Псевдослучайные функции

Ю. Лифшиц*

5 декабря 2005 г.

План лекции

1. Предсказание следующего бита
2. Псевдослучайные функции
3. Стойкость против восстановления ключа
4. Задачи

1 Предсказание следующего бита

Определение: Функция $G : X \mapsto Y$ называется псевдослучайным генератором, если $G(\mathcal{U}_X)$ и $\mathcal{U}_Y$ вычислительно неразличимы ($\mathcal{U}_X$; $\mathcal{U}_Y$: равномерные распределения).

Неформально, генератор должен проходить все полиномиальные тесты на случайность. То есть никакой полиномиальный алгоритм не сможет отличить полностью случайную последовательность от созданной генератором.

Вопрос: Есть ли какой-нибудь конкретный тест, из которого следует случайность по всем остальным?

Ответ: Да, есть. Этот тест - предсказание следующего бита.

Определение: Распределение D ($D = G(X)$ - последовательность создаваемая генератором) проходит тест на предсказание следующего бита, если для любого полиномиального алгоритма A и любого p верно:

$$\Pr_{t \in D}[A(t_1, \ldots, t_p) = t_{p+1}] < \frac{1}{2} + \nu(|t|)$$

*Законспектировал А. Калюкин.

Пусть кто-то написал полиномиальный алгоритм , который предсказывает следующий бит. Тогда возьмчм последовательности, созданные генератором G, с вероятностями $a, b, c, d, \ldots$ Затем берчм p бит из одной и загружаем в . Если алгоритм угадывает следующий бит, то получаем выигрыш = вероятности последовательности. Если общий выигрыш ($\Pr_{t \in D}[A(t_1, \ldots, t_p) = t_{p+1}]$) не слишком больше $\frac{1}{2}$, то генератор G псевдослучаен.

Говорят, что генератор G проходит тест на следующий бит, если распределение его выходов проходит этот тест.

Теорема 1 *Если функция $G : \{0,1\}^n \mapsto \{0,1\}^N$ проходит тест на следующий бит IFF, то она является псевдослучайным генератором (проходит все полиномиальные тесты).*

Доказательство:
($\Rightarrow$) Очевидно.
($\Leftarrow$) Докажем, что если G проваливает какой-то полиномиальный тест , то G проваливает и некоторый тест A' на предсказание следующего бита. Используем для этого гибридный метод!!!

У нас есть распределение G_N , порожденное G. Определим G'_k как распределение префиксов длины к на выходах G. Будем случайно и равновероятно дописывать каждый префикс случайными битами до длины N. Получим распределение G_k.

Наблюдение: $G_0 = U$, и при росте к распределение постепенно становится всч более "псевдослучайным".

Применяем гибридный метод: алгоритм различает G_0 и G_n, значит для какого-то к он различает G_k и G_{k+1}!!!

Пусть $A = 1$ чаще на G_{k+1}.
Тест A':

1. Получает биты $t_1, \ldots, t_k$
2. Выписывает строки $T_0 = t_1, \ldots, t_k, 0, r_{k+2}, \ldots, r_N$ и $T_1 = t_1, \ldots, t_k, 1, r_{k+2}, \ldots, r_N$
3. Вычисляет $a_0 = A(T_0)$ и $a_1 = A(T_1)$
4. Если $a_0 = 0, a_1 = 1$, выдает “$t_{k+1} = 1$”
5. Если $a_0 = 1, a_1 = 0$, выдает “$t_{k+1} = 0$”
6. Если $a_0 = a_1$, выдает случайный ответ

2 Псевдослучайные функции

Определение: Функция $F : K \times D \to R$ называется псевдослучайной, если для любого полиномиального противника A выполнено:

$$|Pr[A^{F_k(\cdot)} = 1] - Pr[A^{R(\cdot)} = 1]| < \nu(log|D|)$$

Теорема 2 *Если псевдослучайные генераторы существуют, то псевдослучайные функции тоже.*

Доказательство:

Доказательство использует конструкцию двоичного дерева. Из предыдущей лекции, мы можем предположить без потери общности что $G : \{0,1\}^n \to \{0,1\}^{2n}$. Пусть $G(x) = G_1(x) \parallel G_2(x)$, где $G_1(x), G_2(x) \in \{0,1\}^n$, и ключ $s \in \{0,1\}^n$. Мы определим нашу псевдослучайную функцию F_s , которая работает как показано на Рисунке 1.

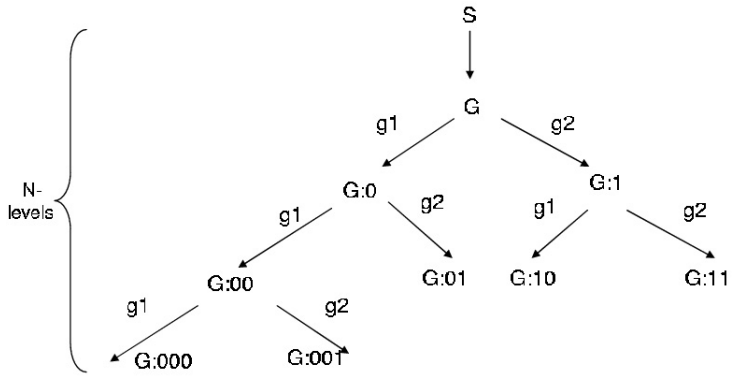

Рисунок 1: Конструкция перехода к PRF от PRG : на каждом уровне алгоритм выбирает верхнюю или нижнюю половину результата работы G, входными данными для которого являлась строка x. Конечный результат - это лист дерева.

Как показано на Рисунке 1, на каждом шаге алгоритм решает, что выводить, основываясь на текущем бите строки x. Дана n-битная строка x. Определим $G_{x_i}(y)$, чтобы вывести верхнюю половину битов $G(y)$ (то есть, $G_1(y)$) если x_i (i-тый бит x) равен 0, и нижнюю половину битов ($G_2(y)$), если x_i равен 1. Повторение этого метода для n (длина x) уровней моделирует псевдослучайную функцию как:

$$F_s(x) = G_{x_1}(G_{x_2}(\dots G_{x_n}(s)\dots))$$

Заметим, что семейство функций $\{F_s\}$, определенное выше, действительно PRF. Предположим противное: A - противник, работающий за время $\leq s$, следовательно $Pr[A^f s(x) = 1] - Pr[A^R(x) = 1] > \epsilon$. Покажем,

что по данному A, можно построить другого противника $'$, который может различить $PRGG$ и случайную функцию с высокой вероятностью. Для доказательства будем использовать гибридный метод (см. Рисунок 2). Слева на Рисунке 2 находится представление двоичного дерева F_s, которое мы назовем H_0, где S (0-ый уровень) случайно. Заметим, что первый уровень Но заключает в себе выполнение $G(s)$ для получения двух псевдослучайных n-битных значений, чтобы затем использовать их как входные данные к следующему уровню. Если мы удалим этот уровень и зададим вместо него два случайных значения r_0, r_1, то получим новый гибрид, который назовем H_1 (все значения на 1-ом уровне случайны). Повторяя этот процесс, получим H_2 (4 случайных строки на уровне 2), H_3 (8 случайных входных строк на уровне 3), и так далее пока мы не достигаем H_n, на котором всч двоичное дерево исчезает, и H_n становится случайной функцией R. Из того, что мы предполагали, что с хорошей вероятностью различает F_s(то есть, H_0) и $R(H_n)$, следует, что должен уметь отличать H_i и $H_(i+1)$ для некоторого i.

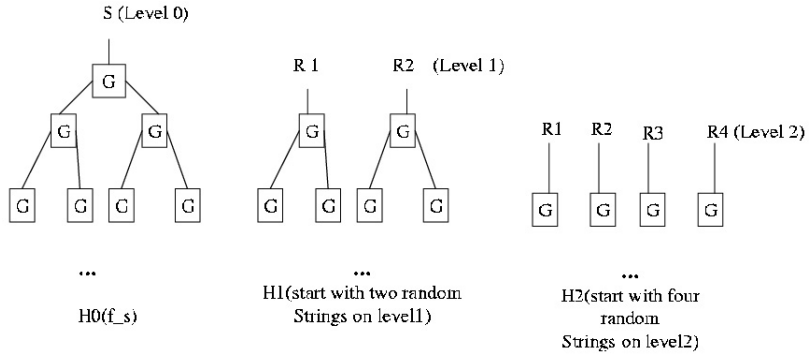

Рисунок 2: Примеры гибридов

Запишем это формально: существует $1 < i < n$ такое, что $Pr[A^{H_i} = 1] - Pr[A^{H_{i+1}} = 1] > \epsilon/n$.

Также определим l_x, где $1 \leq x \leq n$, чтобы обозначить промежуточный вывод двоичного дерева на входную строку x. Таким образом для F_s, $l_{0x} = G_0(l_x)$ и $l_{1x} = G_1(l_x)$, где $l_0 = G_0(s)$, $l_1 = G_1(s)$. Теперь мы готовы, чтобы построить $'$, моделируя A. По входным данным F' случайным образом генерирует список вывода $y_1, z_1, \ldots, y_s, z_s$, где $y_i, z_i \in \{0,1\}^n$ и s - верхняя граница времени выполнения A и число запросов, которые может сделать A. Затем A' строит новый гибрид H следующим образом:

1. Пусть $H = H_0$
2. Смоделируем на H, где - это запросы $H(x)$,
   (a) Найдчм самый длинный общий префикс p между x и $_i$, где $_i$ - некоторое значение которое H запрашивало до этого

(b) Если самый длинный общий префикс $p < i$, то пусть u будет строкой из первых i битов x, возьмчм следующую неиспользованную пару (y_i, z_i), и положим $l_{u0} = y_i l_{u1} = z_i$. Вычислим $H(x)$ основываясь на новых значениях l_{u0} и l_{u1}.

(c) С другой стороны l_{u0} и l_{u1} должно уже быть предварительно назначены для некоторых (y_i, z_i). Таким образом мы вычисляем непосредственно H(x)

Отметим, что гибрид может быть создан за полиномиальное время, потому что он сделан вертикально. Игнорируя пути в двоичном дереве, которые A не использует, может видеть s путей, так как может осуществлять только s запросов. Из конструкции, описанной выше, следует, что если y_i, z_i - случайные строки, то получившийся H будет H_{i+1}, так как в H все на $i+1$ уровне находятся случайные числа. если y_i, z_i - это строки сгенерированные PRG, то H совпадает с H_i, так как i-тый уровень случаен, но значения на $i+1$ уровне генерированы PRG. Поэтому A в состоянии различить H_i и H_{i+1},а следовательно $'$ может определить, что F - псевдослучайный генератор или функция, выбранные случайно, противоречие.

3 Стойкость против восстановления ключа

Определение: Семейство функций F_k называется стойким против восстановления ключа, если для любого полиномиального противника A верно, что:

$$Pr_{k \in K}[A^{F_k} = k] < \nu(|k|)$$

Теорема 3 *Псевдослучайные функции обладают стойкостью против восстановления ключа.*

Доказательство:

1) Пусть $F : Keys(F) \times D \to R$, будет семейством функций, и пусть B будет алгоритмом, который получает ожидаемое значение для функции $g : D \to R$, и выводит строку. Рассмотрим эксперимент($Exp^{kr}_{F,B}$):

$$K \xleftarrow{R} Keys(F)$$

$$K' \leftarrow B^{F_K}$$

Если $K = K'$ тогда возвращаем 1 иначе возвращаем 0

kr-advantage для B определим как

$$Adv^{kr}_{F,B} = P[Exp^{kr}_{F,B} = 1].$$

Для любых $t, q,$ определим kr-advantage для F как:

$$Adv^{kr}_{F}(t, q, \mu) = max_B\{Adv^{kr}_{F,B}\}$$

где максимум берчтся по всем B, имеющим временную сложность t и делающих не более q запросов ожиданий, сумма длин этих запросов должна быть μ бит.

2)Пусть $F : \{0,1\}^k\{0,1\}^l \to \{0,1\}^L$ - семейство функций. Тогда для любых t, q, где $q < 2^l$, мы получим неравенство (1):

$$Adv_F^{kr}(t,q,ql) \leq Adv_F^{prf}(t', q+1, (q+1)l) + \frac{1}{2^L}$$

и кроме того, если $L = l$, то (2):

$$Adv_F^{kr}(t,q,ql) \leq Adv_F^{prp-cpa}(t', q+1, (q+1)l) + \frac{1}{2^L - q},$$

где мы задачм t' как t плюс время для одного вычисления F.

3)Докажем пункт 2.

Мы доказываем первое уравнение и затем кратко указываем, как изменить доказательство, чтобы доказывать второе уравнение. Мы покажем, что по любому заданному противнику B, ресурсы которого ограничены t, q, ql мы можем построить противника A_B, используя ресурсы $t', q+1, (q+1)l$, такой что(*):

$$Adv_{F,B}^{kr} \leq Adv_{F,A_B}^{prf} + \frac{1}{2^L}$$

Если это истинно тогда, мы можем записать неравенство $(*)$ следующим образом:

$$\begin{aligned} Adv_F^{kr}(t,q,\mu) &= max_B\{Adv_{F,B}^{kr}\} \\ &\leq max_B\{Adv_{F,A_B}^{prf} + 2^{-L}\} \\ &\leq max_A\{Adv_{F,A}^{prf} + 2^{-L}\} \\ &= Adv_F^{kr}(t, q+1, (q+1)l) + 2^{-L} \end{aligned}$$

Максимум, в случае B, взят по всем противникам, ресурсы которых - t, q, ql. Во второй строке, мы применяем неравенство $(*)$. В третьей строке, мы максимизируем по всем A, чьи ресурсы - $t, q+1, (q+1)l$. Неравенство на третьей строке истинно, потому что этот набор включает всех противников вида A_B. Последняя строка - просто по определению. Таким образом, остается показать, как строить A_B так, чтобы неравенство $(*)$ выполнялось.

Согласно пункту 1, противник A_B будет обеспечен ожиданием для функции $g : \{0,1\}^l \to \{0,1\}^L$, и будет пытаться определить, в какой области оно существует. Чтобы сделать это нужно использовать противника B как подпрограмму. Рассмотрим описание этого, сопровождаемое объяснением и анализом.

Противник A_B^g

$i \leftarrow 0$

Выполнить противника B, отвечая на его запросы ожидания следующим образом

Когда B делает запрос ожидания x, выполняем

$i \leftarrow i+1; x_i \leftarrow x$
$y_i \leftarrow g(x_i)$
Возвратить y_i к B как ответ

Пока B не останвит и выведет ключ K'

Пусть x - это l-битная строка не из набора $\{x_1, \ldots, x_q\}$

$y \leftarrow g(x)$

Если $F(K', x) = y$ тогда возвратить 1 иначе - 0

Как было отмечено A_B выполняет B и непосредственно обеспечивает ответы на запросы ожидания B через ожидание g. Когда B завершается, то возвращает некоторую k-битную строку K', которую A_B проверяет, сравнивая $F(K'x)$ с $g(x)$. Здесь x - значение, отличное от любого, запрашиваемого B, и верно, что такое значение может быть найдено, т.к. в формулировке утверждения мы требуем, чтобы $q < 2^l$. Теперь мы потребуем следущего:

$$P[Exp^{prf-1}_{F,A_B} = 1] \geq Adv^{kr}_{F,B}$$

$$P[Exp^{prf-0}_{F,A_B} = 1] = 2^{-L}.$$

Вкратце объясним эти требования, но сначала позволим себе использовать их, чтобы закончить. Из выщесказанного следует, что при вычиталнии мы получим:

$$Adv^{prf}_{F,A_B} = P[Exp^{prf-1}_{F,A_B} = 1] - P[Exp^{prf-0}_{F,A_B} = 1] \geq Adv^{kr}_{F,B} - 2^{-L}$$

Перестановка слагаемых дает нам уравнение $(*)$. Остается объяснить уравнения $(**)$.

Уравнение $(**)$ истинно, потому что в Exp^{prf-1}_{F,A_B}g–F$_K$ для некоторого K, который является ожиданием для B также как и в $Exp^{kr}_{F,B}$. Если B успешно выполнилось, т.е. ключ K', который выводит B, равняется K, то A_B возвращает 1. (Возможно, что A_B мог бы возвратить 1 даже при том, что B выполнилось бы неудачно. Это случилось бы если $K' = K$, но $F(K', x) = F(K, x)$. Причина этого - то, что
$P[Exp^{prf-1}_{F,A_B} = 1] \geq$Adv$^{kr}_{F,B}$ вместо того, чтобы просто равняться.) Уравнение $(**)$ истинно потому что в Exp^{prf-0}_{F,A_B} функция g является случайной, и т.к. x никогда не запрашивалось B,поэтому значение $g(x)$ непредсказуемо для B. Предположим, что $g(x)$ выбирается только, когда x запрашивается g. В таком случае K', а следовательно и $F(K', x)$, уже определен. Таким образом

$g(x)$ имеет вероятность 2^{-L} на успешный исход. Отметим, что это истинно независимо от того, как хорошо B пытается приблизить $F(K', x)$ к $g(x)$.

Для доказательства уравнения (2) мы найдчм сокращение $B \rightarrow A_B$ с таким свойством, что (**):

$$Adv_{F,B}^{kr} \leq Adv_{F,A_B}^{prp-cpa} + \frac{1}{2^L - q}$$

Сокращение идентично приведчнному выше, то есть противник A_B - такой же. Проанализировав мы обнаружим, что

$$P[Exp_{F,A_B}^{prp-cpa-1} = 1] = Adv_{F,B}^{kr}$$

$$P[Exp_{F,A_B}^{prp-cpa-0} = 1] \leq \frac{1}{2^L - q}.$$

Вычитая результат получим

$$Adv_{F,A_B}^{prp-cpa} = P[Expprp - cpa - 1_{F,A_B} = 1] - P[Exp_{F,A_B}^{prp-cpa-0} = 1]$$

$$Adv_{F,A_B}^{prp-cpa} \geq Adv_{F,B}^{kr} - \frac{1}{2^L - q}$$

и перестановка членов дает уравнение (**). Первое уравнение истинно по той же причине,что и прежде. Второе уравнение истинно, потому что в области 0 карта g теперь случайная перестановка из l-битной в l-битную. Таким образом $g(x)$ принимает одно из $2^L - q$y$_1, \ldots, y_q$. (Где $L = l$.)

4 Задачи

1. Постройте из псевдослучайного генератора $G : B^n \rightarrow B^N$ псевдослучайную функцию $F : n \times \log N \rightarrow \{0, 1\}$.

2. Докажите, что если существуют псевдослучайные функции, то псевдослучайные генераторы тоже существуют.

5 Источники

1. Предсказание следующего бита - Голдвассер-Белларе, стр. 48-49
   Ссылка: http://www.cs.ucsd.edu/users/mihir/papers/gb.pdf

2. Построение псевдослучайных функций
   Ссылка: http://www.cs.berkeley.edu/ luca/cs276/notes/lecture10.ps

3. Стойкость против восстановления ключа - Голдвассер-Белларе, стр. 70-75
   Ссылка: http://www.cs.ucsd.edu/users/mihir/papers/gb.pdf