Содержание

Крипта ИСП

Disclaymer: доверять этому конспекту или нет выбирайте сами

1. О криптографии

Определение 1.1: **Криптографические средства защиты информации (КСЗИ)** – основанные на математических методах преобразования защищаемой информации.

Определение 1.2: **Теоретическая криптография (математическая криптография, криптология)** – раздел дискретной математики, изучающий математические модели КСЗИ с научной точки зрения.

Основной предмет теоретической криптографии – криптографический протокол. (о нём в следующей главе).

Пример: **Криптографические примитивы**:

- **Односторонняя функция** – эффективно вычислимая функция, задача инвертирования которой вычислительно трудна.
- **Псевдослучайный генератор** – эффективный алгоритм, генерирующий длинные последовательности, которые никакой эффективный алгоритм не отличит от чисто случайных.
- **Криптографическая хэш-функция** – эффективно вычислимое семейство функций, уменьшающих длину аргумента, для которого задача поиска коллизий вычислительно трудна.

Определение 1.3: **Атака** – совокупность предположений о возможностях противника, о том, какие действия ему доступны (помимо вычислений).

Определение 1.4: **Угроза** – цель противника, состоящая в нарушении одного или нескольких из трёх условий (задач) криптографического протокола.

2. О криптографических протоколах

Определение 2.1: **Криптографический протокол** – это протокол, решающий хотя бы одну из трёх задач:
- Обеспечение **конфиденциальности** данных
- Обеспечение **целостности** сообщений и системы в целом – гарантия отсутствия нежелательных последствий вмешательства противника
- Обеспечение **неотслеживаемости** – невозможность установления противником, кто из участников выполнил определённое действие

Пример: **Прикладные КП**:
- Системы шифрования
- Подбрасование монеты по телефону
- Схемы электронной подписи
- Протоколы аутентификации
- Системы электронных платежей

Пример: **Примитивные КП**:
- bit-commitment (схема обязательства)
- oblivious transfer (протокол с забыванием)

Определение 2.2: **Стойкость** – формализация понятия качества криптографического протокола, его способность решать поставленную перед ним задачу.

Замечание 2.1: Стойкость определяется **только** для конкретной модели противника, состоящей из трёх основных компонентов:
- Вычислительные ресурсы (включая модель вычислений)
- Атака
- Угроза

3. О теории сложности

Замечание 3.1: Задача кодируется множеством строк в некотором конечном алфавите $\Sigma, |\Sigma| \geq 2$. Без ограничения общности, будем рассматривать только $\Sigma = \{0, 1\} = \mathbb{B}$.

Определение 3.1: Σ^* – множество всех слов в алфавите Σ, то есть $\Sigma^* = \cup_{n=0}^{\infty} \Sigma^n$.

Определение 3.2: **Язык** – некоторое множество слов, то есть подмножество в Σ^*.

Определение 3.3: Модель вычислений, которую мы будем использовать в дальнейшем – **машина Тьюринга**

$$M = (Q, q_0, q_f, \Sigma, b, \sigma)$$

где:

- Q – множество состояний (конечное, непустое)
- $q_0, q_f \in Q$ – выделенные состояния: начальное и конечное
- Σ – конечный алфавит
- b – специальный «пустой символ»
- $\sigma : \Sigma \times Q \to \Sigma \times Q \times \{-1, 0, 1\}$ – функция перехода (частично определённая, в общем случае многозначная)

Определение 3.4: С машиной Тьюринга M связаны отображения:

- **Вычисляемая машиной функция** $M(\cdot) : \mathbb{B}^* \to \mathbb{B}^* \cup \{\perp\}$, где $M(w)$ – выход машины M, если на вход подана строка w. (Выдаёт $\perp$ если вычисление не закончено)
- **Время её работы** $T_M(\cdot) : \mathbb{B}^* \to \mathbb{N} \cup \{\infty\}$, где $T_M(w)$ – число тактов работы машины M при вычислении на входе w.
- **Используемая ею память** $S_M(\cdot) : \mathbb{B}^* \to \mathbb{N} \cup \{\infty\}$, где $S_M(w)$ – число ячеек ленты, задействованных в вычислении на входе w.

Определение 3.5: Введём $\mathrm{poly}(x)$ – обозначение для «**некоторого полинома**» от переменной x. Важен не сам полином, факт его существования.

Определение 3.6: Введём названия для некоторых видов машин Тьюринга:

- **Детерминированная машина Тьюринга** – функция перехода σ однозначна
- **Полиномиальная (детерминированная) машина Тьюринга M** – обладает свойством:

$$\forall w \in \mathbb{B}^* : T_M(w) \leq \text{poly}(|w|)$$

- **Недетерминированная машина Тьюринга** – функция перехода σ, вообще говоря, многозначна, выбор её значений в конкретном вычислении осуществляется с помощью строки «Недетерминированного выбора» $\psi \in \mathbb{B}^\infty$, записанной на специальную ленту
- **Полиномиальная недетерминированная машина Тьюринга M** – обладает свойством:

$$\forall w \in \mathbb{B}^* : \forall \psi \in \mathbb{B}^* : T_M(\psi; w) \leq \text{poly}(|w|)$$

- **Вероятностная машина Тьюринга** – функция перехода σ принимает случайные значения, $M(w)$ – случайная величина (при фиксированном w). Выбор значения функции перехода в каждом такте осуществляется с помощью случайной строки $\rho \in \mathbb{B}^\infty$, записанной на специальную ленту.
- **Полиномиальная вероятностная машина Тьюринга (п.в.м.Т.) M** – обладает свойством:

$$\forall w \in \mathbb{B}^* : \forall \rho \in \mathbb{B}^\infty : T_M(\rho; w) \leq \text{poly}(|w|)$$

- **Полиномиальная в среднем вероятностная машина Тьюринга M** – обладает свойством:

$$\exists \varepsilon > 0 : \forall n \in \mathbb{N} : \forall \rho \in \mathbb{B}^\infty : \forall w \in \mathbb{B}^n : \mathbb{E}(T_M(\rho; w))^\varepsilon \leq n$$

Определение 3.7: Класс сложностей **Bounded-error Probabilistic Polynomial time**:

$$\text{BPP} = \left\{ L \subseteq \mathbb{B}^* \mid \exists \text{ п.в.м.Т. } M : \begin{cases} w \in L \ \Rightarrow \mu(\{M(w)=1\}) \geq \frac{2}{3} \\ w \notin L \Rightarrow \mu(\{M(w)=1\}) \leq \frac{1}{3} \end{cases} \right\}$$

Определение 3.8: Класс сложностей **Randomzed Polynomial time**:

$$\text{RP} = \left\{ L \subseteq \mathbb{B}^* \mid \exists \text{ п.в.м.Т. } M : \begin{cases} w \in L \ \Rightarrow \mu(\{M(w)=1\}) \geq \frac{2}{3} \\ w \notin L \Rightarrow \mu(\{M(w)=1\}) = 0 \end{cases} \right\}$$

Определение 3.9: **Однородной моделью вычислителя противника** называется полиномиальная вероятностная машина Тьюринга или полиномальная в среднем вероятностная машина Тьюринга.

Определение 3.10: **Булевой схемой** называется отображение $C : \mathbb{B}^n \to \mathbb{B}^m$, такое, что для каждой координаты образа существует логическая функция от входа, тождественно задающая её.

Размером булевой схемы называется размерность её выхода.

Пример: Булевая схема $C : \mathbb{B}^1 \to \mathbb{B}^3$ имеет размер 3:

$$C(x_1) = (x_1, \neg x_1, x_1 \vee \neg x_1)$$

Определение 3.11: **Неоднородной моделью вычислителя противника** называется семейство булевых схем полиномиального размера $C = \{C_n\}_{n=1}^{\infty}$:

$$\forall n : |C_n| \leq \text{poly}(n)$$

причём для каждого размера входа $|w|$ выбирается $C_{|w|}$ схема.

4. Об односторонних функциях

Определение 4.1: Функция $\nu : \mathbb{N} \to \mathbb{R}^+$ называется **пренебрежимо малой**, если

$$\forall \text{ полинома } p : \exists n_0 \in \mathbb{N} : \forall n \geq n_0 : \nu(n) \leq \frac{1}{p(n)}$$

Обозначение: $\text{negl}(n)$.

Определение 4.2: Функция $f : X \to Y; X, Y \subseteq \mathbb{B}^*$ называется **полиномиально вычислимой**, если существует полиномиальная (детерминированная) машина Тьюринга M такая, что

$$\forall x \in X : M(x) = f(x)$$

Замечание 4.1:

- $\mathcal{U}$ – равномерное распределение вероятностей
- $x \underset{\mathcal{U}}{\in} Z$ значит, что x выбран случайно из множества Z в соответствии с равномерным распределением вероятностей на этом множестве
- $y \leftarrow M(x)$ значит, что y – случайный выход в.м.Т. M, на вход которой был подан X.
- Под возведением в степень 0 или 1 имеется в виду декартово умножение

Определение 4.3: Функция $f : \mathbb{B}^* \to \mathbb{B}^*$ называется **(сильно) односторонней**, если

1. f полиномиально вычислима
2.

$$\forall \text{ п.в.м.Т. } A : \mu_{x \underset{\mathcal{U}}{\in} \mathbb{B}^n}(\{A(1^n; f(x)) \in f^{-1}(f(x))\}) = \text{negl}(n)$$

Определение 4.4: Функция $f : \mathbb{B}^* \to \mathbb{B}^*$ называется **слабо односторонней**, если
1. f полиномиально вычислима
2.

$$\exists \text{ полином } p : \forall \text{ п.в.м.Т. } A : \exists n_0 \in \mathbb{N} : \forall n \geq n_0 :$$
$$\mu_{\substack{x \in \mathbb{B}^n \\ U}}(\{A(1^n; f(x)) \in f^{-1}(f(x))\}) \leq 1 - \frac{1}{p(n)}$$

Лемма 4.1: Любую полиномиально вычислимую, а значит и (сильно/слабо) одностороннюю функцию можно преобразовать так, чтобы она сохраняла длину аргумента.

Доказательство:
- Выберем какой-нибудь полином m, существующий в силу полиномиальной вычислимости функции f:

$$\forall x : |f(x)| \leq m(|x|)$$

это верно, так как машина Тьюринга совершит не более некоторого полиномиального числа тактов, а за такт она может прибавить максимум 1 к длине вывода.
- Определим функцию h на множестве $\cup_{n \in \mathbb{N}} \mathbb{B}^{m(n)+1}$, для чего представим каждый x из этого множества в виде $x = x'x''$, где $x' \in \mathbb{B}^n, x'' \in \mathbb{B}^{m(n)+1-n}$, и положим

$$h(x) = f(x') \times 1 \times 0^{m(|x'|)-|f(x')|}$$

Заметим, что вывод теперь имеет такую же длину, как и вход. (Почему нужно добавить единицу, а не все нули?) □

Теорема 4.1 (Яо): Если существует слабо односторонняя функция, то существует и сильно односторонняя функция.

Доказательство: Пусть f – слабо односторонняя функция, БОО предположим, что мы уже преобразовали её к виду, сохраняющему длину входа, то есть

$$\forall n \in \mathbb{N} : f(\mathbb{B}^n) \subseteq \mathbb{B}^n$$

Зафиксируем некоторый полином p из определения слабой односторонности.

Для любой п.в.м.Т. A и для всех достаточно больших n:

$$\mu_{\substack{x \in \mathbb{B}^n \\ U}}(\{A(1^n; f(x)) \in f^{-1}(f(x))\}) \leq 1 - \frac{1}{p(n)}$$

Введём функцию

$$g(x_1, ..., x_t) := (f(x_1), ..., f(x_t)); \quad x_i \in \mathbb{B}^n, t(n) := n \cdot p(n)$$

Предположим, что g – не односторонняя, тогда для произвольного полинома q существует п.в.м.Т. B и бесконечное множество $N \subseteq \mathbb{N}$, что

$$\forall n \in N : \mu_{\substack{x \in \mathbb{B}^{nt(n)} \\ U}}\big(\{B(1^{nt(n)}; g(x)) \in g^{-1}(g(x))\}\big) > \frac{1}{q(nt(n))}$$

Определим вероятностную машину C_0 на входе $y \in \mathbb{B}^n$:

1. for i in [1..t]
2. let z = $B(f(x_1), ..., f(x_{i-1}), y, f(x_{i+1}), ..., f(x_t))$
3. if $f(z_i) = y$: return z_i

Также определим вероятностный алгоритм C на входе y, выполняющий алгоритм C_0 на этом входе $k(n) := 2 \cdot n \cdot t(n) \cdot q(n \cdot t(n))$ раз.

Если на некоторой итерации алгоритм C_0 что-то вернул, то это будет результатом C, иначе C заканчивает работу без выходного значения.

Для произвольного $n \in \mathbb{N}$ положим

$$E_n = \Big\{x \in \mathbb{B}^n \mid \mu(\{C_0(1^n; f(x)) \in f^{-1}(f(x))\}) > \frac{n}{k(n)}\Big\}$$

Где берём те x, при которых вероятность (теперь x фиксирован, случайность осталась лишь в случайном векторе 1^n) обращения f отделима от нуля.

Лемма 4.2:

$$\forall n \in \mathbb{N} : \forall x \in E^n : \mu(\{C(1^n; f(x)) \in f^{-1}(f(x))\}) > 1 - e^{-n}$$

Эта лемма показывает, что ограниченная на E^n f является сильно односторонней.

Доказательство: Зная, что:

- C – применение алгоритма C_0 k раз, а значит если C не угадал прообраз, то и k раз применённый C_0 тоже не угадал. (Оценка вероятности)
- Мы взяли $x \in E_n$, в котором вероятность угадать прообраз алгоритмом $C_0 > \frac{n}{k}$, а значит вероятность не угадать $< 1 - \frac{n}{k}$
- $\forall r : \ln r \le r - 1$

получим:

$$\mu(\{C(1^n; f(x)) \notin f^{-1}(f(x))\}) < \big(1 - \tfrac{n}{k}\big)^k = e^{k \ln\left(1 - \frac{n}{k}\right)} \le e^{-n}$$

□

Лемма 4.3:

$$\exists N_0 \in \mathbb{N} : \forall n > N_0 : \mu(E_n) > 1 - \frac{1}{2p(n)}$$

Этой леммой мы хотим показать, что с какого-то момента E_n достаточно большое.

Доказательство: Пока скип, большое □

Из доказанных лемм вытекает, что

$$\mu(\{C(1^n; f(x)) \in f^{-1}(f(x))\}) \ge$$

$$\mu(\{C(1^n; f(x)) \in f^{-1}(f(x)) \mid E_n\})\mu(E_n) > (1 - e^{-n})\Big(1 - \frac{1}{2p(n)}\Big)$$

Но если вспомним, что f слабо односторонняя, то получим неравенство:

$$1 - \frac{1}{p(n)} > (1 - e^{-n})\left(1 - \frac{1}{2p(n)}\right)$$

Раскрыв скобки в правой части получим, что

$$\frac{1}{p(n)} < e^{-n} + \frac{1}{2p(n)} - \frac{e^{-n}}{2p(n)} < e^{-n} + \frac{1}{2p(n)}$$

Что неверно при достаточно больших n, так как e^{-n} убывает быстрее $\frac{1}{2p(n)}$.

5. О трудных предикатах

Определение 5.1: Функция $\mathbb{B}^* \to \mathbb{B}$ называется **трудным предикатом** для функции $f : \mathbb{B}^* \to \mathbb{B}^*$, если

- b – полиномиально вычислимая функция
- $\forall$ п.в.м.Т. $A : \mu_{\underset{U}{x \in \mathbb{B}^n}}(\{A(1^n; f(x)) = b(x)\}) < \frac{1}{2} + \mathrm{negl}(n)$

Теорема 5.1 (Гольдрайха-Левина): Пусть f – односторонняя функция, определённая всюду и сохраняющая длину, и пусть для всех $x, r \in \mathbb{B}^* : |x| = |r|$, определены функции

$$g(x, r) = (f(x), r) \quad b(x, r) = \oplus_{i=1}^{|x|} x^{[i]} r^{[i]}$$

Тогда b – трудный предикат для функции g.

Доказательство: Предположим, что b не является трудным предикатом для функции g.

Это значит, что существуют полиномиальный вероятностный алгоритм A, полином p и бесконечное множество $N \subseteq \mathbb{N} \setminus \{0\}$ такие, что

$$\forall n \in N : \varepsilon(n) = \mu\left(\left\{A(1^{2n}; f(x), r) = b(x, r)\right\}\right) - \frac{1}{2} > \frac{1}{p(n)}$$

Пусть $n \in N$ и $x \in \mathbb{B}^n$. Положим

$$t(n, x) = \mu\left(\left\{A(1^{2n}; f(x), r) = b(x, r)\right\}\right) \quad E_n = \left\{x \in \mathbb{B}^n \mid t(x) \geq \frac{1}{2} + \frac{\varepsilon(n)}{2}\right\}$$

Тогда, заметив, что

- $\mathbb{E}_x(t(n, x)) = \varepsilon(n) + \frac{1}{2}$ – по определению
- Можно применить неравенство Чебышёва, так как $\frac{1}{2} - \frac{\varepsilon(n)}{2} > 0$.

$$\mu\left(\left\{t(x) < \frac{1}{2} + \frac{\varepsilon(n)}{2}\right\}\right) = \mu\left(\left\{1 - t(x) > \frac{1}{2} - \frac{\varepsilon(n)}{2}\right\}\right) \leq$$

$$\frac{\mathbb{E}_x(1 - t(n,x))}{\frac{1}{2} - \frac{\varepsilon(n)}{2}} = \frac{\frac{1}{2} - \varepsilon(n)}{\frac{1}{2} - \frac{\varepsilon(n)}{2}} = 1 - \frac{\varepsilon(n)}{1 - \varepsilon(n)} < 1 - \varepsilon(n)$$

Воспользовавшись отрицанием обеих частей неравенства, получим

$$\mu(E_n) = \mu\left(\left\{t(x) \geq \frac{1}{2} + \frac{\varepsilon(n)}{2}\right\}\right) > \varepsilon(n) > \frac{1}{p(n)}$$

Для завершения доказательства теоремы достаточно построить полиномиальный вероятностный алгоритм B, определённый для всех n и на $f(E_n)$, такой, что

$$\mu(\{B(1^n; f(x)) = x\}) \geq \frac{1}{\mathrm{poly}(n)}$$

Тогда этой вероятностью мы сможем оценить снизу вероятность угадать прообраз f, что будет противоречить односторонности f.

Введём обозначение $e_i \in \mathbb{B}^n$ – вектор с единицей на i-м месте.

Алгоритм B на входе $(1^n; f(x))$, где $n \in N$ и $x \in E_n$, будет искать каждый бит $x^{[i]}$ отдельно. Для этого алгоритм B:

- Выбирает случайные элементы $r_1, ..., r_{\pi(n)} \underset{U}{\in} \mathbb{B}^n$, где π – некоторый полиномиальный параметр на N, принимающий лишь нечётные значения.
- Для каждого $j \in \{1, ..., \pi(n)\}$ вычисляет биты β_j, ρ_j, являющиеся предполагаемыми значениями $b\big(x, r_j \oplus e_i\big)$ и $b\big(x, r_j\big)$ соответственно
- Выбирает в качестве предполагаемого значения $x^{[i]}$ бит, который встречается в последовательности $\beta_j \oplus \rho_j; j \in \{1, ..., \pi(n)\}$ более $\frac{\pi(n)}{2}$ раз

Очевидно, если $\beta_j = b\big(x, r_j \oplus e_i\big)$ и $\rho_j = b\big(x, r_j\big)$ для более чем половины индексов $j \in \{1, ..., \pi(n)\}$, то $x^{[i]}$ будет найден правильно, так как

$$b\big(x, r_j \oplus e_i\big) \oplus b\big(x, r_j\big) = b(x, e_i) = x^{[i]}$$

Бит β_j вычисляется как $A\big(1^{2n}; f(x), r_j \oplus e_i\big)$. Мы не получим нужную оценку вероятности успеха алгоритма B, если будем вычислять ρ_j как $A\big(1^{2n}; f(x), r_j\big)$. Вместо этого алгоритм пытается угадать значение $b\big(x, r_j\big)$ для всех j.

Но если просто выбрать $\rho_j \underset{U}{\in} \mathbb{B}$, то вероятность того, что $\rho_j = b\big(x, r_j\big)$ для всех $j \in \{1, ..., \pi(n)\}$ будет равна $\frac{1}{2^{\pi(n)}}$, а эта величина при нужном для нас росте $\pi(n)$ будет пренебрежимо малой, как функция от n. Чтобы обойти это препятствие, алгоритм B делает некую грязь. □

6. О вычислительной неотличимости

Определение 6.1: Семейства случайных величин $\{\xi_n\}_{n\in\mathbb{N}}$ и $\{\zeta_n\}_{n\in\mathbb{N}}$ называются **вычислительно неразличимыми**, если для любой п.в.м.т. D:

$$|\mu(\{D(1^n; \xi_n) = 1\}) - \mu(\{D(1^n; \zeta_n) = 1\})| = \text{negl}(n)$$

Замечание 6.1: Равномерно распределённым семейством случайных величин на $\mathbb{B}^n$ будем называть $\{v_n\}_{n\in\mathbb{N}}$:

$$\forall x \in \mathbb{B}^n : \mu(\{v_n = x\}) = \tfrac{1}{2^n}$$

Определение 6.2: Семейство случайных величин $\{\xi_n\}_{n\in\mathbb{N}}$ называется **псевдослучайным**, если оно вычислительно неотличимо от равномерно распределённого семейства случайных величин $\left\{v_{m(n)}\right\}_{n\in\mathbb{N}}$

Определение 6.3: Функция $g : \mathbb{B}^* \to \mathbb{B}^*$, такая, что $g(\mathbb{B}^n) \subseteq \mathbb{B}^{m(n)}$ для некоторого полинома m, называется **псевдослучайным генератором** или, полностью, **криптографически стойким генератором псевдослучайных последовательностей**, если
1. g – полиномиально вычислима
2. $m(n) > n$ для всех $n \in \mathbb{N}$
3. $\{g(v_n)\}_{n\in\mathbb{N}}$ – псевдослучайное семейство случайных величин

7. О предсказании следующего бита

Определение 7.1: Семейство случайных величин $\{\xi_n\}_{n\in\mathbb{N}}$ удовлетворяет условию **непредсказуемости следующего бита**, если для любой п.в.м.Т. P:

$$\mu_{i\underset{U}{\in}\{1,\dots,m(n)\}}\Big(\Big\{P\Big(1^n;\xi_n^{[1,\dots,i-1]}\Big)=\xi_n^{[i]}\Big\}\Big)\le\tfrac{1}{2}+\mathrm{negl}(n)$$

Теорема 7.1 (Яо об эквивалентности): Семейство случайных величин $\{\xi_n\}_{n\in\mathbb{N}}$ псевдослучайно тогда и только тогда, когда $\{\xi_n\}_{n\in\mathbb{N}}$ удовлетворяет условию непредсказуемости следующего бита.

Доказательство: $\Rightarrow$ От обратного, пусть существует п.в.м.Т. P «предсказатель» и полином p:

$$\mu_i\Big(\Big\{P\Big(1^n;\xi_n^{[1,\dots,i-1]}\Big)=\xi_n^{[i]}\Big\}\Big)>\tfrac{1}{2}+\tfrac{1}{p(n)}$$

Построим «различитель» - п.в.м.Т. D, работающую на входах $(1^n;x), x\in\mathbb{B}^{m(n)}$, работающий по алгоритму:
1. Выбираем случайный $i\underset{U}{\in}\{1,\dots,m(n)\}$
2. Если «предсказатель» угадал по $x^{[1,\dots,i-1]}$ битам i-й, то «различитель» возвращает 1, иначе 0.

Рассмотрим вероятность:
- $\mu(\{D(1^n;\xi_n)=1\})=\mu_i\Big(\Big\{P\Big(1^n;\xi_n^{[1,\dots,i-1]}\Big)=\xi_n^{[i]}\Big\}\Big)>\frac{1}{2}+\frac{1}{p(n)}$
- $\mu\Big(\Big\{D\Big(1^n;v_{m(n)}\Big)=1\Big\}\Big)=$

  $\mu\Big(\Big\{P\Big(1^n;v_{m(n)}^{[1,\dots,i-1]}\Big)=v_{m(n)}^{[i]}\Big\}\Big)=$

  $\sum_{k=1}^{m(n)}\mu\Big(\Big\{P\Big(1^n;v_{m(n)}^{[1,\dots,k-1]}\Big)=v_{m(n)}^{[k]},i=k\Big\}\Big)=$

  $\sum_{k=1}^{m(n)}\mu\Big(\Big\{P\Big(1^n;v_{m(n)}^{[1,\dots,k-1]}\Big)=v_{m(n)}^{[k]}\Big\}\Big)\mu(\{i=k\})=m(n)\cdot\frac{1}{2}\cdot\frac{1}{m(n)}=\frac{1}{2}$

Разность этих вероятностей $>\frac{1}{p(n)}$ для бесконечно многих n – противоречие.
$\Leftarrow$ От противного. Предположим, $\{\xi_n\}_{n\in\mathbb{N}}$ и $\Big\{v_{m(n)}\Big\}_{n\in\mathbb{N}}$ не вычислимо неразличимы: существует такая п.в.м.Т. D «различитель» и полином p, что для бесконечно многих n:

$$\left|\mu(\{D(1^n;\xi_n)=1\})-\mu\Big(\Big\{D\Big(1^n;v_{m(n)}\Big)=1\Big\}\Big)\right|>\frac{1}{p(n)}$$

Построим «предсказатель следующего бита» – п.в.м.Т. P, работающую на входах $(1^n;x), x\in\mathbb{B}^{<m(n)}$, следующим образом:

1. Выбираем случайный $y\underset{\mathcal{U}}{\in}\mathbb{B}^{m(n)-|x|}$
2. Если «различитель» на входе $x\times y$ выдал 1, то возвращаем $y^{[1]}$, иначе $\neg y^{[1]}$.

Обозначим $\sigma_i(n)=\mu\Big(\Big\{D\Big(1^n;\xi_n^{[1,\dots,i]}\times v_{m(n)}^{[i+1,\dots,m(n)]}\Big)=1\Big\}\Big); 0\le i\le m(n)$ Тогда рассмотрим цепочку равенств:

$$\mu\Big(\Big\{P\Big(1^n;\xi_n^{[1,\dots,i-1]}\Big)=\xi_n^{[i]}\Big\}\Big)=$$

$$\mu\Big(\Big\{P\Big(1^n;\xi_n^{[1,\dots,i-1]}\Big)=\xi_n^{[i]},v_{m(n)}^{[i]}=\xi_n^{[i]}\Big\}\Big)+\mu\Big(\Big\{P\Big(1^n;\xi_n^{[1,\dots,i-1]}\Big)=\xi_n^{[i]},v_{m(n)}^{[i]}=\neg\xi_n^{[i]}\Big\}\Big)=$$

$$\mu\Big(\Big\{P\Big(1^n;\xi_n^{[1,\dots,i-1]}\Big)=v_{m(n)}^{[i]},v_{m(n)}^{[i]}=\xi_n^{[i]}\Big\}\Big)+\mu\Big(\Big\{P\Big(1^n;\xi_n^{[1,\dots,i-1]}\Big)=\neg v_{m(n)}^{[i]},v_{m(n)}^{[i]}=\neg\xi_n^{[i]}\Big\}\Big)=$$

$$\mu\Big(\Big\{D\Big(1^n;\xi_n^{[1,\dots,i-1]}\times v_{m(n)}^{[i,\dots,m(n)]}\Big)=1,v_{m(n)}^{[i]}=\xi_n^{[i]}\Big\}\Big)+\mu\Big(\Big\{D\Big(1^n;\xi_n^{[1,\dots,i-1]}\times v_{m(n)}^{[i,\dots,m(n)]}\Big)=0,v_{m(n)}^{[i]}=\neg\xi_n^{[i]}\Big\}\Big)=$$

$$\sum\nolimits_{b\in\mathbb{B}}\mu\Big(\Big\{D\Big(1^n;\xi_n^{[1,\dots,i-1]}\times b\times v_{m(n)}^{[i+1,\dots,m(n)]}\Big)=1,v_{m(n)}^{[i]}=b,b=\xi_n^{[i]}\Big\}\Big)+$$

$$\sum\nolimits_{b\in\mathbb{B}}\mu\Big(\Big\{D\Big(1^n;\xi_n^{[1,\dots,i-1]}\times b\times v_{m(n)}^{[i+1,\dots,m(n)]}\Big)=0,v_{m(n)}^{[i]}=b,b=\neg\xi_n^{[i]}\Big\}\Big)=$$

$$\tfrac{1}{2}\mu\Big(\Big\{D\Big(1^n;\xi_n^{[1,\dots,i]}\times v_{m(n)}^{[i+1,\dots,m(n)]}\Big)=1\Big\}\Big)+\tfrac{1}{2}\mu\Big(\Big\{D\Big(1^n;\xi_n^{[1,\dots,i-1]}\times\neg\xi_n^{[i]}\times v_{m(n)}^{[i+1,\dots,m(n)]}\Big)=0\Big\}\Big)=$$

$$\tfrac{1}{2}\sigma_i(n)+\tfrac{1}{2}(1-2\sigma_{i-1}(n)+\sigma_i(n))=\tfrac{1}{2}+\sigma_i(n)-\sigma_{i-1}(n)$$

Где в последнем переходе используется равенство:

$$\mu\Big(\Big\{D\Big(1^n,\xi_n^{[1,\dots,i-1]}\times\neg\xi_n^{[i]}\times v_{m(n)}^{[i+1,\dots,m(n)]}\Big)=1\Big\}\Big)=2\sigma_{i-1}(n)-\sigma_i(n)$$

которое получается аналогичным расписываниям выше, но для предсказания $i-1$ бита.

В итоге

$$\mu\Big(\Big\{P\Big(1^n;\xi_n^{[1,\dots,i-1]}\Big)=\xi_n^{[i]}\Big\}\Big)=\tfrac{1}{m(n)}\sum\nolimits_{k=1}^{m(n)}\mu\Big(\Big\{P\Big(1^n;\xi_n^{[1,\dots,k-1]}\Big)=\xi_n^{[k]}\Big\}\Big)=$$

$$\tfrac{1}{2}+\tfrac{1}{m(n)}\sum\nolimits_{k=1}^{m(n)}(\sigma_k(n)-\sigma_{k-1}(n))=$$

$$\tfrac{1}{2}+\tfrac{1}{m(n)}\Big(\mu(\{D(1^n;\xi_n)=1\})-\mu\Big(\Big\{D\Big(1^n;v_{m(n)}\Big)=1\Big\}\Big)\Big)>\tfrac{1}{2}+\tfrac{1}{m(n)p(n)}$$

Что для бесконечно многих n даёт противоречие с условием непредсказуемости следующего бита. □

8. О псевдослучайных генераторах

Определение 8.1: Функция, являющаяся одновременно односторонней и биекцией называется **односторонней перестановкой**.

Утверждение 8.1 (Яо): Если существует односторонняя перестановка, то существует псевдослучайный генератор.

Доказательство: Пусть f – односторонняя перестановка.

Продолжим её на всё $\mathbb{B}^*$ (обрубаем до префикса, на котором была определена) и построим $f'(x,r) = (f(x), r)$, как в теореме Гольдрайха-Левина.

Получили, что f' также односторонняя перестановка с трудным предикатом $b(\cdot)$.

Определим $g : x \mapsto f'(x)b(x)$, который и будет псевдослучайным генератором. □

Замечание 8.1: То, что f перестановка, нужно не только для обеспечения правильных длин значений, но и для того, чтобы $f(x)$ было равномерно распределено на $\mathbb{B}^n$ при $x \underset{U}{\in} \mathbb{B}^n$.

Теорема 8.1 (Хостада и других, без доказательства): псевдослучайные генераторы существуют тогда и только тогда, когда существуют односторонние функции.

9. О криптосистемах

Замечание 9.1: Будем использовать обозначения:
- $n \in \mathbb{N}$ – **параметр стойкости**
- $M_n \subseteq \mathbb{B}^*$ – пространство сообщений (открытых текстов)
- $\mathrm{supp}(\xi) = \{x \mid \mu(\{\xi = x\}) \neq 0\}$ – **носитель** случайной величины

Определение 9.1: **Система (вероятностного) шифрования с секретным ключом (криптосистема, шифр)** – это тройка алгоритмов (G, E, D):
- Генератор ключей G – п.в.м.Т., $G(1^n) = k$ – **секретный ключ**, можно считать, что k выбирается из $K_n = \mathrm{supp}(G(1^n))$ согласно вероятностному распределению $\mathcal{G}_n$, задаваемому случайной величиной $G(1^n)$.
- Алгоритм шифрования E – п. (в.) м.Т., для $m \in M_n, k \in K_n : E(1^n; k, m) = c$ – **криптограмма (шифртекст)** открытого текста m на ключе k.
- Алгоритм дешифрования D – п.д.м.Т.

$$\forall m \in M_n : \forall k \in K_n : D(1^n; k, E(1^n; k, m)) = m$$

Определение 9.2: Система шифрования называется **блоковой**, если в ней алгоритм шифрования разбивает сообщение произвольной длины на блоки и шифрует каждый блок отдельно по **одному и тому же** алгоритму.

Определение 9.3: **Потоковая** же криптосистема последовательно шифрует элементы открытого текста, такими элементами чаще всего являются биты, данный тип криптосистем имеет внутренне состояние, **изменяющееся** после шифрования каждого нового сообщения.

10. О стойкости криптосистем

Определение 10.1: **Стойкость** криптосистемы определяется относительно конкретного противника.

Замечание 10.1 (Модель противника):
- Вычислительные ресурсы = п.в.м.Т.
- Атака – возможность получения исходных данных
- Угроза – цель противника

Замечание 10.2 (Основные типы атак):
1. **Атака с известными шифртекстами**:
$$c_1, c_2, ..., c_l$$
2. **Атака с известными открытыми текстами**:
$(m_1, c_1), (m_2, c_2), ..., (m_l, c_l); \quad c_i = E(1^n; k, m_i)$
3. **Атака с выбором открытых текстов**:
$m_1, ..., m_l \mapsto (m_1, c_1), ..., (m_l, c_l); \quad c_i = E(1^n; k, m_i)$
4. **Атака с выбором шифртекстов**:
$c_1, ..., c_l \mapsto (m_1, c_1), ..., (m_l, c_l); \quad m_i = D(1^n; k, c_i)$
5. **Атака с выбором текстов** – комбинация 3 и 4.

Атаки 3-5 бывают:
- **Неадаптивными**, когда противник получает весь набор данных разом
- **Адаптивными**, когда пары к выбранным данным он получает последовательно по i, то есть выбор следующего запроса может зависеть от результатов предыдущего.

Замечание 10.3 (Основные типы угроз):
1. **Полное раскрытие** – найти использованный ключ
2. **Извлечение открытого текста** – по известной информации и случайному значению $E(1^n; k, m)$ найти сообщение m
3. **Извлечение частичной информации об открытом тексте**: для некоторой функции $f : \mathbb{B}^* \to \mathbb{B}^*$ по известной информации и случайному значению $E(1^n; k, m)$ найти $f(m)$
4. **Различие двух шифртекстов** – при подходящей выборке m^0, m^1 открытых сообщений, не появлявшихся при атаке, по криптограмме $E\big(1^n; k, m^b\big)$ для случайного $b \underset{\mathcal{U}}{\in} \{0, 1\}$ определить b, то есть какое из двух сообщений было зашифровано

11. Конкретный пример стойкости

Замечание 11.1: Определим IND-CPA-стойкость криптосистемы с секретным ключом – стойкость относительно пары угрозы 4/атака 3.

Формализуем предположения о противнике с помощью специального оракула, к которому имеет доступ алгоритм противника.

Оракул $\mathcal{O}$, определяемый для криптосистемы (G, E, D):
- В начале работы выбирает секретный ключ $k \underset{\mathcal{G}_n}{\in} K_n$
- После этого принимает запросы двух типов:
  1. $(1; x)$, где $x \in M_n$ в ответ на который возвращает $E(1^n; k, x)$
  2. $(2; y^0, y^1)$, где $y^0, y^1 \in M_n$, получив который, проверяет, что y^0 и y^1 не появлялись ранее, выбирает случайный бит $b \underset{\mathcal{U}}{\in} \{0, 1\}$ и в зависимости от значения b возвращает либо $E(1^n, k, y^0)$, либо $E(1^n, k, y^1)$.
- Ответив на один запрос второго типа, завершает свою работу

Определение 11.1: Криптосистема (G, E, D) называется **IND-CPA-стойкой**, если для любой п.в.м.Т. A с вышеописанным оракулом $\mathcal{O}$:

$$\mu\big(\{A^{\mathcal{O}}(1^n) = b\}\big) \leq \tfrac{1}{2} + \mathrm{negl}(n)$$

Пример: Пусть g – псевдослучайный генератор, $g(\mathbb{B}^n) \subseteq \mathbb{B}^{q(n)}$

$m_1, ..., m_t \in \mathbb{B}^n$ – сообщения, причём $t \cdot n + 1 < q(n)$

Участники обмениваются по защищённому каналу секретным ключом $k \underset{\mathcal{U}}{\in} \mathbb{B}^n$, причём $g(k) = g_1 ... g_t, g_i \in \mathbb{B}^n$.

Тогда для $1 \leq i \leq t$:
- Шифрование – $c_i = E(1^n; k, m_i) = m_i \oplus g_i$
- Дешифрование – $m_i = D(1^n; k, c_i) = c_i \oplus g_i$

Утверждение 11.1: Если g – псевдослучайный генератор, то описанная выше криптосистема – IND-CPA-стойкая.

Доказательство: Предположим, что существует такая п.в.м.Т. A, что для некоторого полинома p и бесконечно многих n $\mu(\{A^{\mathcal{O}}(1^n) = b\}) > \frac{1}{2} + \frac{1}{p(n)}$.

Построим п.в.м.Т. S, работающую на входе $(1^n; z), z \in \mathbb{B}^{q(n)}, z = z_1 ... z_t, z_i \in \mathbb{B}^n$ следующим образом:

1. S запускает машину A на входе 1^n и берёт на себя роль оракула, делая xor с элементами z.
2. Вычисляет выход – $S(1^n; z) = \begin{cases} 1 & A^S(1^n)=b \\ 0 & A^S(1^n)\neq b \end{cases}$

Таким образом,

- если $z = g(v_n)$, то по предположению A вычисляет b с вероятностью $> \frac{1}{2} + \frac{1}{p(n)}$
- иначе $z = v_{q(n)}$ – произвольная равномерная случайная величина, отгадывающая ответ с вероятностью подбрасывания монетки

Получили, что $\mu(\{S(1^n, g(v_n)) = 1\}) - \mu\Big(\Big\{S\Big(1^n, v_{q(n)}\Big) = 1\Big\}\Big) > \frac{1}{p(n)}$, что противоречит с тем, что g – псевдослучайный генератор. □

12. О генераторах псевдослучайных функций

Замечание 12.1: Будем рассматривать семейства функций вида

$$F = \cup_{n\in\mathbb{N}} F_n = \left\{f_{n,i} : \mathbb{B}^{l(n)} \to \mathbb{B}^{m(n)}\right\}_{n\in\mathbb{N}, i\in\mathbb{B}^n} \quad F_n \subseteq \left(\mathbb{B}^{m(n)}\right)^{\mathbb{B}^{l(n)}}$$

где $l(\cdot), m(\cdot)$ – некоторые полиномы.

Определение 12.1: F называется **псевдослучайным семейством функций**, если

- F полиномиально вычислимо, в смысле

$$\exists \text{ п.д.м.Т } A : \forall n, i, w : A(1^n; i, w) = f_{n,i}(w)$$

- Для любой п.в.м.Т A:

$$\left|\mu_i\left(\left\{A^{f_{n,i}}(1^n) = 1\right\}\right) - \mu_\varphi(\{A^\varphi(1^n) = 1\})\right| = \text{negl}(n)$$

где $i \underset{U}{\in} \mathbb{B}^n, \varphi \underset{U}{\in} \left(\mathbb{B}^{m(n)}\right)^{\mathbb{B}^{l(n)}}$

Определение 12.2: **Генератор для семейства функций F** – это пара алгоритмов (I, C):

- I – полиномиальная вероятностная машина Тьюринга:

$$I(1^n) = i \text{ – индекс функции а } F_n$$

- C – полиномиальная (детерминированная) машина Тьюринга:

$$\forall n, i, x : C(1^n; i, x) = f_{n,i}(x)$$

Определение 12.3: **Генератором всевдослучайных функций** будем называть генератор для псевдослучайного семейства функций F относительно некоторого семейства вероятностных распределений индексов $\{\mathcal{I}_n\}_{n\in\mathbb{N}}$.

Теорема 12.1 (Гольдрайха и других): Если существует псевдослучайный генератор, то для любых полиномов $l(\cdot), m(\cdot)$ существует псевдослучайное семейство функций F.

Доказательство: Возьмём псевдослучайный генератор $g : \mathbb{B}^n \to \mathbb{B}^{2n}$ для всех n и функции

$$g_0(y) = g(y)^{[1,\dots,n]}; \quad g_1(y) = g(y)^{[n+1,\dots,2n]}; \quad y \in \mathbb{B}^n, n \in \mathbb{N}$$

Определим функции семейства для $x \in \mathbb{B}^{l(n)}$ по всем n:

$$f'_{n,i}(x) = g_{x^{[l(n)]}}(\dots g_{x^{[1]}}(i)\dots) \in \mathbb{B}^n$$

Псевдослучайность семейства $F' = \left\{f'_{n,i}\right\}$ доказывается от противного:

Если п.в.м.Т A отличает функции семейства от случайных, построим п.в.м.Т B, которая запускает A, выдаёт ей (вместо оракула) значение хитро строящейся функции h и возвращает в конце выход машины A.

Тогда B будет отличать случайный вектор $g(y) \in \mathbb{B}^{2n}$ от равномерно случайного вектора v_{2n}, что противоречит определению псевдослучайного генератора g.

Далее «растянем» до длины $m(n)$ значения функций семейства F' с помощью их композиций с подходящим псевдослучайным генератором.

Полученное семейство F будет таким же псевдослучайным, как и F'. □

13. О генераторах псевдослучайных перестановок

Определение 13.1: **Псевдослучайным семейством перестановок** называется псевдослучайное семейство функций, все функции которого являются биекциями.

Также от них требуется неотличимость от раномерно случайной перестановки, а не от произвольной функции.

Утверждение 13.1: Существует **преобразование Файстеля** Φ, которое превращает произвольную функцию, сохраняющую длину, в перестановку

Доказательство: Давайте определим преобразование $\forall n \in \mathbb{N}$ в явном виде, пусть $f : \mathbb{B}^n \to \mathbb{B}^n$.

Тогда её преобразованием Файстеля, а также обратное к нему:

$$\forall x, y, u, v \in \mathbb{B}^n : \begin{cases} \Phi_f(x \times y) = y \times (x \oplus f(y)) \\ \Phi_f^{-1}(uv) = (v \oplus f(u)) \times u \end{cases}$$

□

Определение 13.2: F называется **полиномиально инвертируемым семейством перестановок**, если полиномиально вычислима функция $(1^n; i, y) \mapsto f_{n,i}^{-1}(y)$.

Теорема 13.1 (Луби-Ракоффа, без доказательства): Если существует псевдослучайное семейство функций, сохраняющих длину, то существует полиномиально инвертируемое псевдослучайное семейство перестановок.

14. Использование псевдослучайных семейств для шифрования

Замечание 14.1: Пусть $F = \cup_{n \in \mathbb{N}} \left\{ f_{n,i} : B^{l(n)} \to B^{l(n)} \right\}_{n \in \mathbb{N}, i \in \mathbb{B}^n}$ – полиномиально инвертируемое псевдослучайное семейство перестановок.

n – параметр стойкости.

m – открытый текст, при необходимости дополненный до длины $l(n)$

$M_n = \mathbb{B}^{l(n)}$

Замечание 14.2: Построим криптосистему, используя данное семейство перестановок

- $G(1^n) = i \underset{U}{\in} \mathbb{B}^n$ – секретный ключ
- $E(1^n; i, m) = f_{n,i}(m) = c \in \mathbb{B}^{l(n)}$ – криптограмма
- $D(1^n; i, c) = f_{n,i}^{-1}(c) = m$

D – полиномиальная, так как семейство F полиномиально инвертируемо

Такая (блоковая) криптосистема с секретным ключом – IND-CPA-стойкая.

Замечание 14.3: Если позволить шифровать более длинные сообщения, разбивая их на блоки длины $l(n)$ и применяя к ним перестановку по отдельности, то такая система уже не будет IND-стойкой.

15. О электронной подписи.

15.1. Определения

Определение 15.1.1: Схемой электронной подписи называется следующую тройку алгоритмов (G, S, V) вместе с процедурой A:

1. Генератор ключей G — п.в.м.Т : $G(1^n) = \left(\hat{k}, k\right)$ - пара секретный/открытый ключ. $K_n = \text{supp}(G(1^n))$ — пространство ключей.
2. Генератор подписей S — п.в.м.Т : $m \in M_n, \left(\hat{k}, k\right) \in K_n \Rightarrow S\left(1^n, \hat{k}, m\right) = s$ — подпись для сообщения m.
3. Алгоритм проверки п.д.м.Т $V : V(1^n, k, m, s) \in \mathbb{B}$ - принимается ли подпись. Причём $V\left(1^n, k, m, S\left(1^n, \hat{k}, m\right)\right) = 1$. То есть правильная подпись всегда принимается.
4. A - процедура арбитража (разрешения споров). Арбитр - кто-то кому все участники доверяют, и он может получить доступ к секретам.

Определение 15.1.2: Схема аутентификации сообщений (**MAC**).
Строится аналогично, только без открытого ключа

- $\hat{k}$ известен обоим участникам
- Оба могут как подписывать так и проверять подпись.
- Нет задачи убедить в чём-то третью сторону. MAC предназначен для сети из небольшого числа доверяющих друг другу участников

Замечание 15.1.1: Считаем, что противнику по умолчанию известна схема, то есть (G, S, V), параметр n, а так же открытый ключ.

Замечание 15.1.2: Основные типы атак

1. Атака с известным открытым ключом - **KKA**.
   - Знаем только открытый ключ
2. Атака с известными сообщениями - **KMA**
   - Известен набор сообщений с подписями $\{(m_i, s_i)\}_{i=0..l}$
3. Атака с выбором сообщений - **CMA**.
   - Противник может подписывать некоторые сообщения. $\{m_i\} \rightsquigarrow \{(m_i, s_i)\}$
   - Может быть с априорным знанием k (направленная) или нет (простая).
   - Может быть адаптивной или неадаптивной. (Зависит ли m_i, от ответов на предыдущие запросы).

Замечание 15.1.3: Основные типы угроз
1. Полное раскрытие (total breaking).
2. Универсальная подделка (universal forgery)
   - Найти п.в.м.Т $S' : \forall m \in M_n V(1^n, k, m, S'(1^n, k, m)) = 1$. То есть научиться подписывать без приватного ключа $\hat{k}$.
3. Селективная подделка Selective forgery
   - Найти подпись для какого то конкретного сообщения m. (Здесь и далее всегда неявно подразумевается, что подпись сообщения которое хочет подписать атакующий ему не была известна заранее / сказана оракулом).
4. Экзистенциальная подделка.
   - Найти пару $(m', s') : V(1^n, k, m', s') = 1$.

Определение 15.1.3: Если никакой эффективный алгоритм не может осуществить угрозу экзистенциальной подделки с существенной вероятностью, то схема электронной подписи называется EU-стойкой.

Определение 15.1.4: EU-CMA стойкость
Оракул-подписант: $\mathcal{O} : \mathcal{O}(m) = S\left(1^n, \hat{k}, m\right)$.
Схема (G, S, V) EU-CMA стойкая если
$\forall$ п.в.м.Т $A^{\mathcal{O}}$: $\mu(\{A^{\mathcal{O}}(1^n, k) = (m, s),\ V(1^n, k, m, s) = 1\}) = \text{negl}(n)$

Теорема 15.1.1: **Rompel** Если существует односторонняя функция, то существует EU-CMA стойкая схема электронной подписи.

Замечание 15.1.4: В обратную сторону тоже верно. Предположим противное, тогда можно эффективно обратить генератор ключей, и тем самым по открытому ключу получить секретный. Другими словами $f(r) = k \Leftrightarrow G(r, 1^n) = \left(k, \hat{k}\right)$ должна быть односторонней.

15.2. Примеры схем

15.2.1. RSA

$N = pq,\ p, q \in \mathbb{P}$
$\gcd(e, \varphi(N)) = 1, ed \equiv 1 \bmod \varphi(N)$, где φ – функция Эйлера.

- $G : k = (N, e),\ \hat{k} = (N, d)$
- $S : s = m^d \bmod N$
- $V : s^e \stackrel{?}{=} m \bmod N$. $m^{de} \equiv m^{1+l\varphi(N)} \equiv m \bmod N$

Замечание 15.2.1.1: RSA НЕ является ни EU-KKA стойкой ни UU-CMA стойкой.

15.2.2. Схема Лемпорта (одноразовая)

Пусть $f : \mathbb{B}^n \to \mathbb{B}^n$ односторонняя функция сохраняющая длину. Сообщение $m \in M_n = \mathbb{B}^n$. $m = m^{[1]}m^{[2]}...m^{[n]}$

- G. Нагенерируем $2n$ случайных последовательностей: $\{x_i^0, x_i^1\}_{i=0}^n$. Они будут нашим секретным ключом. Ко всем ним применяем f. Получаем $\{y_i^0, y_i^1\}_{i=0}^n$ это публичный ключ.
- S. В зависимости от значения i-го бита в числе выбираем либо x_i^0 либо x_i^1. Формально $s = s^{[1]}s^{[2]}...s^{[n]} = x_0^{m^{[0]}}x_1^{m^{[1]}}...x_n^{m^{[n]}}$
- V. Применяем f к подписи и сверяемся что все сходится: $y_i^{m^{[i]}} = f(s_i)$.

Замечание 15.2.2.1: Эта схема EU-CMA1-стойкая — при условии, что противнику доступно только одно обращение к оракулу ($\mathrm{l} = 1$).

Замечание 15.2.2.2: Чтобы снять ограничение на длину сообщения, достаточно подписывать хеш от сообщения. Далее это будет считаться стратегией по умолчанию.

16. О криптографических хеш-функций

Определение 16.1: **Семейством хэш-функций** называется

$$H = \cup_{n\in\mathbb{N}} H_n = \left\{h_{n,i} : \mathbb{B}^{t(n)} \to \mathbb{B}^{m(n)}\right\}_{n\in\mathbb{N}, i\in I_n}$$

где $t(\cdot), m(\cdot)$ – такие полиномы, что $m(n) < t(n)$ для всех $n \in \mathbb{N}$, а на каждом случайном векторе I_n задано распределение вероятностей $\mathcal{J}_n$, так, что $\{\mathcal{J}_n\}_{n\in\mathbb{N}}$ – полиномиально конструируемое семейство.

Определение 16.2: H называется **семейством хэш-функций с трудно-обнаружимыми коллизиями** относительно $\{\mathcal{J}_n\}_{n\in\mathbb{N}}$, если

- H – полиномиально вычислимо
- Для любой п.в.м.Т A

$$\mu_{i\leftarrow\mathcal{J}_n}\left(\left\{A(1^n; i) = (x', x''), x' \neq x'', h_{n,i}(x') = h_{n,i}(x'')\right\}\right) = \mathrm{negl}(n)$$

Определение 16.3: H называется **(универсальным) односторонним семейством хэш-функций** относительно $\{\mathcal{I}_n\}_{n\in\mathbb{N}}$, если

- H полиномиально вычислимо
- Для любой такой п.в.м.Т A, которая сначала на входе 1^n выдаёт $x' \in \mathbb{B}^{t(n)}$, а затем на входе $i \in I_n$ выдаёт $x'' \in \mathbb{B}^{t(n)}$ (A реализует алгоритм поиска специфичных колизий)

$$\mu_{i\leftarrow\mathcal{I}_n}\left(\left\{A(1^n;i)=x'', x''\neq x', h_{n,i}(x')=h_{n,i}(x'')\right\}\right)=\mathrm{negl}(n)$$

Теорема 16.1 (Наора-Юнга): Если существует односторонняя перестановка, то существует и одностороннее семейство хэш-функций.

Доказательство: Отождествим $\mathbb{B}^n$ с $\mathbb{GF}(2^n)$ для каждого n. (все рассматриваемые алгебраические операции над векторами будут иметься в виду именно из этого поля).

Пусть

$$G_n := \left\{\forall a,b\in\mathbb{B}^n : a\neq 0 \mid g_{n,a,b}:\mathbb{B}^n\to\mathbb{B}^{n-1} : g_{n,a,b}(x)=(ax+b)^{[1,\dots,n-1]}\right\}$$

Заметим, что строки x', x'' образуют коллизию для $g_{n,a,b}$ тогда и только тогда, когда $g_{n,a,b}(x'), g_{n,a,b}(x'')$ различаются только в последнем бите.

$$ax'+b = ax''+b+1 \Rightarrow a=(x'-x'')^{-1}$$

Пусть $M(x',x'')=\left((x'-x'')^{-1}, r\right)$, где $x', x''\in\mathbb{B}^n, r\underset{\mathcal{U}}{\in}\mathbb{B}^n$.

По сути M по двум элементам выдаёт индекс хэш-функции, на которой на них будет коллизия.

Тогда M удовлетворяет свойству достижимости коллизий.

Кроме того, поскольку $(x'-v)^{-1}\underset{\mathcal{U}}{\in}\mathbb{B}^n\setminus\{0\}$ при $v\underset{\mathcal{U}}{\in}\mathbb{B}^n\setminus\{x'\}$, случайная величина $M(x',v)$ распределена равномерно на множестве описаний всех функций из G_n.

Пусть предполагаемая существуюшая односторонняя перестановка – f. Рассмотрим семейство

$$H=\cup_{n\in\mathbb{N}}H_n=\cup_{n\in\mathbb{N}}\left\{h_{n,a,b}=g_{n,a,b}\circ f \mid g_{n,a,b}\in G_n\right\}$$

Очевидно, оно полиномиально вычислимо, как состоящее из композиций полиномиально вычислимых функций.

Докажем, что задача поиска специфических коллизий для него трудна.

Пусть произвольная п.в.м.Т A, реализующая алгоритм поиска специфических коллизий для семейства хэш-функций H.

Напомним, что п.в.м.Т A работает в два этапа:

- На первом по случайному вектору 1^n выдаёт x, к которому на втором этапе будем искать коллизию
- На втором этапе $A(1^n;v)$ выдаёт x', который должен выдавать коллизию с x на хэш-функции $h_{n,v}$. В этом конкретном случае $v=(a,b)$.

На основе A построим вспомогательную п.в.м.Т B, которая будет дальше применена в доказательстве.

Алгоритм B будет принимать на вход $v \in \mathbb{B}^n$. (Хотим построить B так, чтобы $B(1^n; f(u)) = u$).

- Если $f(x) = v$, где x из описания первого этапа A, то B вернёт x
- Иначе B вернёт результат выполнения $y = A(1^n; M(f(x), v))$, то есть элемент, который имеет коллизию с x на такой хэш-функции, в которой имеют коллизию элементы $f(x)$ и v.

Очевидно, B полиномиальна.

Предположим, что для поданной на вход машине B строки $f(u), u \in \mathbb{B}^n$ выполнено

- $f(x) \neq v = f(u)$
- $y \neq x$
- $h_{n,a,b}(y) = h_{n,a,b}(x)$, где $(a, b) = M(f(x), f(u))$

Из этого случая можно сделать следующие выводы:

- $f(x) \neq f(y)$, така как f перестановка и $x \neq y$.
- $g_{n,a,b}(f(y)) = h_{n,a,b}(y) = h_{n,a,b}(x) = g_{n,a,b}(f(x))$ по предположению
- Но при этом $g_{n,a,b}(f(x)) = g_{n,a,b}(f(u))$, так как по определению M для функции с индексов $(a, b) = M(f(x), v)$ пара $(f(x), v) = (f(x), f(u))$ образует коллизию.

Итого $f(y) \neq f(x) \neq f(u)$, но при этом $g_{n,a,b}(f(y)) = g_{n,a,b}(f(x)) = g_{n,a,b}(f(u))$.

Следовательно, $f(y) = f(u)$, так как у $g_{n,a,b}(f(x))$ не может быть три различных прообраза.

Следовательно $y = u$, так как f – перестановка $\Rightarrow$ на входе $f(u)$ машина B выдаёт u.

Теперь используя эту п.в.м.Т B с доказанным свойством докажем сложность произвольного алгоритма поиска специфических коллизий для семейства хэш-функций H.

Пусть x – выход A после 1 этапа, $(a, b) \underset{\mathcal{U}}{\in} (\mathbb{B}^n \setminus \{0^n\}) \times \mathbb{B}^n /$

Заметим, что $u \underset{\mathcal{U}}{\in} \mathbb{B}^n \setminus \{x\} \Leftrightarrow f(u) \underset{\mathcal{U}}{\in} \mathbb{B}^n \setminus \{f(x)\}$.

Наконец, мы готовы совершить оценку сложности поиска коллизий

$$\mu_{(a,b)}\big(\{A(1^n; (a, b)) = y, y \neq x, h_{n,a,b}(y) = h_{n,a,b}(x)\}\big) =$$

$$\mu_u\big(\{A(1^n; M(f(x), f(y))) = y, y \neq x, h_{n,a,b}(x) = h_{n,a,b}(y)\}\big) =$$

$$\mu_{v_n}(\{B(1^n; f(v_n)) = v_n \mid f(x) \neq f(v_n)\}) \leq \frac{\mu_{v_n}(\{B(1^n; f(v_n)) = v_n\})}{\mu_{v_n}(\{f(x) \neq f(v_n)\})} \leq 2\,\mathrm{negl}(n)$$

В последнем переходе мы воспользовались тем, что $\mu_{v_n}(\{f(x) \neq f(v_n)\}) = 1 - \frac{1}{2^n} \geq \frac{1}{2}$ и тем, что f – односторонняя перестановка. $\square$

17. О применении хэш-функций

Замечание 17.1: Модифицируем схему Лемпорта, описанную в Раздел 15.2.2.

Чтобы снять ограничение на длину $|m| = n$, воспользуемся семейством криптографических хэш-функций

$$H = \cup_{n\in\mathbb{N}} H_n; \quad H_n = \left\{h_{n,d} : \mathbb{B}^{l(n)} \to \mathbb{B}^n\right\}_{d\in I_n}$$

К обоим ключам добавляется $d \leftarrow \mathcal{J}_n$, а вместо $m \in \mathbb{B}^{l(n)}$ подписывается $h_{n,d}(m) \in \mathbb{B}^n$. При проверке также сообщения сначала хэшируется.

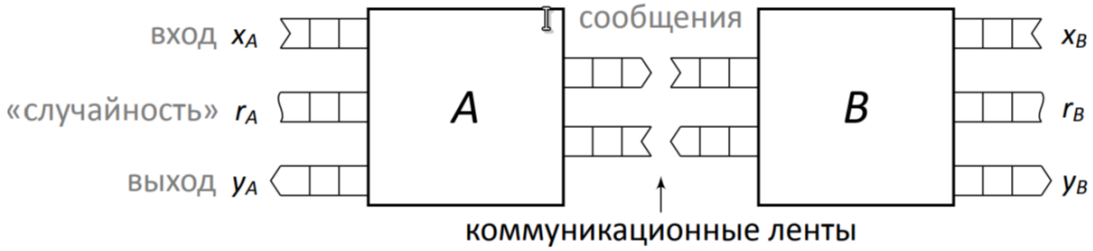
Теорема 17.1 (Ромпеля, без доказательства): Односторонние семейства хэш-функций существуют тогда и только тогда, когда существует односторонняя функция.

18. О нулевых разглашениях

Замечание 18.1: Предположим, что Алиса знает доказательство некоторой теоремы и желает убедить Боба, что теорема верна, так, чтобы не сообщить ему никакой информации, на основании которой он научился бы доказывать эту теорему самостоятельно.

Этим двум противоречивым требованиям удовлетворяют **протоколы с нулевым разглашением**.

Определение 18.1: **Интерактивной парой машин Тьюринга** называется пара п.в.м.Т (A, B), соединённые между собой коммуникационной лентой:



Замечание 18.2 (Особенности интерактивной пары машин Тьюринга):
- Вход может быть общим у обеих машин, но они могут иметь и свои частные входы.
- **Раунд** – период активности однйо машины с записью слова на коммуникационную ленту.
- Если раунд только один, то соответствующий протокол называется **неинтерактивным**.
- Выход пары (A, B) – это (y_A, y_B) или только y_B, если машина B останавливается первой.
- Интерактивная машина **полиномиальна**, если время её работы при совместном вычислении с любой другой машиной Тьюринга на общем входе x ограничено величиной $\text{poly}(|x|)$. В этом случае число раундов также полиномиально, но размер входящих сообщений – вообще говоря, необязательно.

Замечание 18.3: Введём обозначения
- P – **доказывающий**
- V – **проверяющий**, выдающий 1 или 0 (доказательство принимается или нет)
- $L \subseteq \mathbb{B}^*$ – язык, соответствующий некоторой распознавательной задаче.

Определение 18.2: **Интерактивной доказательство** для языка L – это интерактивная пара машин Тьюринга (P, V) с полиномиальной V, для которых выполнены условия:
- Полнота
$$\forall x \in L : \mu(\{(P, V)(x) = 1\}) \geq \tfrac{2}{3}$$
- Корректность
$$\forall \text{ интерактивной в.м.Т. } P' : \forall x \notin L : \mu(\{(P', V)(x) = 1\}) \leq \tfrac{1}{3}$$

Определение 18.3: Рассмотрим выполнение интерактивной пары машин Тьюринга (P, V) и обозначим через $\text{view}_V^P(x)$ **транскрипцию этого выполнения**, состоящую из использованного префикса случайной строки машины V, последовательности пересылаемых сообщений в хронологическом порядке и выхода пары.

Определение 18.4: Интерактивной доказательство (P, V) для бесконечного языка L называется **доказательством с абсолютно нулевым разглашением**, если для любой интерактивной п.в.м.Т V' существует п.в.м.Т S, такая, что семейства $\left\{\text{view}_{V'}^{P}(x)\right\}_{x\in L}$ и $\{S(x)\}_{x\in L}$ распределены одинаково, а точкее, для любого $x \in L$:

- $\mu(\{S(x) \neq \perp\}) \geq \frac{1}{\text{poly}(|x|)}$
- При условии $S(x) \neq \perp$: $\mu(\{S(x) = z\}) = \mu\left(\left\{\text{view}_{V'}^{P}(x) = z\right\}\right)$

Это определение значит, что наличие честного «доказывателя» никак не поможет нечестному «проверяющему» доказывать.

Определение 18.5: Пусть ξ, ζ – случайные величины со значениями в конечном или счётном множестве X.

Статистическим расстоянием между ξ и ζ называют

$$\Delta(\xi, \zeta) = \max_{Z\subseteq X} |\mu(\{\xi \in Z\}) - \mu(\{\zeta \in Z\})|$$

Определение 18.6: Семейства $\{\xi_x\}_{x\in L}, \{\zeta_x\}_{x\in L}$ называются **статистически неразличимыми**, если $\Delta(\xi_x, \zeta_x) = \text{negl}(|x|)$

Определение 18.7: Интерактивное доказательство (P, V) для бесконечного языка L называется **доказательством со статистически нулевым разглашением**, если для любой интерактивной п.в.м.Т V' существует п.в.м.Т S такая, что семейства $\left\{\text{view}_{V'}^{P}(x)\right\}_{x\in L}$ и $\{S(x)\}_{x\in L}$ статистически неразличимы

Определение 18.8: Семейства $\{\xi_x\}_{x\in L}, \{\zeta_x\}_{x\in L}$ называются **вычислительно неразличимыми**, если для любой п.в.м.Т D:

$$|\mu(\{D(x, \xi_x) = 1\}) - \mu(\{D(x, \zeta_x) = 1\})| = \text{negl}(|x|)$$

Определение 18.9: Интерактивное доказательство (P, V) для бесконечного языка L называется **доказательством с вычислительно нулевым разглашением**, если для любой интерактивной п.в.м.Т V' существует п.в.м.Т S такая, что семейства $\left\{\text{view}_{V'}^{P}(x)\right\}_{x\in L}$ и $\{S(x)\}_{x\in L}$ вычислительно неразличимы

Пример (Протокол доказательства с абсолютно нулевым разглашением для языка пар изоморфных графов): Общий вход: (G_0, G_1), где $G_0 = (U, E_0), G_1 = (U, E_1)$.

Пусть $|U| = n$, причём $|E_0| = |E_1|$.

- Prover выбирает случайную n-перестановку $\pi \underset{U}{\in} S_n$ и случайный бит $c \underset{U}{\in} \mathbb{B}$

- Prover создаёт новый граф $H = \pi(G_c)$, изоморфный изначальным, посылает его Verifier
- Verifier выбирает случайный бит $b \underset{U}{\in} \mathbb{B}$, посылает его Prover
- Prover создаёт новую перестановку $\psi = \begin{cases} \pi, b = c \\ \varphi \circ \varphi, b \neq c \end{cases}$, где φ – известная только ему перестановка, такая, что $G_c = \varphi(G_{\neg c})$. Посылает полученную перестановку ψ Verifier.
- Verifier проверяет $\varphi(G_b) \stackrel{?}{=} H$

Описанный выше алгоритм запускается в цикле $|E_1|$ раз и результатом проверки на изоморфизм будет $1 \Leftrightarrow$ во всех итерациях проверка Verifier была пройдена успешна.