Псевдослучайные генераторы

Ю. Лифшиц*

26 ноября 2005 г.

План лекции

1. Понятие псевдослучайного генератора
2. Односторонние функции и генераторы
3. Криптосистема на основе генератора

1 Понятие псевдослучайного генератора

1.1 Основные понятия

Чтобы понять, что такое псевдослучайный генератор, нам нужно вспомнить понятие вычислительной неразличимости двух распределений. Формально, два распределения D_1, D_2 называются вычислительно неразличимыми, если для любого полиномиального вероятностного алгоритма A

$$|Pr_{x \leftarrow D_1}[A(x) = 1] - Pr_{x \leftarrow D_2}[A(x) = 1]| = \nu(|x|)$$

Неформальная постановка задачи: пусть у нас есть множества X и Y. Мы хотим, чтобы по случайным образом выбиранному элементу x из X мы смогли с помощью некой специальной функции $G(x)$ породить якобы случайный элемент из Y. Мы будем рассматривать только случаи, когда мощность множества Y намного больше мощности множества X, иначе задача существенно упрощается. Например, рассмотрим случай, когда $X = \{0...1000\}$, $Y = \{0...100\}$. Тогда, выбрав случайный $x \in X$, мы сможем легко вычислить $y = x \text{ div } 10$, который, очевидно, будет принадлежать Y и будет случайным в силу случайности выбора x.

Итак, пусть $|Y| >> |X|$. Функция $G : X \to Y$ называется псевдослучайным генератором, если $G(\mathcal{U}_X)$ и $\mathcal{U}_Y$ вычислительно неразличимы ($\mathcal{U}_X$,$\mathcal{U}_Y$, равномерные распределения).

*Законспектировал Т.Брыксин.

1.2 Область применения

- Криптография

  Каждый генератор можно использовать как криптосистему с секретным ключом. Также, они являются идельной моделью для блочных шифров.

- Построение алгоритмов

  Имеется ряд алгоритмов, использующих случайные биты. Имея генератор псевдослучайных чисел, мы могли бы брать лишь небольшое число случайных битов, а остальные получать, применяя к ним генератор. Алгоритм будет работать так же хорошо, т.к. иначе это дало бы возможность отличить случайные числа от псевдослучайных, что противоречит нашему предположению о существовании генератора.

- Теория сложности

  Класс BPP (bounded probability polynomial) - задачи, которые можно решить вероятностным алгоритмом за полиномиальное время.
  DTIME(2^{n^ϵ}) - множество всех задач, которые можно решить детерминированным алгоритмом за время $O(2^{n^\epsilon})$ без ограничений в памяти.
  Таким образом, если использовать генератор псевдослучайных чисел, будет иметь место следующее включение: $BPP \subset DTIME(2^{n^\epsilon})$

1.3 О приложениях в криптографии

Пусть P - класс полиномиальных задач. NP - класс задач, решаемых недетерминированной машиной Тьюринга за полиномиальное время. Существует открытая проблема: $P = NP$ или $P \neq NP$? Все задачи взлома входят в класс NP, и если бы выполнялось равенство, то это бы означало, что каждая задача взлома имела бы свой полиномиальный алгоритм решения. Практически все, что мы изучали до этого, можно было бы легко взломать (нулевое разглашение, криптосистемы с открытым ключом, передача данных вслепую, электронные выборы). Пока мы не докажем, что $P \neq NP$, мы не сможем утверждать безопасность. Хотя, может так случиться, что $P \neq NP$, но при этом задачи взлома построенных систем не являются самыми сложными в своем классе и успешно взламываются.

Другое базовое предположение - существование односторонних функций. Стойкость многих решений доказана в том смысле, что мы строим их с помощью каких-то абстрактных односторонних функций. Такой подход и используется для построения псевдослучайных генераторов.

Перечислим изученные нами понятия и рассмотрим связи между ними (см. Рис.1).

- Coin flipping - подбрасывание монет
- Bit commitment - привязка к биту

- Symmetric-key encryption - шифрование с симметричным ключом
- OWP (one-way permutation) - односторонняя перестанока
- OWF (one-way function) - односторонняя функция
- PRG (pseudorandom generator) - псевдослучайный генератор
- PRF (pseudorandom function) - псевдослучайная функция
- PRP (pseudorandom permutation) - псевдослучайная перестановка
- MAC (message authentication code) - код аутентификации сообщения

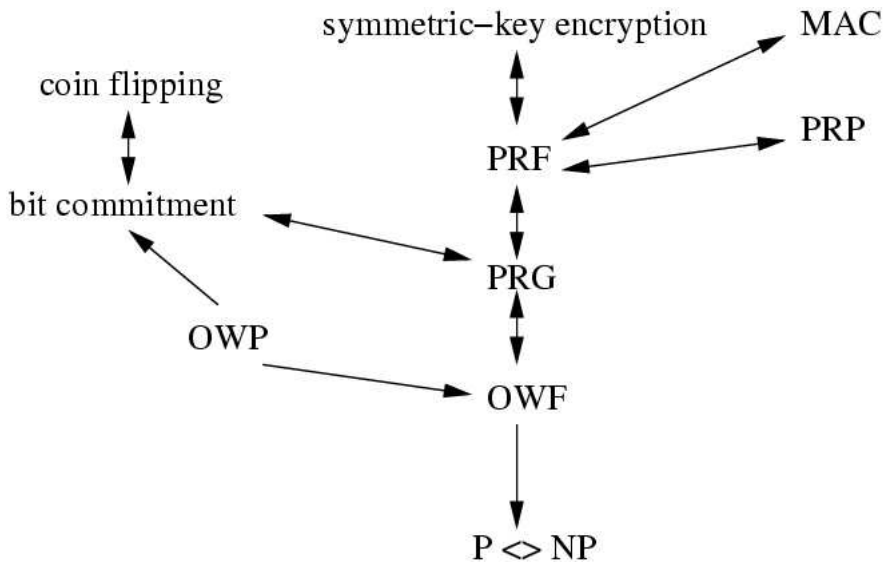

Рис.1. Связи между изученными понятиями

Если выполнено условие $P = PN$, то всего перечисленного не существует. Если кто-то докажет, что односторонних функций не существует, то это будет конец криптографии. Односторонние функции - очень важное понятие. Очень многое можно построить, зная хотя бы одну из таких функций. Если в криптографии что-то сводится к односторонним функциям, то считается, что это решение будет иметь достаточно высокую стойкость.

Немного более слабое допущение - односторонние перестановки. Мы докажем, что если существуют псевдослучайные генераторы, то существуют и односторонние функции. Обратное утверждение будет получено позднее, а в этой лекции мы построим псевдослучайные генераторы на основе односторонних перестановок.

1.4 Физические генераторы

Альтернатива генераторам - настоящие случайные числа. Для генерации случайных чисел можно использовать различные физические явления, помехи, траекторию движения мыши, частоту нажатия клавиш пользователем. Недостатки наблюдений физических явлений - их дороговизна по сравнению с математическими генераторами, также могут потребоваться усилия со стороны пользователя. К тому же нет никакой доказуемой случайности. Возможны также проблемы с балансом между частотой появляющихся значений, их корреляцией.

Поэтому часто на практике получают последовательность случайных битов одним из указанных способов, а потом проводят над ними дополнительные преобразования, чтобы быть полностью уверенным в отсутствии связей между ними.

2 Односторонние функции и генераторы

Напомним, что функция $F : \{0,1\}^n \to \{0,1\}^m$ называется односторонней функцией, если

1) функция F вычислима за полиномиальное время;

2) не существует полиномиального алгоритма, который верно вычисляет F^{-1} с *хорошей вероятностью*;

2') существует предикат $h : \{0,1\}^n \to \{0,1\}$, т.ч. по $F(x)$ *трудно* вычислить $h(x)$.

Односторонняя функция называется односторонней перестановкой, если она является биекцией.

Теорема 1 (PRG $\Rightarrow$ OWF) *Псевдослучайный генератор $G : \{0,1\}^n \to \{0,1\}^{n+1}$ является односторонней функцией.*

Доказательство.

Итак, пусть есть функция-генератор, которая по строчкам длины n выдает строчки длины $n+1$. Нужно доказать, что никакой алгоритм A по второй строчке не смог бы получить первую.

Предплолжим противное - существует алгоритм обращения. Докажем, что тогда бы существовал алгоритм, который бы отличал случайные числа x от псевдослучайных y. Пусть алгоритм A с вероятностью α определяет x по $G(x)$: $P[A(G(x)) = x] = \alpha$. Тогда построим алгоритм A' следующим образом:

- A' считает $A(y)$;
- Если $G(A(y)) = y$, то A' выдает ответ "y - псевдослучайное", иначе - "y - случайное".

Если y изначально было сгенерировано функцией G(x), то после проверки алгоритм A' с вероятностью α выдаст ответ "y - псевдослучайное". Иначе, рассмотрим 2 случая:

- $y \notin G(x)$. В этом случае не важно, что выдаст алгоритм A, функция G на выходе никогда не получит y.
- $y \in G(x)$. Вероятность того, что алгоритм A' внутри этого случая скажет "y - псевдослучайное равна $1/2$.

Следовательно, итоговая вероятность того, что A' ответит "y - псевдослучайное", равна $\alpha/2$. Построили алгоритм, который со значительной вероятностью отличает псевдослучайное число от случайного, что противоречит определению псевдослучайного генератора. □

Теорема 2 (OWP ⇒ PRG) *Если существуют односторонние перестановки, то можно построить и псевдослучайный генератор* $G : \{0,1\}^n \to \{0,1\}^{n+1}$

Доказательство.

Пусть существует односторонняя перестановка $f(x)$, ее трудный бит $h(x)$.

Идея доказательства: покажем, что функция, которая строит по x цепочки вида $f(x), h(x)$, является псевдослучайным генератором. При заданном x возможны два варианта цепочек - $f(x), h(x)$ и $f(x), \bar{h}(x)$. Докажем, что если полученные с помощью такого псевдослучайного генератора биты можно отличить от случайных, то тогда можно было бы по $f(x)$ вычислить $h(x)$.

Предположим противное - пусть существует алгоритм A, который со значимой вероятностью отличает случайные биты от сгенерированных: $P_1(A(f(x), h(x)) = 1) - P_2(A(y) = 1) = \mu > 1/n^k$. Тогда по A построим новый алгоритм A', который будет вычислять $h(x)$. В зависимости от $h(x)$ и сгенерированной строчки получим 4 случая:

	A думает, что $h(x) = 0$	A думает, что $h(x) = 1$
$h(x) = 0$	$f(x), 0$	$f(x), 1$
$h(x) = 1$	$f(x), 0$	$f(x), 1$

В каждой группе должно быть поровну строк, т.к. $h(x)$ - сбалансированная функция (трудный бит обязан быть сбалансированным, т.к. иначе можно было бы точнее предсказать его значение). Пусть вероятность попадания строчки в одну из групп выражается следующей таблицей:

	A думает, что $h(x) = 0$	A думает, что $h(x) = 1$
$h(x) = 0$	α	β
$h(x) = 1$	γ	δ

Построим алгоритм A'_1, который будет угадывать $h(x)$ по $f(x)$ следующим образом: если $A'_1(f(x), 0) = 1$, то ответ "$h(x) = 0$", иначе - "$h(x) = 1$". С какой вероятностью A'_1 угадает трудный бит $h(x)$? Очевидно, он попадает в первый столбец таблицы.

Следовательно,

$$\left.\begin{aligned} P_1(A(f(x),h(x))=1) &= \frac{\alpha+\delta}{2}, \\ P_2(A(y)=1) &= \frac{\alpha+\beta+\gamma+\delta}{4} \end{aligned}\right\} \quad \Rightarrow \quad \mu = \frac{\alpha+\delta-\beta-\gamma}{4}$$

- Если $h(x)$ на самом деле равно 0, то алгоритм даст правильный ответ с вероятностью α.
- Если $h(x)=1$, то с вероятностью δ алгоритм ошибется. Следовательно, вероятность правильного ответа равна $1-\delta$. Таким образом, этот алгоритм имеет вероятность угадать трудный бит, равную $\frac{\alpha+(1-\gamma)}{2}$.

Аналогично построим алгоритм A_2': если $A(f(x),1)=1$, то ответ - "$h(x)=1$", иначе - "$h(x)=0$". Он будет угадывать правильный ответ с вероятностью $\frac{\delta+(1-\beta)}{2}$.

Алгоритм, который случайным образом выбирает между A_1' и A_2' и верит ему, будет угадывать трудный бит с существенной вероятностью $\frac{1}{2}+\frac{\alpha+\delta-\beta-\gamma}{4}$, что противоречит существованию односторонней перестановки. □

Теорема 3 (OWP ⇒ PRG - часть 2) *Если есть псевдослучайный генератор $G : \{0,1\}^n \to \{0,1\}^{n+1}$, то можно построить и генератор $G' : \{0,1\}^n \to \{0,1\}^{f(n)}$ для любого полинома f*

Доказательство.

Будем генерировать последовательности битов следующим образом:
$f(f(x)), h(x), h(f(x))$
$f^{(3)}(x), h(x), h(f(x)), h(f^{(2)}(x))$
...
$f^{(l)}(x), h(x), \cdots, h(f^{(l-1)}(x))$

f-односторонняя перестановка. Пусть существует алгоритм A, который отличает эту строку от совсем случайной $x, b_1, \cdots, b_l$ с вероятностью μ. Для доказательства используем гибридный метод из лекции о нулевом разглашении. Запишем промежуточные строки:
$f(x), b_1, b_2, \cdots, b_{l-3}, b_{l-2}, b_{l-1}, h(x)$
$f^2(x), b_1, b_2, \cdots, b_{l-3}, b_{l-2}, h(x), h(f(x))$
$f^3(x), b_1, b_2, \cdots, b_{l-3}, h(f(x)), h(f(x)), h(f^2(x))$
$\cdots$

Тогда алгоритм должен отличать какие-то 2 соседние строки с вероятностью μ/l. Запишем эти 2 соседних распределения:
$f^{(i)}(x), b_1, \cdots, b_{l-i-1}, b_{l-i}, h(x), \cdots, h(f^{i-1}(x))$
$f^{(i+1)}(x), b_1, \cdots, b_{l-i-1}, h(x), h(f(x)), \cdots, h(f^i(x))$
Подставим в верхнее распределение вместо x $f(x)$ и перепишем:
$f^{(i+1)}(x), b_1, \cdots, b_{l-i-1}, b_{l-i}, h(f(x)), \cdots, h(f^i(x))$
$f^{(i+1)}(x), b_1, \cdots, b_{l-i-1}, h(x), h(f(x)), \cdots, h(f^i(x))$

Они отличаются одним битом. Докажем, что можно построить алгоритм A', который по x и $f(x)$ будет вычислять $h(x)$. Пусть A' действует следующим образом:

- вычисляет $f^{i+1}(x)$;
- выбирает случайно бит b^* и строит последовательность $f^{(i+1)}(x), b_1, \cdots, b_{l-i-1}, b^*, h(f(x)), \cdots, h(f^i(x))$
- применяет алгоритм A. Если A выдает ответ 1, то он принимает в качестве $h(x)$ бит b^*, Если 0, то 1- b^*.

Вероятность того, что A' угадает нужную пару битов, равна $1/l$, после чего алгоритм A с вероятностью μ/l даст правильный ответ. Итак, построили алгоритм, который со значительной вероятностью μ/l^2 вычиляет трудный бит $h(x)$ односторонней перестановки $f(x)$. □

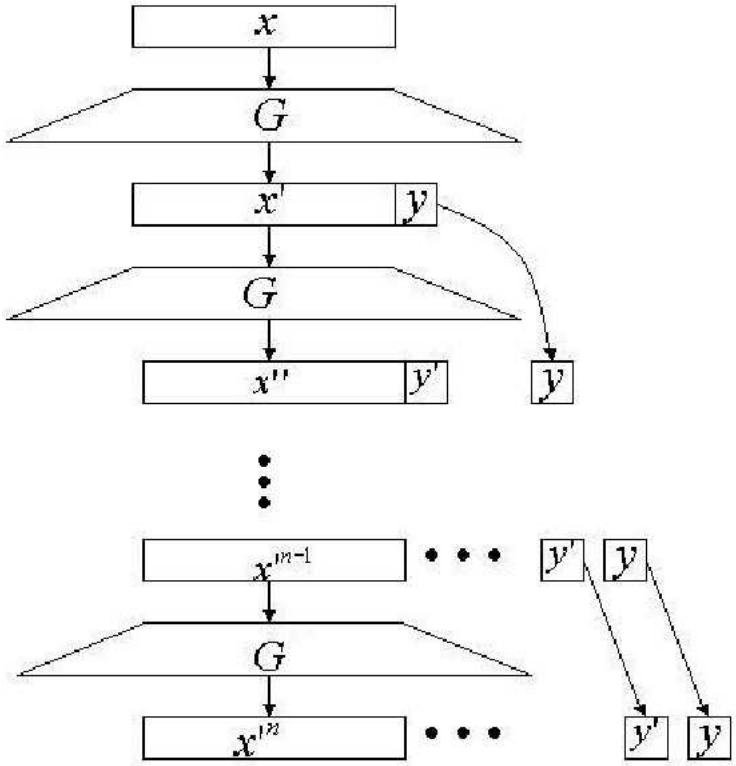

Рис.2. Механизм постоения псевдослучайных строк любой длины

Для построения псевдослучайного генератора $G^* : \{0,1\}^n \to \{0,1\}^{f(n)}$ будем использовать генератор G из теоремы 1, увеличивающий длину строки на 1, следующим образом:

1. пусть x - случайная строка длины n. Применим $G(x)$, получаем строку длины $n+1$, отщепляем последний бит y;

2. остаток будет иметь длину n, обозначим его x'. Повторяем шаг 1 для $x = x'$ до тех, пока не получим достаточное количество псевдослучайных битов.

3 Криптосистема на основе генератора

Как можно использовать псевдослучайный генератор для построения симметричных криптосистем? Один из способов - послать $m \oplus G(k)$ (k - секретный ключ), но тогда, перехватив 2 сообщения, злоумышленник сможет вычислить разность двух сообщений. Более безопасный способ - посылать пару $(r, G(k \oplus r))$, меняя r при каждой пересылке.

Пусть f — односторонняя перестановка с секретом, f имеет параметр k и вычисляется за полиномиальное время. Зная k, f^{-1} также считается за полиномиальное время. Не зная k, посчитать f^{-1} - вычислительно сложная задача (пример - $x^e \ mod \ n$ в RSA).

Пусть f — односторонняя перестановка с секретом. Для кодирования строки $b_1 b_2 \dots b_k$ возьмем случайное x и определим

$$E_f(b_1 b_2 \dots b_k) = f^{(k)(x)} \cdot (h(x) \oplus b_1) \cdot \dots \cdot (h(f^{(k-1)}(x)) \oplus b_k)$$

Алгоритм шифрования E называется семантически стойким, если для любой пары x, y распределения $E(x)$ и $E(y)$ являются вычислительно неразличимыми.

Следствие из Теоремы 3: Криптосистема на основе псевдослучайного генератора является семантически стойкой.