирается из  $U(S_n)$ , т.е. это соответствует равномерному распределению.

Если существует противник, который отличает первое распределение от последнего, то существуют и два последовательных распределения, которые этот противник различает с хорошей вероятностью:

$$\Pr_{z \leftarrow T_0} [B(z) = 1] - \Pr_{z \leftarrow T_{np(n)}} [B(z) = 1] \geq \epsilon_n \implies \exists i : \Pr_{z \leftarrow T_i} [B(z) = 1] - \Pr_{z \leftarrow T_{i+1}} [B(z) = 1] \geq \frac{\epsilon_n}{np(n)}.$$

Дерево распределения  $T_{i+1}$  отличается от  $T_i$  в трёх вершинах:



Таким образом противник может отличить образ генератора  $G(s_v) = G^{(0)}(s_v)G^{(1)}(s_v)$  от полностью случайной строки  $s_u s_w$  с вероятностью больше  $\epsilon_n / (n \cdot p(n))$ .  $\square$

*Замечание 4.1.* Это было доказательство надёжности в смысле п.3. То же самое доказательство работает и для п.3'.

**Определение 4.3.** Пусть  $d_n$  — доступная случайная величина  $d_n \in \{0, 1\}^{k(n)}$ . *Много-разовый протокол с секретным ключом* — это пара полиномиальных вероятностных алгоритмов  $E(d, x)$  и  $D(d, m)$  таких, что  $D(d, E(d, x)) = x$ , и выполняются следующее условие надёжности.

- а) (*слабая надёжность*) Для любых полиномов  $p$  и  $q$  и для любого набора сообщений  $x_1, \dots, x_{p(n)} \in \{0, 1\}^{q(n)}$  при  $d \leftarrow d_n$  случайные величины  $E(d, x_1) \cdots E(d, x_{p(n)})$  и  $E(d, 0^{q(n)}) \cdots E(d, 0^{q(n)})$  вычислительно неотличимы.
- б) (*сильная надёжность*) Для любых полиномов  $p$  и  $q$ , любого  $x_1 \in \{0, 1\}^{q(n)}$  и семейства полиномиальных схем  $\{C_i\}_{i=1}^{p(n)-1}$  определим  $\{x_i\}_{i=2}^{p(n)}$  следующим образом:

$$\begin{aligned}
 d &\leftarrow d_n, \\
 c_1 &= E(d, x_1), \\
 x_2 &= C_1(x_1, c_1), \\
 c_2 &= E(d, x_1), \\
 x_3 &= C_2(x_1, c_1, c_2), \\
 c_3 &= E(d, x_2), \\
 x_4 &= C_3(x_1, c_1, c_2, c_3), \\
 &\vdots \\
 x_{p(n)} &= C_{p(n)-1}(x_1, c_1, \dots, c_{p(n)-1}), \\
 c_{p(n)} &= E(d, x_{p(n)}).
 \end{aligned}$$

Тогда получившаяся случайная величина  $c_1, \dots, c_{p(n)}$  вычислительно неотличима от случайной величины  $E(d, 0^{q(n)}) \cdots E(d, 0^{q(n)})$ .

**Теорема 4.2.** Если существует семейство псевдослучайных функций, то существует и много-разовый протокол с секретным ключом.

*Доказательство.* Докажем для сообщения из одного бита. Из этого будет следовать, что есть протокол для сообщения любой длины. Пусть  $f_s : \{0, 1\}^n \rightarrow \{0, 1\}$  — семейство псевдослучайных функций,  $s \in \{0, 1\}^n$ . Тогда  $d_n = U_n$ . Алгоритм шифрования  $E(d, m)$  для ключа  $d \leftarrow d_n$  и бита  $m \in \{0, 1\}$  выбирает  $z \leftarrow U_n$  и выдаёт  $(m \oplus f_d(z), z)$ . Алгоритм дешифровки  $D(d, y)$ , где  $y \in \{0, 1\}^{n+1}$ ,  $y = bz$ , выдаёт  $b \oplus f_d(z)$ . Корректность протокола очевидна. Нужно доказать много-разовую надёжность.

Пусть коды сообщений  $m_1, \dots, m_{p(n)}$  соответственно

$$(f_d(z_1) \oplus m_1, z_1), (f_d(z_2) \oplus m_2, z_2), \dots, (f_d(z_{p(n)}) \oplus m_{p(n)}, z_{p(n)}).$$

Нужно показать, что такая случайная величина вычислительно неотличима от строки из  $U_{(n+1) \cdot p(n)}$ . Если это не так, то мы могли бы отличить псевдослучайные функции от равномерного распределения (см. соответствующее определение надёжности псевдослучайных функций).  $\square$

*Замечание 4.2.* У этой схемы есть существенный недостаток: длина шифра в  $n + 1$  раз длиннее самого сообщения.

#### 4.1. Эффективная схема шифрования с закрытым ключом

Для шифрования сообщений  $m \in \{0, 1\}^{k(n)}$  возьмём семейство псевдослучайных функций  $f^s : \{0, 1\}^n \rightarrow \{0, 1\}^n$ , где  $s \in S_n = \{0, 1\}^n$ , и генератор псевдослучайных чисел  $G : \{0, 1\}^n \rightarrow \{0, 1\}^{k(n)}$ . Тогда алгоритм шифрования может быть устроен так: выберем случайное  $z \leftarrow U_n$  и определим  $E(d, m) = (G(f^d(z)) \oplus m, z)$ ,  $D(d, (c, z)) = G(f^d(z)) \oplus c$ .

**Утверждение 4.1.** *Описанная схема шифрования надёжна.*

*Схема доказательства.* Доказываем гибридным методом в два этапа. Сначала заменяем функции  $f^s$  на случайные строки длины  $n$  (за один шаг), а потом заменяем применения генераторов на случайные строки длины  $k(n)$  (за  $p(n)$  шагов).  $\square$

### 5. Протоколы шифрования с открытым ключом

**Определение 5.1.** Пусть  $(e_n, d_n)$  — доступные случайные величины. Пара вероятностных полиномиальных алгоритмов  $(E, D)$  называется *системой шифрования с открытым ключом*, если алгоритмы шифрования и дешифрования,  $E(e, m)$  и  $D(e, d, c)$ , связаны следующим соотношением (корректность):  $D(e, d, E(e, m)) = m$ . *Надёжность* протокола определяется следующим образом: для любого полинома  $p$  и любых сообщений  $x, y \in \{0, 1\}^{p(n)}$  случайные величины  $\langle e, E(e, x) \rangle$  и  $\langle e, E(e, y) \rangle$  вычислительно неотличимы для  $e \leftarrow e_n$ .

*Замечание 5.1.* В такой системе шифрования значение  $e$  сообщается всем (*публичный ключ*), а  $d$  — никому (*секретный ключ*).

*Замечание 5.2.* Шифрование с закрытым ключом называют *симметричным* (для шифрования и дешифрования используется один и тот же ключ), а шифрование с открытым ключом — *асимметричным* (ключи разные).

*Замечание 5.3.* Для протоколов с открытым ключом из одноразовой надёжности следует многократная надёжность.

**Определение 5.2** (trapdoor permutation family). Пусть задан полиномиальный вероятностный алгоритм  $K$ , который получив на вход  $1^n$  генерирует пару слов  $\langle e, d \rangle$  или выдаёт  $\perp$  (символ неудачи), причём последнее происходит с пренебрежимо малой вероятностью. Слова  $e$  и  $d$  называются соответственно *открытым и закрытым ключами*. 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  $l(n)$ . Последовательность пар  $\{(\langle e_n, d_n \rangle, f_n^e)\}$  будем называть *семейством односторонних перестановок с секретом*, если выполнены следующие условия.

1. (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 распределённой на  $D_n^e$  случайной величины при известном  $e_n$ . (Т.е. пара, состоящая из  $e$  и выхода алгоритма  $B$  статистически неотличима от пары, состоящей из  $e$  и случайной строки  $D_n^e$ .)

Из этого условия следует доступность случайной величины  $\langle e, U(D_n^e) \rangle$ .

2. (Полиномиальная вычислимость.) Функция  $\langle 1^n, e, x \rangle \rightarrow f_n^e(x)$  вычислима за полиномиальное от  $n$  время.

3. (Необратимость.) Для любой последовательности схем  $C_n$  полиномиального размера вероятность того, что  $C_n$  по  $\langle e, f_n^e(x) \rangle$  найдёт  $x$ , стремится к нулю быстрее любого обратного полинома от  $n$  ( $e$  выбирается случайно из  $A_n$ ,  $x$  выбирается случайно из  $D_n^e$ ).
4. (Возможность обращения при известном закрытом ключе.) Существует полиномиальный вероятностный алгоритм, который по тройке  $\langle 1^n, d, f_n^e(x) \rangle$  с вероятностью близкой к 1 вычисляет  $x$ . Здесь пара  $\langle e, d \rangle$  выбирается генератором случайной величины  $\langle e_n, d_n \rangle$ , а  $x$  выбирается равномерно в  $D_n^e$ .

*Замечание 5.4.* Предположение о существовании односторонних перестановок с секретом сильнее, чем существование односторонних функций. Если функция Рабина или RSA является односторонней, то существуют односторонние перестановки с секретом.

- В случае RSA открытым ключом будет  $\langle p \cdot q, z \rangle$ , а закрытым  $z^{-1} \bmod \phi(p \cdot q)$ .
- В функции Рабина открытым ключом будет  $p \cdot q$ , а секретным ключом будет  $\langle p, q \rangle$ .

**Определение 5.3.** Для односторонней перестановки с секретом  $f_n^e : D_n \rightarrow D_n$ , где  $D_n \subset \{0, 1\}^{k(n)}$ , будем называть *трудным битом* такую полиномиально вычислимую функцию  $h_n : D_n \rightarrow \{0, 1\}$ , для которой случайная величина  $h_n(U(D_n))$  трудна для  $\langle e, f_n^e(U(D_n)) \rangle$ .

**Теорема 5.1.** Если существует семейство перестановок с секретом, то существует протокол с открытым ключом.

*Доказательство.* Заметим, что если есть многократная надёжность, то достаточно научиться шифровать сообщения из одного бита (будем пересылать сообщение зашифрованное побитово). Воспользуемся теоремой Голдрейха-Левина и по перестановке с секретом  $f_n^e$  построим перестановку с секретом  $g_n^e$  с трудным битом  $h_n^e$ :

$$g_n^e(x, r) = f_n^e(x)r, \quad h_n^e(x, r) = x \odot r.$$

**Утверждение 5.1.** Величина  $h_n^e(x, r)$  является трудной для  $\langle e, g_n^e(x, r) \rangle$ .

*Доказательство.* Аналогично доказательству теоремы Голдрейха-Левина. □

Теперь построим протокол для кодирования одного бита. Для ключей  $\langle e, d \rangle \leftarrow K$ :

- алгоритм  $E(e, m)$ , где  $m \in \{0, 1\}$ , выбирает случайный элемент  $x \leftarrow U(D_n)$ , случайные биты  $r \leftarrow U_{l(n)}$  и выдаёт  $\langle m \oplus h_n^e(x, r), g_n^e(x, r) \rangle$ ;
- алгоритм  $D(e, d, \sigma y r)$  вычисляет  $x = (f_n^e)^{-1}(y)$  и возвращает  $\sigma \oplus h_n^e(x, r)$ .

Давайте докажем, что при известном  $e$  для любых последовательностей битов  $y_n, z_n$  случайные величины  $\langle y_n \oplus h_n^e(x, r), g_n^e(x, r), e \rangle$  и  $\langle z_n \oplus h_n^e(x, r), g_n^e(x, r), e \rangle$  будут вычислительно неотличимы. Обе эти случайные величины неотличимы от  $\langle U_1, U(D_n), e \rangle$  по соответствующей теореме о трудном бите. □

*Замечание 5.5.* Можно научиться посылать несколько длинные сообщения, если воспользоваться конструкцией, которую мы использовали для псевдослучайного генератора (т.е. получится последовательность случайных битов).

## 6. Протокол привязки (схема обязательства, commitment protocol)

Один участник хочет послать другому участнику некоторый зашифрованный секрет таким образом, что

- без знания ключа секрет невозможно расшифровать,
- не существует другого ключа, который расшифровывал код, в другой секрет.

### 6.1. Неинтерактивный протокол привязки к строке

**Определение 6.1.** *Неинтерактивный протокол привязки к строке* — это пара полиномиальных алгоритмов  $S$  и  $R$ :

- вероятностный алгоритм  $S(x, 1^n) = (c, k)$  шифрует секрет, где  $x$  — это секрет (строка),  $c(x)$  — шифр секрета, а  $k(x)$  — ключ,
- детерминированный алгоритм  $R(c, k) \in \{x, \perp\}$  по зашифрованному секрету и ключу либо восстанавливает *исходный* секрет, либо сообщает, что ключ не подходит.

При этом выполняются следующие условия.

1. (Полнота) Если  $S(x, 1^n) = (c, k)$ , то  $R(c, k) = x$  с вероятностью близкой к 1.
2. (Неразглашение) Для любого полинома  $p$  и любых последовательностей строк  $x_n, y_n$ ,  $|x_n| = |y_n| = p(n)$ , если  $c_n$  — привязка  $x_n$ ,  $d_n$  — привязка  $y_n$ , то случайные величины  $c_n$  и  $d_n$  вычислительно неотличимы.
3. (Недвусмысленность) Ни для какого  $n$  не существует таких  $c, k_1, k_2$ , что одновременно  $R(c, k_1) \neq \perp$ ,  $R(c, k_2) \neq \perp$  и  $R(c, k_1) \neq R(c, k_2)$ .

*Замечание 6.1.* Достаточно научиться привязывать к биту (т.е. достаточно построить конструкцию для  $p(n) = 1$ ).

**Определение 6.2.** Будем называть одностороннюю перестановку  $f_n : D_n \rightarrow D_n$  *правильной*, если равномерное распределение на  $D_n \cup \{\perp\}$  полиномиально моделируемо (т.е., если не получилось сгенерировать элемент из  $D_n$ , то выдаётся особый символ  $\perp \notin D_n$ ).

**Теорема 6.1.** *Если существует правильная односторонняя перестановка, то существует и неинтерактивный протокол привязки к биту.*

*Доказательство.* Пусть  $f_n : D_n \rightarrow D_n$  — односторонняя перестановка с трудным битом  $h_n$ . Будем привязывать бит  $\sigma \in \{0, 1\}$ . Выберем строку  $x$  из распределения на  $D_n$ , которое вычислительно неотлично от равномерного (есть по свойству доступности). Тогда

$$S(\sigma, 1^n) = (\underbrace{\langle h_n(x) \oplus \sigma, f_n(x) \rangle}_c, \underbrace{r_n}_k),$$

где  $r_n$  — это случайные биты, которые использовались для генерации  $x$ .<sup>2</sup> Соответственно,  $R(c, k)$  генерирует  $x$  используя в качестве случайных битов  $k$  и проверяет, что  $f_n(x)$  совпадает с соответствующими битами  $c$ . Если это так, то бит  $\sigma$  восстанавливается по первому биту  $c$ . Несложно увидеть, что свойство 1 выполняется по построению, свойство 2 следует из свойства трудного бита (утверждение 3.1), а свойство 3 из того, что  $K$  возвращает только элементы из  $D_n$ .  $\square$

<sup>2</sup>Нельзя в качестве ключа выбрать  $x$ , т.к. тогда нужно было бы уметь проверять, что  $x \in D_n$ .

## 7. Интерактивные протоколы

**Определение 7.1.** *Интерактивный протокол* — это пара вероятностных алгоритмов  $A(1^n, x, h)$  и  $B(1^n, y, h)$ , которые „общаются между собой”, передавая сообщения друг другу. Интерактивный протокол состоящий из  $t$  раундов устроен так:

$$\begin{aligned} s_1 &\leftarrow A(1^n, x, \lambda), \\ s_2 &\leftarrow B(1^n, y, s_1), \\ s_3 &\leftarrow A(1^n, x, \langle s_1, s_2 \rangle), \\ s_4 &\leftarrow B(1^n, y, \langle s_1, s_2, s_3 \rangle), \\ &\vdots \\ s_{t-1} &\leftarrow A(1^n, x, \langle s_1, s_2, \dots, s_{t-2} \rangle), \\ s_t &\leftarrow B(1^n, y, \langle s_1, s_2, \dots, s_{t-1} \rangle). \end{aligned}$$

В последнем сообщении  $s_t$  содержится некоторый признак (например, специальный символ), который сигнализирует об окончании общения. Алгоритмы такого вида будем называть *интерактивными алгоритмами*. Через  $\pi_{A(x), B(y)}$  будем обозначать историю взаимодействия алгоритмов  $A$  и  $B$  на входах  $x$  и  $y$  соответственно. Для  $t$ -раундового протокола  $\pi_{A(x), B(y)} = \langle s_1, s_2, \dots, s_t \rangle$ . После завершения общения алгоритмы  $A$  и  $B$  ещё раз запускаются на полученной истории и печатают результат, т.е. на стороне алгоритма  $A$  результатом будет  $A(1^n, x, \pi_{A(x), B(y)})$ , а на стороне алгоритма  $B$  —  $B(1^n, y, \pi_{A(x), B(y)})$ . Аналогичное определение можно дать и для схем, нужно только позаботиться о том, чтобы аккуратно закодировать неполную историю (можно, например, использовать алфавит с дополнительными символами).

### 7.1. Интерактивные протоколы привязки к строке

Для неинтерактивного протокола привязки нам потребовалось предположить существование правильных односторонних перестановок. Оказывается, что если добавить интерактивность, то можно построить протокол привязки при более слабом предположении.

**Определение 7.2.** *Интерактивный протокол привязки к строке* — полиномиальный (по суммарному времени, а следовательно и по количеству раундов) интерактивный протокол, заданный полиномиальными вероятностными алгоритмами  $S(1^n, \sigma, h)$  и  $T(1^n, h)$ , и детерминированный полиномиальный алгоритм  $R(1^n, c, k)$ , где  $S$  — алгоритм стороны, которая осуществляет привязку,  $T$  — алгоритм стороны, которая получает привязку, а  $R$  — алгоритм проверяющий, что привязка действительно соответствует исходной строке. Привязкой строки  $\sigma$  в этом интерактивном алгоритме будет  $c(\sigma) = \pi_{S(\sigma), T}$ . После окончания взаимодействия алгоритм  $S$  выдаёт ключ  $k(\sigma)$ , соответствующий этой привязке. Алгоритм  $R(c, k)$  выдаёт либо строку  $\sigma$ , по которой была построена привязка, либо один из специальных символов  $\perp_S$  и  $\perp_T$ , которые „обвиняют” соответственно, протокол  $S$  и  $T$  в жульничестве. Кроме того, должны выполняться следующие свойства.

1. (Корректность)  $R(\pi_{S(\sigma), T}, k_{n, \sigma}) = \sigma$  с вероятностью 1.
2. (Полнота) Для любого (схемного) противника  $T^*$

$$R(\pi_{S(\sigma), T^*}, k_{n, \sigma}) \in \{\sigma, \perp_T\}$$

с вероятностью пренебрежимо отличающейся от 1.



3. (Неразглашение) Для любого противника  $T^*$ , полинома  $p(n)$  и любых последовательностей строк  $x_n, y_n, |x_n| = |y_n| = p(n)$ , случайные величины  $\pi_{S(x_n), T^*}$  и  $\pi_{S(y_n), T^*}$  вычислительно неотличимы.
4. (Недвусмысленность привязки) Для любого противника  $S^*$  вероятность

$$\Pr[\exists k_1, k_2 \mid R(\pi_{S^*, T}, k_1) \neq \perp_S, R(\pi_{S^*, T}, k_2) \neq \perp_S, R(\pi_{S^*, T}, k_1) \neq R(\pi_{S^*, T}, k_2)]$$

пренебрежимо мала.

5. Вероятность  $\Pr[\exists k \mid R(\pi_{S^*, T}, k) = \perp_T]$  пренебрежимо мала.

**Теорема 7.1.** *Если существует генератор псевдослучайных чисел, то существует интерактивный протокол привязки к биту.*

*Доказательство.* Определим протокол привязки для бита  $\sigma \in \{0, 1\}$ . Возьмём генератор псевдослучайных чисел  $G_n : \{0, 1\}^n \rightarrow \{0, 1\}^{3n}$ . Алгоритм  $T(1^n)$  генерирует случайную строку  $r \leftarrow U_{3n}$  и посылает её  $S$ . Алгоритм  $S(1^n, \sigma)$  ожидает получения строки  $r \in \{0, 1\}^{3n}$ , генерирует строку  $s \leftarrow U_n$  и посылает сообщение

$$t = \begin{cases} G(s) \oplus r, & \sigma = 1, \\ G(s), & \sigma = 0. \end{cases}$$

Ключом для этой привязки будет  $k = \sigma s$ .

Алгоритм  $R(1^n, (r, t), k)$  проверяет, что все длины строк в привязке соответствуют ожидаемым:  $|r| = |t| = 3n, |k| = n + 1$ . Если длины неправильные, то  $R$  обвиняет соответствующую сторону. После этого алгоритм  $R$  по  $k = \sigma s$  вычисляет  $t \oplus G(s)$ :

- если получилось  $0^{3n}$ , то проверяет, что  $\sigma = 0$ , и выдаётся 0;
- если получилось  $r$ , то проверяет, что  $\sigma = 1$ , и выдаётся 1;
- если получилось что-то ещё, то выдаёт  $\perp_S$ .

Свойства 1, 2 и 5 выполняются по построению. Свойство 3 означает, что следующие случайные величины  $\langle r, G_n(s) \rangle$  и  $\langle r, G_n(s) \oplus r \rangle$  должны быть вычислительно неотличимы. Если мы умеем их различать, то умеем одну из них отличать от  $\langle r, U_{3n} \rangle$ , а это позволяет взломать генератор  $G_n$ . Осталось проверить свойство 4. Давайте оценим

$$\Pr[\exists s_0, s_1 \mid G_n(s_0) \oplus t = 0^{3n}, G_n(s_1) \oplus t = r] = \Pr[\exists s_0, s_1 \mid G_n(s_0) \oplus G_n(s_1) = r] \leq \frac{2^{2n}}{2^{3n}} = 2^{-n}.$$

□

## 7.2. Протокол подбрасывания монетки

**Определение 7.3.** *Интерактивный протокол подбрасывания монетки* — это полиномиальный (по суммарному времени и по количеству раундов) интерактивный протокол, заданный полиномиальными вероятностными алгоритмами  $A(1^n, h)$  и  $B(1^n, h)$ , и детерминированный полиномиальный алгоритм  $D(1^n, h)$ , где  $A$  и  $B$  — это алгоритмы сторон, а  $D$  — алгоритм, позволяющий по истории общения получить результат подбрасывания монетки. Стороны запускают  $D$  после завершения протокола и определяют результат. Для стороны  $A$  и любого противника  $B^*$  должны выполняться следующие свойства (интересы стороны  $A$ ):

1. вероятность  $\Pr[D(\pi_{A,B^*}) = \perp_A]$  пренебрежимо мала,
2. для любого  $\sigma \in \{0, 1\}$   $\Pr[D(\pi_{A,B^*}) = \sigma] \leq \frac{1}{2} + \alpha_n$ , где  $\alpha_n$  — пренебрежимо мала.

Такие же свойства должны выполняться и для стороны  $B$  и любого противника  $A^*$ .

**Теорема 7.2.** *Если существует интерактивный протокол привязки к биту, то существует и интерактивный протокол подбрасывания монетки.*

*Доказательство.* Пусть  $(S, T, R)$  — интерактивный протокол привязки. Сторона  $A$  генерирует бит  $\sigma$  и запускает алгоритм привязки  $S(\sigma)$ , сторона  $B$  запускает алгоритм  $T$ ; в конце протокола обе стороны знают привязку  $c(\sigma)$ , а сторона  $A$  ещё дополнительно знает ключ  $k(\sigma)$ . Потом сторона  $B$  генерирует бит  $\tau$  и посылает его алгоритму  $A$ , а сторона  $A$  в ответ посылает ключ  $k(\sigma)$ .

Алгоритм  $D(c(\sigma), \tau, k(\sigma))$  запускает алгоритм  $R(c(\sigma), k(\sigma))$  для проверки для привязки. Если алгоритм проверки привязки обвиняет какую-то сторону, то  $D$  обвиняет соответствующую сторону. В противном случае  $D$  выдаёт  $\sigma \oplus \tau$ .

Давайте проверим, что интересы стороны  $A$  соблюдаются. Свойство 1 следует из протокола привязки (т.к. обвинение стороны возможно только в протоколе привязки). Предположим, что второе свойство нарушается. Тогда  $B^*$  может подбирать бит  $\tau$  так, чтобы угадывать  $\sigma$  с нетривиальной вероятностью, а значит  $B^*$  можно использовать для взлома протокола привязки.

Интересы стороны  $B$  соблюдаются из-за недвусмысленности протокола привязки, поэтому вероятность того, что  $A$  сможет подменить ключ пренебрежимо мала.  $\square$

## 8. Протоколы с нулевым разглашением

### 8.1. Понятие разглашения

Пусть алгоритм  $A(1^n, x, e)$ , где  $x \in D_n$ ,  $e \leftarrow \nu_n$ , общается с некоторой полиномиальной схемой  $C_n$ , которая уже имеет некоторую информацию о  $x$  и  $e$ , а именно  $C_n$  знает значение функции  $g(1^n, x, e)$ . Пусть имеется ещё одна функция  $f(1^n, x, e)$ . Давайте определим, что значит, алгоритм  $A$  *разглашает* информацию  $f(1^n, x, e)$ , если противнику уже известна информация  $g(1^n, x, e)$  о входах  $x, e$  (для краткости будем опускать параметр  $1^n$ ).

**Определение 8.1.** Будем говорить, что алгоритм  $A$  *разглашает только  $f$  при известном  $g$  на области  $D_n$* , если существует полиномиальный вероятностный алгоритм  $M$ , который для любой последовательности слов  $\{x_n \in D_n\}$  и любой последовательности полиномиальных схем  $C_n$  получая на вход  $f(x_n, e)$  и схему  $C_n$  в качестве чёрного ящика (т.е. как оракул) генерирует случайную величину  $\alpha_n(x_n, e)$ , вычислительно неотличимую от протокола общения  $\beta_n(x_n, e)$  между алгоритмом  $A(x_n, e)$  и схемой  $C_n(g(x_n, e), \cdot)$  при известном  $g(x_n, e)$  (т.е. случайные величины  $g(x_n, e)\alpha_n(x_n, e)$  и  $g(x_n, e)\beta_n(x_n, e)$  вычислительно неотличимы). Вероятность берётся по выбору  $e \leftarrow \nu_n$  и случайным битами алгоритмов  $A(x_n, e)$  и  $M$ . Множество  $D_n$  называется *областью неразглашения алгоритма*, а алгоритм  $M$  — *симулятором*.

*Пример 8.1.* Пусть  $e \leftarrow U_n$ , а  $A$  посылает  $x \oplus e$ . Этот алгоритм имеет нулевое разглашение. Симулятору достаточно выдать случайную строку из  $U_n$ .

*Пример 8.2.* Пусть дана схема шифрования  $K, E, D$  с открытым ключом. Для открытого ключа  $e$  алгоритм шифрования  $E(e, x)$  разглашает пару  $(|x|, e)$  при известном  $e$ . Действительно,  $E(e, x)$  вычислительно неотличима от  $E(e, 0^{|x|})$ . Симулятор  $M$  выдаёт  $E(e, 0^{|x|})$ .

**Пример 8.3.** Пусть дан интерактивный протокол привязки  $S, T, R$  для бита. Алгоритм  $S$  имеет нулевое разглашение. Симулятор по схеме  $C$  моделирует общение алгоритма  $S(0)$  и схемы  $C$  и выдаёт протокол общения.

**Лемма 8.1.** *Выполняются следующие свойства разглашения.*

1. Для любых функций  $f, g, h$ , если  $A$  разглашает только  $f$  при известном  $g$ , то  $A$  разглашает только пару  $f, h$  при известном  $g$ .
2. Для любых функций  $f, g, h$ , если  $A$  разглашает только  $f$  при известной паре  $g, h$ , то  $A$  разглашает только  $f$  при известном  $g$ .

**Определение 8.2.** Если  $A$  разглашает только  $g$  при известном  $g$ , то мы будем говорить, что  $A$  имеет нулевое разглашение при известном  $g$ . Если  $g$  является константной функцией, то мы говорим, что  $A$  разглашает только  $f$ . Если при этом ещё и  $f$  является константной функцией, то мы говорим, что  $A$  имеет нулевое разглашение.

**Определение 8.3.** Пусть  $A$  и  $B$  вероятностные полиномиальные интерактивные алгоритмы. Выполним сначала  $A$  на паре входов  $x$  и  $e$ , а потом  $B$  на паре входов  $(x, c)$  и  $e$ , где  $c$  — протокол общения, полученный в ходе выполнения  $A$ . Полученный алгоритм будем обозначать как  $(A, B)$ .

**Теорема 8.1** (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  $D_n$ .

*Доказательство.* Пусть  $M$  и  $N$  — симуляторы для  $A$  и  $B$  соответственно. Построим симулятор для  $(A, B)$ . Алгоритмы  $A$  и  $B$  получают на вход некоторое неизвестное  $x$  и случайное  $e$  и после этого по очереди беседуют с некоторой схемой  $C$ , которой известно  $g(x, e)$ . В результате получается протокол беседы  $\pi_{(A,B)(x,e),C(g(x,e))}$ , который является случайной величиной с параметром  $x$ .

Симулятор для  $(A, B)$  должен по схеме  $C$  и подсказке о  $x$  и  $e$  в виде пары  $f(x, e), g(x, e)$  сгенерировать случайную величину, вычислительно неотличимую от случайной величины  $\pi_{(A,B)(x,e),C(g(x,e))}$  при известном  $g(x, e)$ .

В дальнейшем совместно распределённые случайные величины  $f(x, e), g(x, e)$  и  $h(x, e)$  будем для краткости обозначать  $f, g$  и  $h$  соответственно. Через  $C^w$  мы будем обозначать схему  $C$ , в которую подставили (зашили) слово  $w$  (т.е. первые  $|w| + 1$  символ входа равны слову  $w$ , а остальные входы свободны).

Общение алгоритма  $(A, B)(x, e)$  со схемой  $C$  устроено так: схема  $C$  читает  $g$  и превращается в  $C^g$ , затем беседует с  $A(x, e)$ , в результате чего получается некоторый протокол общения  $u$ , потом алгоритм  $B(x, e)$  общается уже со схемой  $C^{gu}$ . Давайте обозначим протокол общения  $B$  и  $C^{gu}$  через  $\mathcal{B}(C^{gu})$ . Тогда весь протокол общения  $(A, B)(x, e)$  со схемой  $C$  — это конкатенация случайных величин  $u$  и  $\mathcal{B}(C^{gu})$ .

Симулятор для  $(A, B)$  должен по  $f, h$  и  $C$  сгенерировать некоторую случайную величину, вычислительно неотличимую от  $u\mathcal{B}(C^{gu})$  при известном  $g$ . Для этого он сначала запускает симулятор  $M$  на  $f$  и  $C$  (значение  $g$  известно), в результате чего получается протокол общения  $\tilde{u}$ . После этого он запускает  $N$  на входе  $h$  и  $C^{\tilde{u}}$ , в результате чего получается последовательность сообщений  $\mathcal{N}(C^{g\tilde{u}})$ . В результате симулятор выдаёт  $\tilde{u}\mathcal{N}(C^{g\tilde{u}})$ .

Отметим, что по свойству симулятора  $N$  известно, что  $\mathcal{N}(C^{g\tilde{u}})$  вычислительно неотличима от протокола общения  $B(x, e)$  и  $C^{g\tilde{u}}$ .

Зафиксируем произвольную последовательность строк  $x_n$  полиномиальной длины. Нам нужно показать, что случайные величины

$$\alpha = \tilde{u}\mathcal{N}(C^{g\tilde{u}}) \quad \text{и} \quad \gamma = u\mathcal{B}(C^{gu})$$

вычислительно неотличимы при известном  $g$ . Докажем, что эти две случайные величины неотличимы от какой-нибудь третьей случайной величины и потом воспользуемся транзитивностью. Рассмотрим два случая.

1. Пусть  $h$  является полиномиально вычислимой. Рассмотрим случайную величину

$$\beta = u\mathcal{N}(C^{gu}).$$

Покажем сначала, что  $\beta$  вычислительно неотличима от  $\alpha$ . Так как мы зафиксировали  $x_n$ ,  $e$  является частью  $g$ , а функция  $h$  полиномиально вычислима, то следующее преобразование можно вычислить схемой полиномиального размера:

$$gv \rightarrow gv\mathcal{N}(C^{gv}).$$

Применяя это преобразование к  $gu$  мы получим  $g\beta$ , а применяя его к  $g\tilde{u}$  получим  $g\alpha$ . Исходные  $gu$  и  $g\tilde{u}$  вычислительно неотличимы, значит неотличимы и  $g\alpha$  и  $g\beta$ .

Покажем теперь неотличимость  $\gamma$  и  $\beta$ . Если мы зафиксируем случайные биты  $A$ , то  $u$  будет функцией от  $g$  ( $e$  входит в состав  $g$ ), которую можно вычислить схемой полиномиального размера. Поэтому нам достаточно показать вычислительную неотличимость

$$g\mathcal{B}(C^{gu}) \quad \text{и} \quad g\mathcal{N}(C^{gu}).$$

„Зашьём“ схему для  $u$  в схему  $C^{gu}$ , в результате получится семейство схем  $H^g$ :

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

По свойству симулятора  $N$  случайные величины  $g\mathcal{B}(H^g)$  и  $g\mathcal{N}(H^g)$  вычислительно неотличимы, следовательно неотличимы и  $g\mathcal{B}(C^{gu})$  и  $g\mathcal{N}(C^{gu})$  (эти величины имеют такие же распределения, т.к. симулятор общается со схемой как с чёрным ящиком, и поэтому не видит разницы между  $H^g$  и  $C^{gu}$ ). Из этого, как мы уже заметили, следует вычислительная неотличимость и  $\gamma$  и  $\beta$ .

2. Пусть  $f$  является полиномиально вычислимой. Рассмотрим случайную величину

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

Доказательство устроено аналогично. Для неотличимости  $\alpha$  и  $\beta$  при известном  $g$  нам нужно, чтобы  $f$  была полиномиально вычислимой функцией от  $g, x$  — это позволяет вычислить  $\tilde{u}$  по  $g$  и случайным битам  $M$  схемой полиномиального размера. А для отличимости  $\alpha$  и  $\beta$  при известном  $g$  нам опять нужно, чтобы  $e$  было полиномиально вычислимой функцией от  $g, x$ , что позволяет вычислить  $\mathcal{B}(C^{gu})$  по  $g$  и  $u$  вероятностной схемой полиномиального размера.

□

*Замечание 8.1.* Теорема о неразглашении при повторении выполняется и в ситуации, когда алгоритму  $B$  дополнительно на вход подаётся протокол  $u$  общения  $A$  и  $C_n$ , а так же использованные при этом случайные биты  $A$ .

**Теорема 8.2** (о неразглашении при последовательном выполнении, улучшенная). 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  $B(u, r, x, e)$  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 и алгоритма  $A$  с любой схемой  $C_n$  полиномиального размера и его случайных битов  $r$  с почти единичной вероятностью  $(u, r, x) \in E_n$ . Тогда алгоритм  $(A, B)$  разглашает только пару  $(f(x, e), h(x, e))$  при известном  $g(x, e)$  на области  $D_n$ .

**Определение 8.4.** Для интерактивного алгоритма  $A$  через  $A^k$  будем обозначать последовательное выполнение  $A$  на входах  $(x, 1, e), (x, 2, e), \dots, (x, k, e)$ .

**Теорема 8.3** (о неразглашении при последовательном применении полиномиального числа алгоритмов). Пусть  $g(x, e) = (e, h(x, e))$ , где  $h$  — произвольная функция, а  $f(x, e)$  некоторая полиномиально вычисляемая функция. Если алгоритм  $A$  разглашает только  $f(x, e)$  при известном  $g(x, e)$ , то алгоритм  $A^{k(n)}$  для некоторого полинома  $k$  разглашает только  $f(x, e)$  при известном  $g(x, e)$ .

*Доказательство.* Обозначим через  $M$  симулятор для алгоритма  $A$  и построим симулятор  $M^k$  для алгоритма  $A^k$ . Такой симулятор должен по входам  $f$  и  $C$  породить случайную величину, вычислительно неотличимую от протокола общения  $A^k(x, e)$  и  $C^g$ .

Обозначим через  $\mathcal{A}_i(C, e)$  протокол общения алгоритма  $A(x, i, e)$  со схемой  $C^g$ , а через  $\mathcal{M}_i(C, e)$ , соответственно, результат работы симулятора  $M$  на входах  $f$  и  $C$ .

Зафиксируем произвольную полиномиальную схему  $C$ . Протокол общения  $\gamma$  алгоритмов  $A^k(x, e)$  и  $C^g$  можно получить последовательным применением к пустой строке для  $j = 1, \dots, k$  вероятностных операторов

$$R_j^A : u \rightarrow u\mathcal{A}_j(C^{g^u}, e).$$

Если применять „близкие“ операторы

$$R_j^M : u \rightarrow u\mathcal{M}_j(C^{g^u}, e),$$

то получится некоторая случайная величина  $\alpha$ . Покажем, что  $\alpha$  вычислительно неотличима от  $\gamma$  при известном  $g$ .

Зафиксируем произвольную последовательность  $x_n$  слов полиномиальной длины для алгоритма  $A^k$ . Рассмотрим гибридные случайные величины  $\beta_0, \beta_1, \dots, \beta_k$ , которые получаются смешиванием этих двух вероятностных операторов: для получения случайной величины  $\beta_i$  мы сначала последовательно применяем  $R_1^A, \dots, R_i^A$ , а потом  $R_{i+1}^M, \dots, R_k^M$ . Покажем, что две последовательные случайные величины  $\beta_{i-1}$  и  $\beta_i$  неотличимы при известном  $g$  (из этого будет следовать неотличимость  $\gamma = \beta_0$  и  $\alpha = \beta_k$ ).

Обозначим через  $\delta$  случайную величину, полученную применением к пустой строке операторов  $R_1^A, \dots, R_i^A$ , т.е. это протокол первых  $i - 1$  запуска  $A$ . Через  $F(u, e)$  обозначим случайную величину, полученную из строки  $u$  путём последовательного применения  $R_{i+1}^M, \dots, R_k^M$ . В этих обозначениях

$$\beta_i = F(\delta\mathcal{A}_i(C^{g^\delta}, e), e), \quad \beta_{i-1} = F(\delta\mathcal{M}_i(C^{g^\delta}, e), e).$$

Поскольку оператор  $F(u, e)$  можно вычислить вероятностной схемой полиномиального размера по  $x_n, e$  и  $u$  ( $f$  полиномиально вычислимо), нам достаточно доказать неотличимость  $\delta\mathcal{A}_i(C^{g^\delta}, e)$  и  $\delta\mathcal{M}_i(C^{g^\delta}, e)$  при известном  $g$ . Слово  $\delta$  зависит от  $e$  и случайных битов, использованных  $A$  в первых  $i - 1$  запусках. Давайте зафиксируем эти биты. Теперь  $\delta$  становится функцией от  $e$  ( $x_n$  мы тоже зафиксировали), а следовательно вычислима

схемой полиномиального размера. Поэтому достаточно доказать неотличимость  $\mathcal{A}_i(C^{g^\delta}, e)$  и  $\mathcal{M}_i(C^{g^\delta}, e)$  при известном  $g$ . Теперь „зашьём“ вычисление  $\delta$  в схему. Получится схема  $H^g \equiv C^{g^\delta}$ . Тогда

$$\mathcal{A}_i(C^{g^\delta}, e) = \mathcal{A}_i(H^g, e), \quad \mathcal{M}_i(C^{g^\delta}, e) = \mathcal{M}_i(H^g, e).$$

По определению симулятора  $\mathcal{A}_i(H^g, e)$  и  $\mathcal{M}_i(H^g, e)$  вычислительно неотличимы при известном  $g$ . Это наблюдение завершает доказательство.  $\square$

## 8.2. Языки с нулевым разглашением

**Определение 8.5.** Для языка  $L$  мы будем говорить, что он обладает *доказательством с нулевым разглашением*, если существуют такие полиномиальные вероятностные интерактивные алгоритмы  $P$  и  $V$ :

1. если  $x \in L$ , то существует  $y$ ,  $|y| \leq p(x)$ :  $\Pr[\text{out}_{P(x,y), V(x)=1}] \geq 1 - \alpha_n$ ,
2. если  $x \notin L$ , то для любой функции  $P^*$ :  $\Pr[\text{out}_{P^*, V(x)=1}] < \alpha_n$ ,
3.  $P(x, y)$  разглашает только  $(x, L(x))$ ,

где  $\alpha_n$  пренебрежимо мала.

**Определение 8.6.**  $ZKP$  — это класс языков, для которые обладают доказательством с нулевым разглашением.

*Замечание 8.2.*  $GI$  обладает протоколом со статистически неотличимым нулевым разглашением (т.е. в определении нулевого разглашения можно заменить вычислительную неотличимость на статистическую).

**Определение 8.7.** Язык изоморфизма графов  $GI$  — это язык пар графов  $(G_0, G_1)$  таких, что  $G_1$  изоморфен  $G_0$ .

**Лемма 8.2.**  $GI \in ZKP$ .

*Доказательство.* Алгоритм  $P$  на своей стороне получает  $G_0, G_1$  и перестановку вершин  $\tau \in S_n$ :  $\tau(G_0) = G_1$  ( $n$  обозначает число вершин). Алгоритм  $P$  выбирает случайную перестановку  $\pi \leftarrow U(S_n)$  и посылает  $H = \pi(G_0)$  алгоритму  $V$ . Алгоритм  $V$  в ответ посылает случайный бит  $\alpha \in \{0, 1\}$ . В ответ алгоритм  $P$  вычисляет  $\sigma = \pi \circ \tau^{-\alpha}$  (т.е. такую перестановку, что  $\sigma(G_\alpha) = \pi(G_0)$ ). Алгоритм  $V$  на своей стороне проверяет, что  $\sigma(G_0) = H$  и принимает, если это так. Этот протокол нужно повторить полиномиальное число раз для уменьшения ошибки.

Условие 1 выполняется по построению. Условие 2 выполняется, т.к. на каждой итерации алгоритм  $V$  ловит алгоритм  $P^*$  с вероятностью  $\frac{1}{2}$ . Остаётся проверить условие 3 и тут нам потребует теорема 8.3.

Пусть  $F$  — это схема, с которой общается (честный) алгоритм  $P$ . Будем рассматривать поведение только на входе из языка, т.е. на паре изоморфных графов. Давайте считать, что  $F$  — детерминирована и не нарушает протокол, т.е.  $F(H)$  — это один бит (который выбирается детерминировано). Тогда протокол задаётся следующей тройкой из множества

$$\{(H, \alpha, \sigma) : \alpha \in \{0, 1\}, \sigma \in S_n, F(H) = \alpha, \sigma(G_\alpha) = H\}.$$

Будем называть это множество *множеством хороших троек*.

Давайте покажем, что протокол (как случайная величина) равномерно распределен на множестве хороших троек. Рассмотрим такую хорошую тройку  $(H, \alpha, \sigma)$ . Так как  $\sigma = \pi \circ \tau^{-\alpha}$ , то по  $\sigma$ ,  $\alpha$  и  $\tau$  можно однозначно восстановить  $\pi$ , а по  $\pi$  однозначно получается  $H$ . Обратно

по  $\pi$  можно восстановить хорошую тройку ( $\alpha = F(\pi(G_0))$ ). Таким образом хороших троек столько же, сколько различных перестановок  $\pi$ , а именно  $n!$ .

Теперь нам нужно научиться генерировать равномерное распределение на хороших тройках. Будем генерировать следующим образом:  $\alpha \leftarrow U_1$ ,  $\sigma \leftarrow U(S_n)$ ,  $H = \sigma(G_\alpha)$ , если теперь  $F(H) = \alpha$ , то вернём  $(H, \alpha, \sigma)$ , иначе вернём  $\perp$ . Заметим, что

$$\Pr[F(\sigma(G_\alpha)) = \alpha] = \frac{1}{2} \Pr[F(\sigma(G_0)) = 0] + \frac{1}{2} \Pr[F(\sigma(G_1)) = 1].$$

Так как  $\sigma$  выбирается случайно, а  $G_0 \simeq G_1$ , то это то же самое, что

$$\Pr[F(\sigma(G_\alpha)) = \alpha] = \frac{1}{2} \Pr[F(\sigma(G_0)) = 0] + \frac{1}{2} \Pr[F(\sigma(G_0)) = 1] = \frac{1}{2}.$$

Теперь этот алгоритм можно повторить  $n$  раз и получить распределение, неотличимое от равномерного.

Если  $F$  выдаст не один бит, а некоторую строку  $\beta$ , то симулятор должен выдать „оборванный протокол“  $(H, \beta)$ .

Мы доказали эту теорему для детерминированного  $F$ . Если  $F$  использует случайные биты, для каждого фиксированного значения этих битов теорема верна, значит и для случайного выбора случайных битов тоже.  $\square$

Покажем, что если существуют односторонние функции, то  $NP \subseteq ZKP$ .

**Теорема 8.4.** *Если существуют односторонние функции, тогда для любого  $L \in NP$  существует вероятностный полиномиальный интерактивный алгоритмы  $P$  и  $V$ :*

1. если  $x \in L$ , то любой подсказки  $y$  для  $x$ ,  $|y| \leq p(x)$ :  $\Pr[\text{out}_{P(x,y),V(x)=1}] \geq 1 - \alpha_n$ ,
2. если  $x \notin L$ , то для любой функции  $P^*$ :  $\Pr[\text{out}_{P^*,V(x)=1}] < \alpha_n$ ,
3.  $P(x, y)$  разглашает только  $(x, R(x, y))$ , где  $R$  — предикат задающий  $L$ .

*Доказательство.* Достаточно доказать для какого-либо  $NP$ -полного языка относительно сведений, которые сохраняют подсказки. Пусть  $L \in NP$  с предикатом  $R$ ,  $L^*$  —  $NP$ -полный с предикатом  $R^*$ . Пусть существует полиномиально вычислимая функция  $f : x \in L \iff f(x) \in L^*$ , и существует полиномиально вычислимая функция  $g : R(x, y) = 1 \implies R(f(x), g(y))$ . Если мы докажем существование  $P$  и  $V$  для  $L^*$ , то из этого легко можно построить  $P$  и  $V$  для  $L$  с необходимыми свойствами.

Будем доказывать для языка 3-*COL* неориентированных три-раскрашиваемых графов. И при этом докажем с усиленным свойством 2.

- 2'. Существует такой оракульный вероятностный полиномиальный по времени алгоритм  $A$ , что если для какой-то вероятностной функции  $P^*$  и любого  $x$

$$\Pr[\text{out}_{P^*,V(x)} = 1] = p,$$

то  $A^{P^*}$  выдаёт подсказку для  $x$  с вероятностью не менее  $p - \alpha(|x|)$ , где  $\alpha(n)$  пренебрежимо мало.

(Это свойство гарантирует, что если  $p$  не пренебрежимо мало, то  $x \in L$ .)

**Алгоритм.**  $P$  на входе  $(G, \sigma)$  выбирает случайную перестановку  $\pi \leftarrow S_3$  и вычисляет  $\alpha = \pi \circ \sigma$ . После этого запускается интерактивный алгоритм привязки для цвета каждой вершины. Если на этом этапе  $V$  заблокировал, то протокол останавливается.  $V$  выбирает случайное ребро  $(u, v)$ .  $P$  в ответ посылает привязки  $k_u$  и  $k_v$  для вершин  $u$  и  $v$ .  $V$  проверяет привязки. Если привязки корректны и цвета различные, то попытка успешна. В противном случае  $V$  возвращает 0. Нужно повторить этот алгоритм  $2n|E|$  раз.

Можно считать, что  $p \geq e^{-n}$  (иначе  $p$  пренебрежимо мало). Определим вероятности  $p_1, p_2, \dots, p_{2n|E|}$ , где  $p_i = \Pr[\text{успех в раунде } i \mid \text{успех в предыдущих раундах}]$ . Тогда

$$p = \prod_{i=1}^{2n|E|} p_i.$$

Тогда по предположению существует  $i$ :  $p_i > e^{-\frac{1}{2|E|}} \geq 1 - \frac{1}{2|E|}$ . Построим алгоритм для поиска раскраски  $A$ . Этот алгоритм будет общаться с  $P^*$ : в раунде  $j$  алгоритм  $A$  будет спрашивать  $P^*$  про все рёбра (для этого мы будем подсовывать ей историю предыдущих  $j-1$  раундов без истории  $j$ -го раунда). Если нам удалось найти раскраску, то возвращаем её. Если нет, то повторим этот раунд  $n$  раз. Для выбранного  $i$  вероятность  $p_i$  большая — если мы спросим все рёбра, то вероятность успеха как минимум  $\frac{1}{2}$ . После  $n$  повторений получается вероятность  $1 - 2^{-n}$ . Заметим, что построенный алгоритм найдёт раскраску только при условии, что в предыдущих раундах был успех, т.е.

$$\Pr[\text{найдёт раскраску в раунде } i \mid \text{успех в предыдущих раундах}] \approx 1.$$

Отсюда следует, что

$$\Pr[\text{найдёт раскраску в раунде } i] \geq \Pr[\text{успех в предыдущих раундах}] \geq p.$$

Осталось доказать нулевое разглашение. Алгоритм ведёт себя одинаково в каждом раунде, так что достаточно доказать для одного раунда. Будем строить симулятор  $M$ . Возьмём  $\beta$  — случайную раскраску в три цвета и запустим протокол привязки для цвета каждой вершины, в результате чего получим историю  $S_\beta$ . Запустим  $V^*(\pi_{S_\beta, V^*})$  и получим ребро  $(u, v)$ . Если  $\beta(u) \neq \beta(v)$ , то выдадим ключи  $k_u$  и  $k_v$ . Иначе повторим. Всего  $n$  повторений. Если нам ни разу не повезло, то вернём пустую строку.

Давайте сначала покажем, что каждую попытку нам везёт с вероятностью  $2/3$ .

$$\begin{aligned} \Pr[V^*(\pi_{S_\beta, V^*}), \beta(u) \neq \beta(v)] &= \sum_{\substack{(u,v) \in E \\ a,b \in \{1,2,3\} \\ a \neq b}} \Pr[V^*(\pi_{S_\beta|v \leftarrow a, u \leftarrow b, V^*}) = (u, v), \beta(u) = a, \beta(v) = b] \\ &= \sum_{\substack{(u,v) \in E \\ a,b \in \{1,2,3\} \\ a \neq b}} \frac{1}{9} \Pr[V^*(\pi_{S_\beta|v \leftarrow a, u \leftarrow b, V^*}) = (u, v)]. \end{aligned}$$

По свойствам привязки существует случайная величина  $\Theta_{V^*}$ , вычислительно неотличимая от случайной величины  $\pi_{S_\beta|v \leftarrow a, u \leftarrow b, V^*}$ . Тогда продолжая предыдущее рассуждение

$$\approx \sum_{\substack{(u,v) \in E \\ a,b \in \{1,2,3\} \\ a \neq b}} \frac{1}{9} \Pr[V^*(\Theta_{V^*}) = (u, v)] = \frac{6}{9} \sum_{(u,v) \in E} \Pr[V^*(\Theta_{V^*}) = (u, v)] = \frac{2}{3}.$$

Осталось показать, что полученное распределение будет вычислительно неотлично от настоящего. Обозначим полученное распределение  $\eta$ .

$$\Pr[\eta = d] = \Pr[\pi_{P_\beta, V^*} = d \mid \beta(u) \neq \beta(v)]. \quad (1)$$



Пусть  $A$  — схема полиномиального размера (различитель). Мы хотим показать, что

$$\Pr[A(\pi_{P_\alpha, V^*}) = 1] \approx \Pr[A(\eta) = 1] \quad (2)$$

По определению условной вероятности из (1) получаем

$$\Pr[A(\eta) = 1] \approx \frac{3}{2} \Pr[A(\pi_{P_\beta, V^*}) = 1, \beta(u) \neq \beta(v)].$$

С другой стороны

$$\Pr[A(\pi_{P_\alpha, V^*}) = 1] = \sum_{(u,v)} \sum_{a \neq b} \Pr[A(\pi_{P_\alpha, V^*}) = 1, V^*(\pi_{S_\alpha, V^*}) = (u, v), \alpha(u) = a, \alpha(v) = b].$$

Применим тот же трюк с избавлением зависимости от  $a$  и  $b$ , получаем

$$\begin{aligned} &= \sum_{(u,v)} \sum_{a \neq b} \Pr[A(\pi_{P_{\alpha|u \leftarrow a, v \leftarrow b}, V^*}) = 1, V^*(\pi_{S_{\alpha|u \leftarrow a, v \leftarrow b}, V^*}) = (u, v), \alpha(u) = a, \alpha(v) = b] \\ &= \frac{1}{6} \sum_{(u,v)} \sum_{a \neq b} \Pr[A(\pi_{P_{\alpha|u \leftarrow a, v \leftarrow b}, V^*}) = 1, V^*(\pi_{S_{\alpha|u \leftarrow a, v \leftarrow b}, V^*}) = (u, v)]. \end{aligned} \quad (3)$$

Раскладывая таким же образом  $\Pr[A(\eta) = 1]$  получаем

$$\begin{aligned} &\approx \frac{3}{2} \sum_{(u,v)} \sum_{a \neq b} \Pr[A(\pi_{P_{\beta|u \leftarrow a, v \leftarrow b}, V^*}) = 1, V^*(\pi_{S_{\beta|u \leftarrow a, v \leftarrow b}, V^*}) = (u, v), \beta(u) = a, \beta(v) = b] \\ &= \frac{1}{6} \sum_{(u,v)} \sum_{a \neq b} \Pr[A(\pi_{P_{\beta|u \leftarrow a, v \leftarrow b}, V^*}) = 1, V^*(\pi_{S_{\beta|u \leftarrow a, v \leftarrow b}, V^*}) = (u, v)]. \end{aligned} \quad (4)$$

Мы хотим показать, что можно построить распределение, вычислительно неотличимое от

$$\pi_{S_{\beta|u \leftarrow a, v \leftarrow b}, V^*} = \pi_{P_{\beta|u \leftarrow a, v \leftarrow b}, V^*}, (k_u, k_v)$$

для какой-то фиксированной раскраски  $\beta$ . Проблема только в генерации  $k_u$  и  $k_v$ , т.к. мы не знаем, при помощи каких случайных битов они сгенерированы, но знаем  $a$  и  $b$ . Поэтому можно сгенерировать  $k_u$  и  $k_v$  при помощи других случайных бит и воспользоваться свойством неразглашения протокола привязки. Получится некоторое распределение  $\Theta_{u,v,a,b}$ , которое “не зависит” от  $\beta$  (тут мы пользуемся нулевым разглашением протокола привязки). Поэтому (3) и (4) примерно равны, а следовательно соотношение (2) выполняется.  $\square$

## 9. Протоколы цифровой подписи

Цифровая подпись позволяет подписывать цифровые документы, при этом все могут эту подпись проверить, но никто не может её подделать.

**Определение 9.1.** *Протокол цифровой подписи* — это тройка полиномиальных вероятностных алгоритмов  $(G, S, V)$ : алгоритм  $G(1^n)$  генерирует пару ключей  $(e_n, d_n)$ , где  $e_n$  известен всем, а  $d_n$  только владельцу, алгоритм  $S(e_n, d_n, m)$  создаёт подпись  $s$  для документа  $m$ , алгоритм  $V(e_n, m, c)$  проверяет корректность подписи. Должны выполняться следующие свойства.

1. (Полнота) Вероятность того, что  $V(e_n, m, S(e_n, d_n, m)) = 1$  отличается от 1 на пренебрежимо малую величину.

2. (Надёжность) Для любого полинома  $k(n)$  и любого полиномиального противника  $C = \{C_i\}_{i=1}^{k(n)+1}$  определим  $\{x_i\}$  и  $\{c_i\}$  следующим образом

$$\begin{aligned}(e, d) &\leftarrow G(1^n), \\ x_1 &= C_1(e), \\ c_1 &= S(e, d, x_1), \\ x_2 &= C_2(e, c_1), \\ c_2 &= S(e, d, x_2), \\ &\vdots \\ x_{k(n)} &= C_{k(n)}(x_1, c_1, c_2, \dots, c_{k(n)-1}), \\ c_{k(n)} &= S(e, d, x_{k(n)}), \\ (x_{k(n)+1}, c_{k(n)+1}) &= C_{k(n)+1}(x_1, c_1, c_2, \dots, c_{k(n)}).\end{aligned}$$

Тогда  $\Pr[x_{k(n)+1} \notin \{x_1, x_2, \dots, x_{k(n)}\} \wedge V(e_n, x_{k(n)+1}, c_{k(n)+1}) = 1]$  пренебрежимо мала.

**Теорема 9.1.** *Если существует односторонняя функция, то существует и протокол цифровой подписи.*

### 9.1. Одноразовый протокол подписи одного бита

Это частный случай протокола для  $m \in \{0, 1\}$ ,  $k(n) = 1$ : противник просит нас подписать бит  $b$ , а потом пробует подписать  $1 - b$ .

**Теорема 9.2.** *Если существует сильная односторонняя функция, то существует и одноразовый протокол подписи одного бита.*

*Доказательство.* Пусть  $f_n : \{0, 1\}^{k(n)} \rightarrow \{0, 1\}^{l(n)}$  сильно односторонняя. Сгенерируем пару  $x_n^0, x_n^1 \leftarrow U_{k(n)}$  и положим  $e_n = (f(x_n^0), f(x_n^1))$ ,  $d_n = (x_n^0, x_n^1)$ . Тогда алгоритм  $S(e_n, d_n, \sigma)$  выдаёт  $x_n^\sigma$ , а алгоритм  $V((e_n^0, e_n^1), c, \sigma)$  будет проверять, что  $e_n^\sigma = f_n(c)$ .

Полнота очевидна по построению. Давайте проверим надёжность. Пусть

$$\begin{aligned}\sigma &= C_1(e_n), \\ c &= S(e_n, d_n, \sigma), \\ (\bar{\sigma}, \bar{c}) &= C_2(e_n, c),\end{aligned}$$

где  $\bar{c}$  — корректная подпись для  $\bar{\sigma}$  для бесконечного числа  $n$  с вероятностью не менее  $\epsilon_n \geq 1/p(n)$  для некоторого полинома  $p(n)$ . Цель: научиться обращать  $f_n$  с вероятностью не менее  $\epsilon_n/2$ . Для бесконечного числа  $n$  схема  $C_1$  в случае успеха возвращает  $\sigma \in \{0, 1\}$  с вероятностью не менее  $\epsilon_n/2$ . Не умаляя общности будем считать, что  $\sigma = 0$ . Тогда, если мы хотим обратить  $f_n(x)$ , то пробуем запустить  $C_1, C_2$  на ключе  $(f_n(r), f_n(x))$  для случайного  $r \leftarrow U_{k(n)}$ . Вероятность успеха  $\epsilon_n/2$ .  $\square$

### 9.2. Одноразовый протокол подписи $p(n)$ битов

**Теорема 9.3.** *Если существует одноразовый протокол подписи одного бита, то для любого полинома  $p(n)$  существует одноразовый протокол подписи сообщений из  $p(n)$  битов.*

*Доказательство.* Пусть  $(G, S, V)$  — одноразовый протокол подписи одного бита. Тогда одноразовый протокол подписи сообщений из  $p(n)$  битов  $(\tilde{G}, \tilde{S}, \tilde{V})$  будет устроен так:  $\tilde{G}(1^n) = ((e_n^1, e_n^2, \dots, e_n^{p(n)}), (d_n^1, d_n^2, \dots, d_n^{p(n)}))$ , где каждая пара  $(e_n^i, d_n^i)$  получена из независимого

запуска  $G(1^n)$ . Алгоритм  $S(e_n, d_n, m)$  подписывает сообщение побитово, алгоритм  $V(e_n, m, c)$  проверяет подпись каждого бита. Пусть теперь  $\tilde{C}$  — успешный противник для  $(\tilde{G}, \tilde{S}, \tilde{V})$ , т.е.

$$\begin{aligned} x &= \tilde{C}_1(e_n), \\ (c_1, c_2, \dots, c_{p(n)}) &= \tilde{S}(e_n, d_n, x), \\ (y, c') &= \tilde{C}_2(e_n, c), \end{aligned}$$

где  $c'$  — корректная подпись для  $y \neq x$  с вероятностью не менее  $\epsilon_n$  для бесконечного числа  $n$ . Построим взломщик  $C$  для  $(G, S, V)$ , который успешен с вероятностью как минимум  $\epsilon_n/p(n)$ .

Пусть  $I \subseteq \mathbb{N}$  — множество индексов  $n$ , на которых  $\tilde{C}$  успешен. Для любого  $n$  существует  $i \in [p(n)] : \Pr[\tilde{C} \text{ успешен} \wedge x_i \neq y_i] \geq \epsilon_n/p(n)$ . Это  $i$  для каждого  $n$  можно записать в схему. Противник  $C_1(e_n)$  генерирует пары  $(e_n^j, d_n^j)$  при помощи  $G$  для всех  $j \neq i$  и создаёт ключ  $e$ , в котором на позиции  $i$  стоит  $e_n$ . Этот ключ передаётся  $\tilde{C}_1$ , в результате чего получается сообщение  $x$ . Подпись для  $x_i$  спрашивается у алгоритма  $S$ , подпись для всех оставшихся битов вычисляется по известным  $(e_n^j, d_n^j)$ . После этого нужно вызвать  $\tilde{C}_2$  и взять  $i$ -ую координату ответа.  $\square$

### 9.3. Одноразовый протокол подписи полиномиального числа битов

Нам потребуется дополнительный примитив.

**Определение 9.2.** Семейство хеш-функций с трудно обнаружимыми коллизиями (СТОК) — это последовательность полиномиально вычислимых функций  $h^n : \{0, 1\}^{l(n)} \times \{0, 1\}^* \rightarrow \{0, 1\}^{k(n)}$ , где  $l(n)$  и  $k(n)$  — некоторые полиномы. Кроме того задана доступная случайная величина  $\alpha_n$  на строках длины  $l(n)$ . Будем обозначать  $h_t^n(x) = h^n(t, x)$ . Для любого полиномиального противника  $C$  вероятность  $\Pr_{t \leftarrow \alpha_n}[C(t) = (x, y) \mid h_t^n(x) = h_t^n(y)]$  пренебрежимо мала.

**Определение 9.3.** Семейство перестановок с трудно обнаружимыми зацеплениями (СТОЗ) — это семейство полиномиально вычислимых функций  $f^n : \{0, 1\} \times \{0, 1\}^{l(n)} \times \{0, 1\}^{m(n)} \rightarrow \{0, 1\}^{k(n)}$ , где  $l(n)$ ,  $m(n)$  и  $k(n)$  — некоторые полиномы. Кроме того задана доступная случайная величина  $\alpha_n$  на строках длины  $l(n)$ . Будем обозначать  $f_t^{(b)}(x) = f^n(b, t, x)$ . Для любого  $t \in \text{rng } \alpha_n$  функции  $f_t^{(0)}$  и  $f_t^{(1)}$  являются перестановками множества  $D_t \subseteq \{0, 1\}^{m(n)}$ . Существует полиномиальный алгоритм, который по  $n$  и  $t$  выдаёт элемент  $D_t$ . И для любого полиномиального схемного противника  $C$  вероятность  $\Pr_{t \leftarrow \alpha_n}[C(t) = (x, y) \mid f_t^{(0)}(x) = f_t^{(1)}(y)]$  пренебрежимо мала.

**Теорема 9.4.** Если существует СТОЗ, то существует и СТОК.

*Доказательство.* Пусть  $f : \{0, 1\} \times \{0, 1\}^{l(n)} \times \{0, 1\}^{m(n)} \rightarrow \{0, 1\}^{k(n)}$  — СТОЗ. Пусть  $\beta_n$  выдаёт элемент из  $D_t$ . Определим  $g : \{0, 1\}^{l(n)} \times \{0, 1\}^* \rightarrow \{0, 1\}^{k(n)}$  следующим образом: для  $x$  длины  $m$  положим

$$g_t(x) = f_t^{x_1} \circ f_t^{x_2} \circ \dots \circ f_t^{x_m}(\beta_t).$$

Пусть противник выдаёт пару строк  $(x, y)$  так, что никакая из строк не является началом другой, и  $g_t(x) = g_t(y)$  с вероятностью хотя бы  $\epsilon_n$ . Существует  $i$ ,  $i \leq |x|$  и  $i \leq |y|$ :  $x_i \neq y_i$ . Тогда

$$\begin{aligned} g_t(x) &= f_t^{x_1} \circ f_t^{x_2} \circ \dots \circ f_t^{x_i} \circ \dots \circ f_t^{x_{|x|}}(\beta_t), \\ g_t(y) &= f_t^{x_1} \circ f_t^{x_2} \circ \dots \circ f_t^{\bar{x}_i} \circ \dots \circ f_t^{y_{|y|}}(\beta_t), \end{aligned}$$

Так как функции  $f_t^{(0)}$  и  $f_t^{(1)}$  — перестановки, то мы нашли зацепление:  $z_1 = f_t^{x_{i+1}} \circ \dots \circ f_t^{x_{|x|}}(\beta_t)$ ,  $z_2 = f_t^{y_{i+1}} \circ \dots \circ f_t^{y_{|y|}}(\beta_t)$ ,  $f_t^{x_i}(z_1) = f_t^{\bar{x}_i}(z_2)$ . Остаётся понять, как можно избавиться от

требования, чтобы  $x$  и  $y$  не были началом друг друга. Это можно сделать при помощи префиксных кодов.  $\square$

**Теорема 9.5.** *Если функция Рабина односторонняя, то существует СТОЗ.*

*Доказательство.*  $R(x, y) = x^2 \bmod y$ , где  $x$  — квадратичный вычет, а  $y = p \cdot q$ , где  $p$  и  $q$  простые вида  $4k+3$ . Определим  $f_{x,y}^{(0)}(z) = z^2 \bmod y$  и  $f_{x,y}^{(1)}(z) = xz^2 \bmod y$ . Если мы научимся находить зацепления, то научимся и обращать. Действительно, если  $f_{x,y}^{(0)}(z) = f_{x,y}^{(1)}(t)$ , то  $z^2 \equiv xt^2 \bmod y$ , а следовательно  $x = z^2/t^2$ .  $\square$

**Теорема 9.6.** *Если существует СТОК  $g_n : \{0, 1\}^{l(n)} \times \{0, 1\}^* \rightarrow \{0, 1\}^{p(n)}$  и одноразовый протокол подписи сообщений длины  $p(n)$ , то существует одноразовый протокол подписи произвольной длины.*

*Доказательство.* По протоколу  $(G, S, V)$  для подписи сообщений длины  $p(n)$  построим протокол  $(\tilde{G}, \tilde{S}, \tilde{V})$  для сообщений произвольной длины. Положим  $\tilde{G} = G$ ,  $\tilde{d}_n = d_n$ ,  $\tilde{e}_n = (e_n, t)$ . Алгоритм  $\tilde{S}(d_n, e_n, t, x) = S(d_n, e_n, g_t(x))$ . Алгоритм проверки  $\tilde{V}$  устроен аналогично.

Пусть  $\tilde{C}$  — взломщик:

$$\begin{aligned} x &= \tilde{C}_1(e_n, t), \\ c &= \tilde{S}(e_n, d_n, t, x) = S(e_n, d_n, g_t(x)), \\ (y, c') &= \tilde{C}_2(e_n, t, x, c), \end{aligned}$$

где  $c'$  — корректная подпись для  $y \neq x$  с вероятностью не менее  $\epsilon_n$  для бесконечного числа  $n$ . Рассмотрим два события:  $g_t(x) = g_t(y)$  и  $g_t(x) \neq g_t(y)$ . Хотя бы одно из событий будет бесконечное число раз встречаться с вероятностью не менее  $\epsilon_n/2$ . В первом случае существует взломщик для СТОК (генерируем  $e_n$  и  $d_n$  самостоятельно), во втором — для протокола подписи сообщений длины  $p(n)$  (сообщениями будут  $g_t(x)$  и  $g_t(y)$ ).  $\square$

## Список литературы

- [1] Н.К. Верещагин. *Курс лекций “Теоретико-сложностные проблемы криптографии”*, МГУ, <http://lpcs.math.msu.su/~ver/teaching/cryptography/index.html>.
- [2] Д.М. Ицксон. *Курс “Теоретико-сложностные основы криптографии”*, CS центр, <https://compsclub.ru/courses/cryptography-foundations/2016-spring/>.
- [3] O. Goldreich. *Foundations of cryptography*.
- [4] J. Håstad, R. Impagliazzo, L.A. Levin, M. Luby. *A Pseudorandom Generator from any One-way Function*. SIAM J. Comput. 28, 4 (March 1999), 1364-1396.  
DOI: <https://doi.org/10.1137/S0097539793244708>
- [5] J. Katz, Y. Lindell. *Introduction to Modern Cryptography*.

## Todo list