„Теоретико-сложностные основы криптографии“. Заметки к курсу в СПбАУ

А.В. Смаль

21 февраля 2018 г.

Аннотация

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

Содержание

Введение

Мы будем предполагать, что алгоритмы шифрования/дешифрования всем известны (т.е. no 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 ($\mathcal{K}$ — 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 $\mathcal{K} = \{0, 1\}^{n-1}$. Возьмём в качестве $m_0 = 0^n$. Пусть $S = \{E(k, 0^n) \mid k \in \mathcal{K}\}$. 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 $t_m = \left|\{k \mid E(k, m) \in S\}\right|$. Если существует сообщение m^*, для которого $t_{m^*} \leq 2^{n-1}$, 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. □

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 $f_n : \{0,1\}^{k(n)} \to \{0,1\}^{l(n)}$, 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 $f_n : \{0,1\}^{k(n)} \to \{0,1\}^{l(n)}$ 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 $f_n : \{0,1\}^{k(n)} \to \{0,1\}^{l(n)}$ 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 λ — 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 $f^{-1}(x, 1)$.

$\square$

2.2. Односторонние функции для алгоритмов

Мы будем определять *односторонние функции* 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 существует такой полином 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 существует такой полином 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 необратимым даже для равномерного противника. Более того, существует детерминированный алгоритм, который для всех x по n и $f_n(x)$ за полиномиальное от n время находит некоторый прообраз $f_n(x)$ длины $k(n)$.*

Доказательство. См. пункт "$\Rightarrow$" доказательства теоремы 2.1. $\square$

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 $x_1, \ldots, x_n$, а последний блок интерпретируется как подмножество $[n]$. Тогда $f_n(x) = \left\langle x_1, x_2, \ldots, x_n, \sum_{i \in I} x_i \right\rangle$.

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:

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

т.е. $F_n : \{0,1\}^{N \cdot k(n)} \to \{0,1\}^{N \cdot l(n)}$. 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 $\big(1 - 1/p(n)\big)^N$. 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 $f_n(x_1), f_n(x_2), \ldots, f_n(x_N)$. 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 $x_2, x_3, \ldots, x_N$ и запустим алгоритм R_F на входе $\langle y, f_n(x_2), f_n(x_3), \ldots, f_n(x_N)\rangle$. 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 $x_1, \ldots, x_N$. 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 $\langle f_n(x_1), \ldots, f_n(x_{i-1}), y, f_n(x_{i+1}), \ldots, f_n(x_N)\rangle$ для случайных $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_N$;[1]

- будем повторять каждую итерацию M раз для различных случайных независимых наборов $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_N$.

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

Введём обозначение $s_i(x)$ — вероятность того, что алгоритм R_F найдёт прообраз $\langle f_n(x_1), \ldots, f_n(x_{i-1}), f_n(x), f_n(x_{i+1}), \ldots, f_n(x_N)\rangle$ для случайных $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_N$. 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 ϵ, который мы выберем дальше. Долю трудных "трудных" слов среди всех слов длины n мы обозначим через δ, 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 большее $1/p(n)$. Из этого будет следовать, что доля трудных слов δ довольно большая (больше некоторого обратного полинома). А раз трудных слов много, то

[1] 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 $1/2$.

и вероятность успеха R_F не может быть больше $1/q(n)$. Таким образом мы придём к противоречию. Для реализации этого плана потребуется 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.

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

□

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

Доказательство. Оценим вероятность успеха сверху по формуле полной вероятности:

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

□

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 $1 - p(n)$. Положим $M = n/\epsilon$. Тогда второе слагаемое в лемме 2.1 будет порядка e^{-n}, что при при достаточно больших n меньше, чем $\frac{1}{2p(n)}$. Таким образом $\delta > \frac{1}{2p(n)}$.

Теперь мы хотим определить N и ϵ 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 ϵ 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 ϵ 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.

□

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 формализации этого определения нам описать, как будет устроено равномерное распределение на D_n.

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

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

Определение 2.9. *Статистическим расстоянием* между распределениями вероятностей μ и ν 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 μ_n и ν_n называются *статистически неотличимыми*, если статистическое расстояние между ними стремится к нулю быстрее любого обратного полинома от n при $n \to \infty$.

Определение 2.11. Случайные величины α_n и β_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 μ_n называется *доступным*, если оно статистически неотличимо от некоторого полиномиально моделируемого распределения ν_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 существует такой полином q, что 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*, 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 xyz, 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 xyz равно конкатенации слов $(x^z \bmod y)$, y и z.

3. *Дискретная экспонента.* Функция определена на словах вида xyz, 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 xyz 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 $f_n : \{0,1\}^{k(n)} \to \{0,1\}^{l(n)}$ 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.

Определение 2.15 (trapdoor permutation)**.** 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, d будут называться *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 $\mathrm{poly}(n)$. Последовательность пар $\langle\langle e_n, d_n\rangle, \{f_n^e\}\rangle$ будем называть *односторонней перестановкой с секретом*, если выполнены следующие условия.

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

2. (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 независимо от e).

3. (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 равномерного распределения случайной величина в D_n^e 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
   $$\langle e_n, \text{случайный элемент } D_n^e\rangle.$$

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 приблизительно равной 1 вычисляет x. Здесь пара $\langle e, d\rangle$ выбирается генератором случайный величины $\langle e_n, d_n\rangle$, а x выбирается равномерно в D_n^e.

Замечание 2.3. 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.

3. Генераторы псевдослучайных чисел

3.1. Вычислительно неотличимые случайные величины

Определение 3.1 (Для неравномерного противника)**.** Случайные величины α_n и β_n, зависящие от натурального параметра n, со значениями в множестве слов некоторой длины $l(n)$ называются вычислительно неотличимыми, если для любой последовательности схем $C_0, C_1, \ldots, C_n, \ldots$ размера $\mathrm{poly}(n)$ (с $l(n)$ входами и одним выходом) вероятность событий $C_n(\alpha_n) = 1$ и $C_n(\beta_n) = 1$ отличаются на пренебрежимо малую величину.

Схема C_n в этом контексте называется *тестом* и мы говорим, что случайная величина α_n проходит тест C_n, если $C_n(\alpha_n) = 1$. Таким образом, мы требуем, чтобы α_n и β_n проходили любые тесты полиномиального размера с приблизительно равной вероятностью.

Определение 3.2 (Для равномерного противника)**.** Случайные величины α_n и β_n, зависящие от натурального параметра n, со значениями в множестве слов некоторой длины $l(n)$ называются вычислительно неотличимыми, если для 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. Алгоритм T в этом контексте называется *тестом* и мы говорим, что случайная величина α_n проходит тест T, если $T(1^n, \alpha_n) = 1$.

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

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

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

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

3. *Если α_n и β_n вычислительно неотличимы, а C_n — последовательность вероятностных схем полиномиального 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 α_n, β_n и γ_n имеют совместное распределение. И пусть для любой последовательности значений c_n случайной величины γ_n случайные величины $(\alpha_n \mid \gamma_n = c_n)$ 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 α_n, γ_n, причём случайная величина должны 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 полиномального размера для случайных величин α_n и β_n.

4. Допустим, что существует последовательность схем-тестов T_n полиномиального размера такая, что вероятность $T_n(\alpha_n\gamma_n) = 1$ $T_n(\beta_n\gamma_n) = 1$ отличается $\epsilon = 1/\mathrm{poly}(n)$ для бесконечно многих n. Разность вероятностей событий $T_n(\alpha_n\gamma_n) = 1$ $T_n(\beta_n\gamma_n) = 1$ 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 γ_n). Поэтому для бесконечно многих n найдётся $c = c_n$ в множестве значений γ_n, для которой разность вероятности не меньше ϵ. 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* $k(n) \to l(n)$ будем называть семейство функций $G_n : \{0,1\}^{k(n)} \to \{0,1\}l(n)$, 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 распределенной на всех словах длины $l(n)$.

Определение 3.4. *Генератор псевдослучайных чисел типа* $k(n) \to \infty$ будем называть семейство отображений $G_n : \{0,1\}^{k(n)} \to \{0,1\}^*$, 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 $k(n) \to \infty$ можно построить генератор ПСЧ типа $k(n) \to l(n)$ для любого полинома $l(n)$ (нужно взять первый $l(n)$ битов).

Теорема 3.1. *Если существует генератор ПСЧ G_n типа $k(n) \to l(n)$, 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). $\square$

Обратное утверждение тоже верно.

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

Мы же докажем более простое утверждение.

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

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 (γ_n, β_n), где β_n распределена на $\{0, 1\}$ будем говорить, что β_n является *вычислительно трудной* относительно γ_n, если для любого полиномиального противника B

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

для любого полинома p для достаточно больших n.

Теорема 3.4. *Случайная величина β_n вычислительно трудна относительно γ_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 β_n по γ_n с хорошей вероятностью, т.е.

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

где ϵ_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 ϵ_n — обратный полином. Построим противника B, который предсказывает β_n по γ_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.

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.

$\square$

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

Утверждение 3.1. *Пусть $f_n : D_n \to D_n$ — односторонняя перестановка с трудным битом $h_n : D_n \to \{0,1\}$. 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 $rf_n(x)$. Осталось применить теорему 3.4 для $rf_n(x)$ и $h_n(x)f_n(x)$. $\square$

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

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

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

Доказательство. Доказательство „гибридным“ методом.

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

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

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

Можно воспользоваться тем, что x распределён также как $f_n(x)$ и переписать T_{i+1}

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

Используя противника, который отличает эти две случайные величины мы можем отличать $h_n(f)f_n(x)$ от $rf_n(x)$ — по этим случайным величинам можно детерминированным алгоритмом построить T_i и T_{i+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.

$\square$

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

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

Тогда g_n — односторонняя перестановка с трудным битом h_n.

Определение 3.7. *Код Уолша-Адамара* — это код исправляющий ошибки $WH : \{0,1\}^k \to \{0,1\}^{2^k}$, 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 $y \odot r^{\oplus i}$. Т.е. все $\{0,1\}^k$ можно разбить на пары, различающиеся в одном бите, то все пары строк имеют коды, отличающиеся ровно в половине всех битов. $\square$

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 Хеммнига $\Delta(WH(x), s) \leq (1/2 - \epsilon)2^m$). Существует вероятностный алгоритм A^s со временем работы $\mathrm{poly}(m, \frac{1}{\epsilon})$, 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 ϵ_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

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

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

$$\Pr_x[C(f_n(x)) = x] \geq \Pr_x[C(f_n(x)) = x \mid x \in M] \cdot \Pr_x[x \in M] \geq \frac{1}{2} \cdot \frac{\epsilon_2}{2} = \frac{\epsilon_n}{4}.$$

□

Доказательство леммы 3.2. Будем рассматривать код Уолша-Адамара как таблицу истинности некоторой функции $f : \{0,1\}^m \to 1$. Пусть $WH(x)$ соответствует f, а кодовое слово s — некоторой функции $\tilde{f}$. Таким образом для восстановления x нам нужно вычислить f на входах $100\cdots0$, $010\cdots0$, $001\cdots0$,… Действительно,

$$\begin{aligned}&x_1 = f(100\cdots0)\\ &x_2 = f(100\cdots0)\\ &\vdots\\ &x_n = f(000\cdots n)\end{aligned}$$

Для того, чтобы вычислить $f(z)$ выберем случайный r и вычислим $f(z) = f(r) + f(z+r)$. Проблема в том, что у нас есть доступ только к $\tilde{f}$, т.е. мы можем попробовать вычислять $f(z) = \tilde{f}(r) + \tilde{f}(z+r)$. 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 f случается с большей вероятностью.

Если бы у нас был доступ к истинному значению, то тогда бы тоже всё сработало $f(z) = f(r) + \tilde{f}(z+r)$, но доступа к f у нас нет. Давайте попытаемся вычислить верное $f(r)$: выберем случайные $r_1, r_2, \ldots, r_N$ и вычислим $\mathrm{Maj}_i\{f(r_i) + f(r + r_i)\}$.

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

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

Давайте выберем $N = 2^k - 1$. Выберем $t_1, t_2, \ldots, t_k \leftarrow U_m$ — независимые случайные строки. Из этих случайных строк можно сгенерировать N попарно независимых следующим образом:

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

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. Выберем $N = 2^k - 1$. Сгененируем независимые $t_1, t_2, \ldots, t_k \leftarrow U_m$. Для всех $a_1, \ldots, a_k \in \{0, 1\}$ добавим в список строку $x_1, \ldots, x_m$, где

$$x_i = \mathrm{Maj}_J \Big\{ \bigoplus_{j \in J} a_j + \tilde{f}(0 \cdots \underset{i}{1} \cdots 0 + r^J) \Big\},$$

где $r_J = \bigoplus_{i \in J} t_i$. Для успеха x_i должен быть верен с вероятностью не менее $1 - \frac{1}{10m}$. Определим случайные величины $\{X_J\}$ такие, что

$$X_J = 1 \iff \tilde{f}(0 \cdots \underset{i}{1} \cdots 0 + r^J) = f(0 \cdots \underset{i}{1} \cdots 0 + r^J).$$

По закону больших чисел вероятность того, что при вычислении Maj мы получим более половины неправильных значений $\tilde{f}$ не более $\frac{1}{\epsilon^2 N}$. Т.е. требуется, что $\frac{1}{\epsilon^2 N} \leq \frac{1}{10m}$.

$$\Pr[x_i \text{ вычислен не верно}] = \Pr\Big[\frac{1}{N} \sum X_J \leq \frac{1}{2}\Big] \leq \frac{1}{\epsilon^2 N}.$$

Таким образом $N > \frac{10m}{\epsilon^2}$. □

4. Протоколы с секретным ключом

Определение 4.1. Пусть дана доступная случайная величина d_n. Пусть даны два вероятностных полиномиальных по времени алгоритма $E(d_n, x)$ и $D(d_n, m)$ такие, что $D(d_n, E(d_n, x)) = x$, будем называть *одноразовым протоколом с секретным ключом*. Будем говорить, что этот протокол *надёжный*, если для любого полинома k, последовательности x_n слов длины $k(n)$ и последовательности y_n длины $k(n)$ 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 $G : \{0, 1\}^n \to \{0, 1\}^{k(n)}$. Возьмём $d_n = U_n$ и для $x \in \{0, 1\}^{k(n)}$ определим

$$E(d_n, x) = G(d_n) \oplus x,$$
$$D(d_n, m) = G(d_n) \oplus m.$$

Если бы противник научился бы 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 $U_{k(n)}$. Пусть он умеет отличать $E(d_n, x_n)$ от $U_{k(n)}$. Тогда мы можем зашить x_n в схему и таким образом научиться отличать образ $G(d_n)$ от $U_{k(n)}$.

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

1. Существует полиномиальный по времени алгоритм, который по (s, x) вычисляет $f_s(x)$.

2. Распределение $U(S_n)$ доступно.

3. (*слабая надёжность*) Для любого полинома p и для любого набора различных $t_1, \ldots, t_{p(n)} \in \{0,1\}^n$ случайная величина $f_s(t_1)f_s(t_2)\cdots f_s(t_{p(n)})$ 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:

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

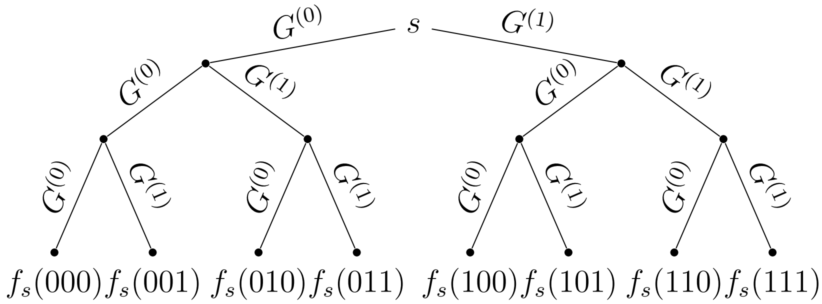
Тогда случайная величина $y_1 y_2 \cdots y_{p(n)}$ вычислительно неотличима от $U_{np(n)}$ (при условии, что все t_i различны).

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

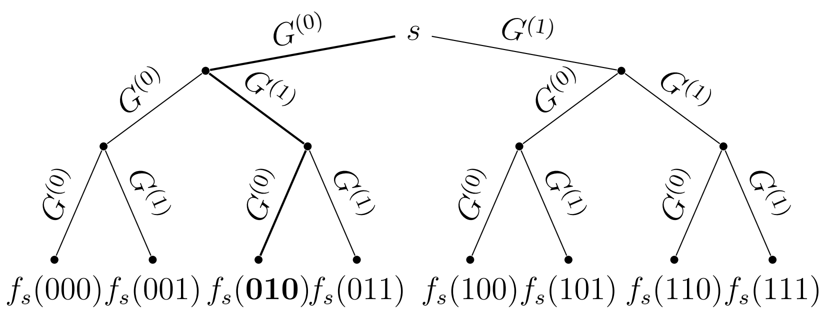
Доказательство. Пусть $G_n : \{0,1\}^n \to \{0,1\}^{2n}$. Для $S_n = \{0,1\}^n$ определим $\{f_s\}_{s \in S_n}$ следующим образом. Определим функции $G_n^{(0)}, G_n^{(1)} : \{0,1\}^n \to \{0,1\}^n$ исходя из следующего соотношения: $G_n(r) = G_n^{(0)}(r)\, G_n^{(1)}(r)$. Теперь определим

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

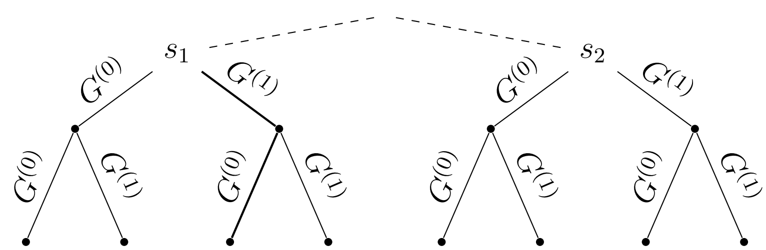
Можно представить вычисление f_s на всевозможных x в виде бинарного дерева: в корне записано число s, если в вершине записано число z, то в его наследниках будут записаны $G^{(0)}(z)$ и $G^{(1)}(z)$, в листьях дерева окажутся все возможные $f_s(x)$.



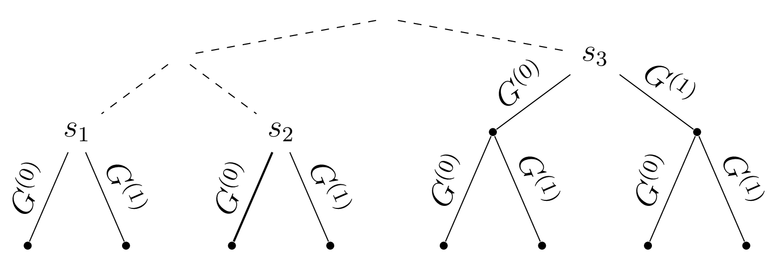
Будем доказывать гибридным методом. Для этого нам нужно построить последовательность распределений, которые постепенно сходятся к равномерному. Изначальное распределение задаёт дерево T, в корне которого выбирается s, а дальше вычисление происходит детерминировано. Каждое t_i задаёт некоторый путь от 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 $U(S_n)$.



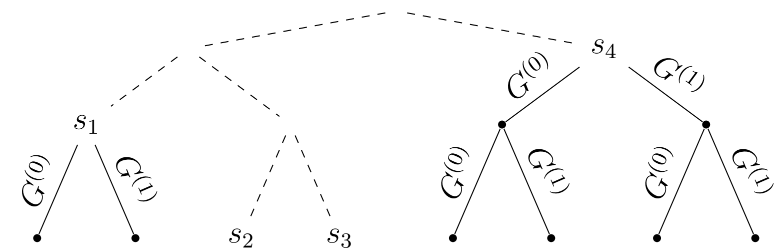
Для примера $t_1 = 010$.



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 $U(S_n)$.



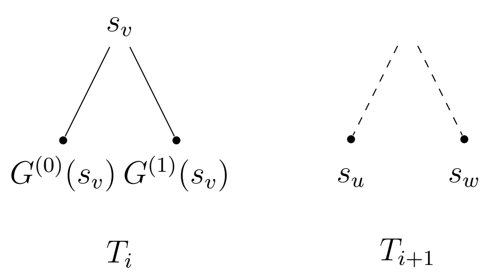
Удаляем последнюю вершину и заменяем сыновей на случайные элементы из $U(S_n)$.

В конце этого процесса в вершине, которая раньше соответствовала t_1 будет некоторый $s_j \leftarrow U(S_n)$. Мы повторяем этот процесс для всех t_i. Таким образом получается последовательность $n \cdot p(n)$ деревьев, которые задают распределения $\{T_i\}_{i=1}^{np(n)}$. В последнем распределении $T_{np(n)}$ значение для каждого $f_n(t_i)$ выбирается из $U(S_n)$.

Если есть противник, который отличает первое распределение от последнего, т.е. конкретные два последовательных распределения, которые этот противник различает с вероятностью.

$$\Pr_{z \leftarrow T_1}[B(z)=1] - \Pr_{z \leftarrow T_{np(n)}}[B(z)=1] \geq \epsilon_n \implies \exists i : \Pr_{z \leftarrow T_i}[B(z)=1] - \Pr_{z \leftarrow T_{i+1}}[B(z)=1] \geq \frac{\epsilon_n}{np(n)}.$$

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 $d_n \in \{0,1\}^{k(n)}$. *Многоразовый протокол с секретным ключом* — это пара полиномиальный по времени вероятностных алгоритмов $E(d_n, x)$ и $D(d_n, m)$ таких, что $D(d_n, E(d_n, x)) = x$ и выполняются следующее условие надёжности.

а) (*слабая надёжность*) Для любых полиномов p и q и для любых $x_1, \ldots, x_{p(n)} \in \{0,1\}^{q(n)}$ случайные величины $E(d_n, x_1) \cdots E(d_n, x_{p(n)})$ и $E(d_n, 0^{q(n)}) \cdots E(d_n, 0^{q(n)})$ вычислительно неотличимы.

b) (*сильная надёжность*) Для любых полиномов p и q, любого $x_1 \in \{0,1\}^{q(n)}$ и семейства схем $\{C_i\}_{i=1}^{p(n)-1}$ полиномиального размера определим $\{x_i\}_{i=2}^{p(n)}$ следующим образом:

$$\begin{aligned} x_2 &= C_1(x_1, E(d_n, x_1)), \\ x_3 &= C_2(x_1, E(d_n, x_1), E(d_n, x_2)), \\ x_4 &= C_3(x_1, E(d_n, x_1), E(d_n, x_2), E(d_n, x_3)), \\ &\vdots \\ x_{p(n)} &= C_{p(n)-1}(x_1, E(d_n, x_1), \ldots, E(d_n, x_{p(n)-1})). \end{aligned}$$

Случайные величины $E(d_n, x_1) \cdots E(d_n, x_{p(n)})$ и $E(d_n, 0^{q(n)}) \cdots E(d_n, 0^{q(n)})$ должны быть вычислительно неотличимы.

Теорема 4.2. *Если существует семейство псевдослучайных функций, то существует многоразовый протокол с секретным ключом.*

Доказательство. Докажем для сообщения из одного бита. Из этого будет следовать, что есть протокол для сообщения любой длины. Пусть $f_s : \{0,1\}^n \to \{0,1\}$ — семейство псевдослучайных функций, $s \in \{0,1\}^n$. Определим случайную величину d_n следующим образом: $d_n = (s_n, z_n)$, где $s_n, z_n \leftarrow U_n$. Алгоритм шифрования $E(d_n, m)$ для ключа s_n и бита $m \in \{0,1\}$ выбирает $z \leftarrow U_n$ и выдаёт $(m \oplus f_{s_n}(z), z)$. Алгоритм дешифровки $D(s_n, y)$, где $y \in \{0,1\}^{n+1}$, $y = bz$, выдаёт $b \oplus f_{s_n}(z)$. Корректность протокола очевидна. Нужно доказать многоразовую надёжность.

Разобраться с обозначениями для d_n.

Пусть есть сообщения $m_1, \ldots, m_{p(n)}$. Их коды соответственно

$$f_{s_n}(z_1) \oplus m_1, \ldots f_{s_n}(z_{p(n)}) \oplus m_{p(n)}$$

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

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

Пусть $m \in \{0,1\}^{k(n)}$. Нам потребуется семейство псевдослучайных функций $f_s : \{0,1\}^n \to \{0,1\}^n$. Тогда алгоритма шифрования может работать так: выберем случайное $z \leftarrow U_n$. Пусть $G : \{0,1\}^n \to \{0,1\}^{k(n)}$ — ГПСЧ. Определим шифрование так $E(s_n, m) = (G(f_{s_n}(z)) \oplus m, z)$.

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). □

5. Протоколы с открытым ключом

Определение 5.1. Пусть (d_n, e_n) — доступные случайные величины. Пара вероятностных полиномиальных алгоритмов (E, D) называется *системой шифрования с отрытым ключом*, если Алгоритмы шифрования и дешифрования $E(e_n, m)$ и $D(e_n, d_n, c)$ связанны следующим соотношением (корректность): $E(e_n, d_n, E(e_n, m)) = m$. Надёжность протокола определяется следующим образом: для любого полинома p и любой последовательности сообщений $x_n, y_n \in \{0,1\}^{p(n)}$ случайные величины $e_n E(e_n, x_n)$ и $e_n E(e_{n,}{}_{n})$ вычислительно неотличимы.

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

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. Ицыксон. *Курс "Теоретико-сложностные основы криптографии"*, CS центр, https://compsciclub.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