Записки научных
семинаров ПОМИ
Том 358, 2008 г.

Э. А. Гирш*, **Д. Ю. Григорьев**, **К. В. Первышев****

ИЕРАРХИИ ПО ВРЕМЕНИ С НЕРАВНОМЕРНОЙ ПОДСКАЗКОЙ ДЛЯ КРИПТОГРАФИЧЕСКОГО ОБРАЩЕНИЯ ФУНКЦИЙ

1. Введение

Один из наиболее важных вопросов криптографии – сложность обращения функций: цифровые подписи и криптосистемы (даже с секретным ключом) имеют право на существование, только если существует функция, которую просто вычислить, но трудно обратить (см., напр., [6]). В этой статье мы изучаем естественный смежный вопрос: верно ли, что противник, располагающий бóльшим временем, может обратить больше функций, – т.е. криптографический аналог классической задачи о наличии *иерархии по времени*.

Формальное утверждение о криптографической иерархии по времени приведено в теореме 1, показывающей существование иерархии по времени для обращения функций, вычислимых за время $O(n^w)$. Для простоты изложения начнём с рассмотрения постановки задачи в наихудшем случае, которая также вызывает интерес и сама по себе. В естественном предположении, что $\mathbf{P} \neq \mathbf{NP}$, по любому $\mathbf{NP}$-полному языку L из класса $\mathbf{NTime}[n^w]$ можно построить одностороннюю функцию $f(x,y)$ следующим образом: пусть $M(x,y)$ – детерминированная полиномиальная по времени машина, которая, имея на входе строку x и предполагаемое решение y, проверяет, что $x \in L$; положим $f(x,y) = (1,\, x)$, если $M(x,y) = 1$, и $f(x,y) = (0,\, x,\, y)$ в противном случае (см., напр., [8]). Поэтому теорема об иерархии может быть сформулирована следующим образом:

$$\mathbf{DTime}[n^k] \cap \mathbf{NTime}[n^w] \stackrel{?}{\neq} \mathbf{DTime}[n^l] \cap \mathbf{NTime}[n^w] \qquad (1)$$

(где $n^w = \mathrm{poly}(|x| + |y|)$ — сложность вычисления функций, которые

*При частичной поддержке фонда “Династия”, а также грантов INTAS 04-77-7173, РФФИ 08-01-00640-а и гранта Президента РФ для поддержки ведущих научных школ НШ-4392.2008.1.
**При поддержке стипендии фонда Focht-Powell .

мы пытаемся обратить). Классы языков $\mathbf{DTime}[n^k]$ и $\mathbf{DTime}[n^l]$, где $l > k \geqslant w$, отражают сложность противников, пытающихся обратить функции, определённые как сказано выше.

Хотя иерархии по времени для детерминированных и недетерминированных вычислений известны с 60-х годов [9, 3], иерархия (1) не может быть доказана теми же методами, поскольку соответствующий класс языков $\mathbf{DTime}[n^k] \cap \mathbf{NTime}[n^w]$ не является *синтаксическим*, т.е. по детерминированной и недетерминированной машинам невозможно удостовериться, что они принимают один и тот же язык. В последнее время наметился прогресс в доказательстве иерархий по времени для таких *семантических* классов, или классов с дополнительной гарантией (*promise*). Данная работа мотивирована этими новыми результатами; мы развиваем используемую в них технику для того, чтобы доказать наши теоремы об иерархии как для сложности в наихудшем случае, так и в криптографическом варианте (см. более подробное изложение ниже). Далее мы приводим краткий исторический обзор иерархий по времени для синтаксических и семантических сложностных классов, более подробно определяем криптографический случай и неформально объясняем наши методы.

Исторический обзор. В 60-х годах XX-го века Хартманис и Стернс [9] показали, что для всяких констант k и l, для которых $1 \leqslant k < l$, выполняется *строгое* включение $\mathbf{DTime}[n^k] \subsetneq \mathbf{DTime}[n^l]$, где $\mathbf{DTime}[n^d]$ – класс языков, принимаемых многоленточными детерминированными машинами Тьюринга, заканчивающими свою работу за $O(n^d)$ шагов. В следующем десятилетии Кук [3] доказал иерархию по времени для недетерминированных вычислений: $\mathbf{NTime}[n^k] \subsetneq \mathbf{NTime}[n^l]$. Ещё два доказательства этого результата были позже даны Зейферасом, Фишером и Меером [15] и Заком [16].

Однако такими методами не удаётся доказать теоремы об иерархии для вероятностных вычислений, в частности для классов языков **BPTime**, **RTime** и **ZPTime**, распознаваемых вероятностными алгоритмами с ограниченной дву-, односторонней и нулевой ошибкой соответственно. Основным препятствием является то, что в этих классах имеются ограничения на вероятность ошибки, а множество машин, удовлетворяющих им, может не являться рекурсивно перечислимым. Следовательно, обычная диагонализация приводит к машине, не определяющей язык, удовлетворяющий этим ограничениям.

Вследствие отсутствия подходящих методов, множество проблем в этой области не решены до сих пор; в том числе $\mathbf{BPTime}[n]$ vs

$\mathbf{BPP} = \bigcup_d \mathbf{BPTime}[n^d]$ и аналогичные вопросы для классов **RTime** и **ZPTime**.

В 2002-м году Барак [2] предложил новый метод доказательства иерархий по времени, использующий понятие *оптимального алгоритма*, введённое Левиным [11]. Этот метод был далее развит Фортноу и Сантанамом [4], а также Голдрейхом, Суданом и Тревисаном [7], которые доказали, что существует иерархия по времени для **BPTime** с одним битом подсказки, т.е. для класса языков, распознаваемых вероятностными алгоритмами с ограниченной двусторонней ошибкой, использующими один бит подсказки для каждой длины входа (ограничения на вероятность ошибки должны здесь выполняться, только когда алгоритму дана *правильная* подсказка). Более формально, они доказали, что $\mathbf{BPTime}[n^k]/1 \subsetneq \mathbf{BPTime}[n^l]/1$, где $k < l$. Такой же результат справедлив и для класса **RTime** [5].

В 2005-м году Фортноу, Сантанам и Тревисан [5] предложили доказывать иерархию по времени с небольшой подсказкой одновременно для многих семантических классов. По существу их метод позволяет доказать иерархию для любого разумного класса вычислений, определённого при помощи недетерминированных (в том числе, вероятностных) полиномиальных по времени машин с

$$a(n) = O(\log n \cdot \log\log n)$$

битами подсказки, например для классов $\mathbf{UTime}/a(n)$ (недетерминированные вычисления с однозначно определённым принимающим путём), $\mathbf{MATime}/a(n)$ (протоколы типа Мерлин-Артур с ограниченным по времени Артуром), $\mathbf{AMTime}/a(n)$ (протоколы типа Артур-Мерлин с ограниченным по времени Артуром), $\mathbf{ZPTime}/a(n)$, $\mathbf{RTime}/a(n)$, $(\mathbf{NTime} \cap \mathbf{coNTime})/a(n)$ (формальные определения см. в [1]).

Недавно ван Мелкебек и Первышев [12, 14] для этих же классов вычислений доказали наличие иерархии по времени с *одним* битом подсказки. В частности, они доказали, что

$$\mathbf{ZPTime}[n^k]/1 \subsetneq \mathbf{ZPTime}[n^l]/1.$$

Результаты этой работы. В данной статье мы доказываем две теоремы об иерархии для обращения односторонних функций с одним битом подсказки. Именно, мы доказываем вариант утверждения (1)

для сложности в наихудшем случае: если $\mathbf{P} \neq \mathbf{NP}$, то для всякой положительной константы $w \geqslant 1$ и всяких $l > k \geqslant w$ выполняется

$$(\mathbf{DTime}[n^k] \cap \mathbf{NTime}[n^w])/1 \subsetneq (\mathbf{DTime}[n^l] \cap \mathbf{NTime}[n^w])/1;$$

мы также доказываем аналогичную теорему в криптографических определениях, в которых один бит подсказки используется также для вычисления самой функции. Наш результат *не* следует из общих результатов [12, 14], поскольку рассматриваемые вычисления не являются частным случаем какой-либо общей вычислительной модели. Вместо этого речь идёт о весьма специальной модели взлома некоторого "криптографического примитива" в наихудшем случае; поэтому для того, чтобы доказать иерархию по времени, недостаточно показать, что большее количество времени позволяет решить некоторую (произвольную) задачу, которая не разрешима за меньшее время. Что необходимо доказать, так это то, что большее количество времени предоставляет возможность решить задачу взлома некоторой реализации криптографического примитива. Тем не менее метод, используемый нами для сложности в наихудшем случае, похож на метод, использованный недавно в [14] для доказательства иерархий по времени для алгоритмов с подсказкой. Также он похож на один из методов доказательства иерархии по времени для недетерминированных алгоритмов, предложенный ранее в [15].

Для доказательства результата в криптографических определениях мы используем ещё несколько приёмов, которые неформально излагаются ниже.

Сильно односторонняя функция – это функция, которую легко вычислить, но трудно обратить. Более точно, (i) она вычислима за полиномиальное время, (ii) она является "честной", т.е. каждая строка в её образе имеет прообраз полиномиальной длины, и (iii) никакой полиномиальный по времени вероятностный алгоритм не обращает эту функцию для бесконечно многих длин входов с существенной вероятностью. Эта вероятность успеха зависит от длины n входа. *Существенной вероятностью* мы называем такую, которая больше $1/r(n)$ для какого-то полинома $r(n)$.

Мы пишем, что (полиномиальный по времени вероятностный) противник M $r(n)$-*взламывает* функцию f, если M обращает f с вероятностью хотя бы $1/r(n)$ на бесконечном количестве длин входов. Очевидно, если имеются полином r и противник, который $r(n)$-взламывает f, то функция f не является сильно односторонней.

Возникает естественный вопрос: какое конкретно время требуется взломщикам для взлома функции, в частности, возможно ли взломать больше функций за большее время?

В предположении, что сильно односторонние функции существуют, мы отвечаем на этот вопрос утвердительно, доказывая, в частности, что для всякого полинома r и любого $k > 1$ существует функция G, сохраняющая длину, вычислимая за линейное время с одним битом подсказки, для которой имеется вероятностный полиномиальный по времени противник, $r(n)$-взламывающий её, но не имеется таковых противников, работающих время $O(n^k)$ (даже с логарифмическим количеством битов подсказки). (См. теорему 1, в которой даётся более конкретная оценка на время работы успешного противника и более сильная формулировка, позволяющая этому противнику взломать функцию с вероятностью более высокой, чем слабые противники.) Другими словами, мы доказываем *теоремы об иерархии по времени* для противников для сильно односторонних функций с одним битом подсказки. Отметим, что сам успешный противник не будет использовать никакой подсказки.

Наши результаты с необходимостью являются условными: как указал нам один из анонимных рецензентов, равенство $\mathbf{P} = \mathbf{NP}$ влечёт коллапс иерархии для сложности в наихудшем случае. В самом деле, рассмотрим язык из $(\mathbf{P} \cap \mathbf{NTime}[n])/1$. Этот язык распознаётся некоторой недетерминированной линейной по времени машиной M с подсказкой. Получив эту подсказку и вход x, мы можем детерминированно построить индивидуальную задачу выполнимости (т.е. булеву формулу) длины $|x|^{l_1}$, которая имеет значение $M(x)$, т.е. ответ, даваемый на входе x машиной M с подсказкой. Если $\mathbf{P} = \mathbf{NP}$, то эта индивидуальная задача может быть решена за время $|x|^{l_1 \cdot l_2}$. Следовательно, язык принадлежит классу $(\mathbf{DTime}[n^{l_1 \cdot l_2}] \cap \mathbf{NTime}[n])/1$ (заметим, что описанный детерминированный алгоритм требует той же самой подсказки, что и недетерминированная машина M).

Используя универсальную одностороннюю функцию Левина, можно также доказать, что если не существует односторонних функций, то наступает коллапс иерархии и в криптографических определениях (т.е. справедливо утверждение, обратное следствию 1).

Предлагаемый метод. Взяв за основу какую-либо сильно одностороннюю функцию f, мы строим функцию F, которая обратима проще, чем f, на некоторых длинах входов. Сложность обращения функции F постепенно снижается на (бесконечной) последовательно-

сти длин π, $l(\pi)$, $l(l(\pi))$, ... для простого числа π и подходящей "раздувающей" функции l; функция F совпадает с f на входах длины π и легко обратима на бóльших длинах входов данной последовательности. Рассмотрим какого-нибудь противника M, $r(n)$-обращающего функцию F за время $O(n^k)$. Можно доказать, что для некоторого $n = l(l(\ldots(\pi)\ldots))$ противник M взламывает функцию F с вероятностью $1/r(l(n))$ на длине $l(n)$, но не взламывает её на длине n с вероятностью $1/r(n)$. Наша конструкция функции F позволяет вычислить (и обратить) F на длине n при помощи запросов к F (либо противнику) длины $l(n)$. Таким образом, мы можем сконструировать другого противника N, который может использовать M на входах длины $l(n)$ для обращения функции F на длине n.

Однако наш взломщик обращает функцию F для входов длины n лишь с вероятностью $1/r(l(n))$, что всё ещё меньше, чем требуемая вероятность $1/r(n)$. Чтобы преодолеть это препятствие, мы используем метод увеличения сложности (ср. [6]). Именно, наша функция F получается объединением нескольких экземпляров исходной функции f для меньшей длины входа. Подставляя интересующую нас строку в разные экземпляры результата такой функции и применяя противника M несколько раз на полученных входах длины $l(n)$, мы увеличиваем вероятность успеха до $1/r(n)$ (а если требуется – и больше). Естественно, это делается ценой увеличения времени работы.

Несложно изменить F таким образом, чтобы противник N $r(n)$-взламывал её, а противник M – нет (можно просто использовать один бит подсказки, чтобы устранить "неинтересные" длины входов из нашей последовательности). Однако наша цель – чтобы *любой* противник M_i, работающий время $O(n^k)$, не мог $r(n)$-взломать F. Поэтому наш противник N моделирует всех противников M_1, M_2 и т.д., работающих время $O(n^k)$, аналогично конструкции оптимального алгоритма Левина [11].

Доказательство теоремы для сложности в наихудшем случае несколько проще, в частности, поскольку нет необходимости применять метод увеличения сложности.

Для начала "раздуем" SAT (**NP**-полный язык, состоящий из всех выполнимых пропозициональных формул) таким образом, чтобы сложность вновь конструируемого (и по-прежнему **NP**-полного) языка B постепенно убывала на некоторых последовательностях длин входов.

Далее, построим детерминированный "оптимальный" алгоритм

OPT, работающий следующим образом. Получив вход, он раздувает его до определённой длины, моделирует на том, что получилось, детерминированные машины $M_1, \ldots, M_k$ со всеми возможными подсказками и использует самосводимость SAT, чтобы выбрать правильный ответ. Другими словами, OPT распознаёт B на "трудных" длинах входов, моделируя $M_1, \ldots, M_k$ на "более простых" длинах.

Рассмотрим теперь те длины входов, на которых OPT успешно распознаёт язык B, и назовём их *хорошими*. По построению алгоритма OPT должна существовать некоторая *хорошая* длина, на которой ни одна из машин $M_1, \ldots, M_k$ не распознаёт B (если $\mathbf{P} \neq \mathbf{NP}$).

Наконец, выбросим из B все слова *плохих* длин, получив таким образом новый язык C. Этот новый язык может быть легко распознан недетерминированной машиной с линейным ограничением по времени (мы используем представление формул, позволяющее проверить выполняющий набор за линейное время) с одним битом подсказки (которая сообщает, является ли данная длина входа *хорошей*). Этот язык также может быть распознан вариантом нашего "оптимального" алгоритма, использующим аналогичную подсказку.

Дальнейшие исследования. По сложившейся в криптографии традиции мы считаем противника успешным, если он взламывает функцию на бесконечном числе длин входов. Было бы также интересно рассмотреть более естественную формулировку в терминах сложности в среднем.

Организация статьи. В последующих разделах мы даём определения (раздел 2), доказываем основную теорему для криптографического случая (раздел 3) и доказываем её аналог для сложности в наихудшем случае (раздел 4).

2. Определения

Определение 1. *Функция принадлежит классу* $\mathbf{FTime}[t(n)]$, *если её можно вычислить за время* $O(t(n))$ *на многоленточной детерминированной машине Тьюринга. Определим также* $\mathbf{FP} = \bigcup_k \mathbf{FTime}[n^k]$. *Классы* $\mathbf{DTime}[t(n)]$ *и* $\mathbf{P}$ *содержат предикаты (т.е. языки) из* $\mathbf{FTime}[t(n)]$ *и* $\mathbf{FP}$; *классы* $\mathbf{NTime}[t(n)]$ *и* $\mathbf{NP}$ *содержат предикаты, вычислимые на подходящих многоленточных недетерминированных машинах Тьюринга.*

Обозначим через $\mathbf{PFTime}[t(n)]$ *класс всех многоленточных вероятностных машин Тьюринга, вычисляющих функции за время* $O(t(n))$

в наихудшем случае. Заметим, что классы $\mathbf{PFTime}[\cdot]$ *– это классы машин, а не функций; эти машины могут возвращать разные результаты в зависимости от используемых ими случайных чисел (в дальнейшем нас будет интересовать вероятность того, что подобная машина возвращает значение какой-то конкретной функции входа). Мы также предполагаем, что каждая из рассматриваемых нами полиномиальных машин снабжена конкретным "будильником", т.е. завершает свою работу не позднее конкретного полиномиального количества шагов; эта функция вида* $C \cdot n^k$ *также входит в описание машины. Наконец, определим* $\mathbf{PFP} = \bigcup_k \mathbf{PFTime}[n^k]$.

Если $\mathcal{C}$ *– сложностной класс (языков, функций или машин), мы пишем* $\mathcal{C}/a(n)$, *если машины Тьюринга, используемые в определении класса* $\mathcal{C}$, *снабжены подсказкой длины* $a(n)$ *на одной из их лент.*

Для каждого из определённых выше классов

$$\mathbf{DTime}[n^k],\ \mathbf{DTime}[n^k]/a(n),\ \mathbf{PFTime}[n^k] \text{ и } \mathbf{PFTime}[n^k]/a(n)$$

мы фиксируем легко перечислимый порядок $M_1, M_2, \ldots$ *используемых машин (но не подсказок), такой, что будильник машины* M_i *прерывает её после* $T_{M_i}(n) = C_{M_i} \cdot n^k$ *шагов для* $0 < C_{M_i} \leqslant i$.

Определение 2. *Функция* $f : \{0,1\}^* \to \{0,1\}^*$ *называется сильно односторонней функцией, надёжной относительно равномерного вероятностного противника (или, для краткости, просто сильно односторонней функцией), если* $f \in \mathbf{FP}$, f *– честная (т.е. для всякого* $y \in \operatorname{Im} f$ *имеется строка в его прообразе* $f^{-1}(y)$ *длины, ограниченной полиномом от* $|y|$*) и для всякого полинома* p *и всякого* $A \in \mathbf{PFP}$

$$\exists N\ \forall n > N \quad \Pr\{A(f(x)) \in f^{-1}(f(x))\} < \frac{1}{p(n)},$$

где вероятность берётся по входам x, *равномерно распределённым на* $\{0,1\}^n$, *и по случайным числам, используемым машиной* A.

Определение 3. *Мы называем машину* $A \in \mathbf{PFP}$ *успешным* $r(n)$*-противником для функции* $f : \{0,1\}^* \to \{0,1\}^*$, *если она является контрпримером к определению 2, т.е. для бесконечного количества длин входов* n *выполняется неравенство*

$$\Pr\{A(f(x)) \in f^{-1}(f(x))\} \geqslant \frac{1}{r(n)}. \tag{2}$$

Если неравенство (2) *выполняется для конкретной длины* n, *мы говорим, что* A $r(n)$-*взламывает* f *на этой длине.*

Замечание 1. Честная функция из **FP** является сильно односторонней тогда и только тогда, когда для неё не существует успешных $p(n)$-противников ни для какого полинома p.

Определение 4. *Сильно односторонние функции с подсказкой определяются аналогично.*

Определение 5. *Назовём функцию* $T\colon \mathbb{N}\cup\{0\} \to \mathbb{N}\cup\{0\}$ *допустимой (ср.* [13, 7.1]*; отметим, что нам не нужны ни монотонность, ни ограничения на память, имеющиеся в* [13]*), если строка* $0^{T(n)}$ *(т.е.* $T(n)$ *в унарной записи) вычислима за время* $O(n + T(n))$ *по любой строке длины* n *в качестве входа.*

3. За большее время можно обратить больше функций с большей вероятностью

Нам потребуется техническая лемма о конструкции раздувающей функции с некоторыми свойствами. Пусть $l^s(n)$ обозначает $\underbrace{l(\dots(l}_{s}(n))\dots)$.

Лемма 1. *Пусть* $\{\pi_1, \pi_2, \dots\}$ *– множество всех нечётных простых чисел. Пусть* $r(n) = R{\cdot}n^\rho$ *(где* $R > 0$ *и* $\rho \geqslant 0$ *– вычислимые константы) и* r' *– допустимая функция, такая, что* $1 < b \leqslant r'(n) \leqslant r(n)$ *(где* b *– константа). Тогда существует функция* $l : \mathbb{N} \to \mathbb{N}$, *такая, что*

1. $\exists n_0 \forall n \geqslant n_0\ (r'(n))^{l(n)/n} > 2 \cdot r(l(n))$.
2. $\forall n\ n | l(n)$.
3. $\forall s, s', j, j'\ ((s, j) \neq (s', j') \Rightarrow l^s(\pi_j) \neq l^{s'}(\pi_{j'}))$.
4. l *– допустимая функция.*
5. *Имея* $l^s(\pi_j)$ *в унарной записи, можно вычислить* π_j *за время* $O(l^s(\pi_j))$.
6. $l(n) = O\left(\frac{n \log n}{\log r'(n)}\right)$.

Доказательство. Во-первых, если $\rho = 0$, то подходящая линейная функция $l(n) = n{\cdot}\mathrm{const}$ удовлетворяет всем требуемым ограничениям. Поэтому будем предполагать, что r – не константа.

Начнём с построения функции $m(n)$, удовлетворяющей требованию 1, а затем заменим её на чуть большую $l(n)$, удовлетворяющую и

остальным условиям тоже. Взяв двоичный логарифм от неравенства в условии 1 (для $m(n)$), преобразуем его в

$$\frac{m(n)}{n}\log r'(n) \stackrel{?}{>} \log(2\cdot R\cdot m(n)^{\rho}), \tag{3}$$

т.е.

$$m(n) \stackrel{?}{>} \frac{n\rho}{\log r'(n)}\left(\log m(n) + \rho^{-1}(1+\log R)\right). \tag{4}$$

Пусть

$$\mu(n) = \left(1+\left\lceil\frac{1}{\log b}\right\rceil\right)\left\lceil\frac{n\rho}{\lceil\log r'(n)\rceil}\right\rceil, \qquad m(n) = 2\mu(n)\lceil\log\mu(n)\rceil.$$

Легко заметить, что $\mu(n) = \Theta(\frac{n}{\log r'(n)})$ и $m(n) = \Theta(\frac{n\log n}{\log r'(n)})$. Также

$$\begin{aligned}\mu(n) &\geqslant \left(1+\frac{1}{\log b}\right)\frac{n\rho}{\log r'(n)+1}\cdot\frac{\log r'(n)}{\log r'(n)} = \frac{1+\frac{1}{\log b}}{1+\frac{1}{\log r'(n)}}\cdot\frac{n\rho}{\log r'(n)}\\ &\geqslant \frac{n\rho}{\log r'(n)}\end{aligned}$$

и

$$\log m(n) + \rho^{-1}(1+\log R) = \log\mu(n) + \log\lceil\log\mu(n)\rceil + \text{const} < 2\log\mu(n)$$

для достаточно больших чисел n. Следовательно, неравенство (4) выполняется для всех достаточно больших n.

Наконец, пусть $l(n) = n\cdot t^2$, где $(t-1)^2 \leqslant \max\{\frac{m(n)}{n}, 1\} < t^2$. Легко заметить, что она удовлетворяет условию 1. Условие 2 тривиальным образом выполняется; условие 3 выполняется, поскольку π_j и $\pi_{j'}$ – единственные простые делители $l^s(\pi_j)$ и $l^{s'}(\pi_{j'})$ нечётной кратности; ясно также, что l – допустимая функция (условие 4). Чтобы убедиться в выполнении условия 5, заметим, что для вычисления числа π_j достаточно проверить не более $\sqrt{l^s(\pi_j)}$ возможных делителей и произвести для проверки кратности деление на каждый из них не более чем логарифмическое количество раз. Наконец, легко заметить, что

$$l(n) = O(m(n)) = O\left(\frac{n\log n}{\log r'(n)}\right).$$

В самом деле, если $\frac{m(n)}{n} \geqslant 16$, то $t > 4$ и $2(t-1)^2 > t^2$. Следовательно, $l(n) = n \cdot t^2 < 2n \cdot (t-1)^2 \leqslant 2n \cdot \frac{m(n)}{n} = O(m(n))$. В противном случае $\frac{m(n)}{n} < 16$. Итого, имеем $t^2 \leqslant 16$ и $l(n) = O(n)$. Вспомним, что $m(n) = \Theta(\frac{n \log n}{\log r'(n)})$ и $\log r'(n) = O(\log n)$. Значит, $m(n) = \Omega(n)$ и $l(n) = O(m(n))$.

Замечание 2. Если $r'(n) = r(n)$ (например, этот случай рассматривается в приводимом ниже следствии 1), то достаточно взять $l(n) = 2n$.

Следующая лемма утверждает, что если существуют сильно односторонние функции, то существует сильно односторонняя функция, сохраняющая длину, вычислимая за *линейное* время. Нам также понадобится вариант этой леммы, в котором исходная функция является перестановкой, сохраняющей длину; тогда сконструированная функция, вычислимая за линейное время, наследует это свойство. Эти леммы нам понадобятся, поскольку лучше, чтобы функции в нашей иерархии были вычислимы как можно проще (но, тем не менее, обращать их было бы трудно). При помощи этих двух лемм мы построим иерархию по времени таким образом, что функции будут вычислимы за *линейное время* лишь с одним *битом подсказки*. Для полноты мы приводим доказательства этих лемм (эти результаты общеизвестны; однако, похоже, нигде не опубликованы в формулировке с линейным временем).

Лемма 2 (ср. [6, 2.4.1])**.** *Пусть существует сильно односторонняя функция. Тогда имеется сохраняющая длину сильно односторонняя функция, вычислимая за линейное время.*

Доказательство. Пусть f – сильно односторонняя функция, вычислимая за время n^k, где, не умаляя общности, $k \geqslant 1$. Определим вспомогательную функцию t, которую мы будем использовать при построении сохраняющей длину односторонней функции g для того, чтобы справиться с тем, что у исходной функции f могут быть образы различной длины:

$$t(1 \circ s) = 11 \circ t(s),$$
$$t(0 \circ s) = 10 \circ t(s),$$
$$t(\epsilon) = \epsilon.$$

Определим новую одностороннюю функцию g как

$$g(x) = t(f(\bar{x})) \circ 0^{n-|t(f(\bar{x}))|}, \tag{5}$$

где $n = |x|$ и $\bar{x} = x_{1..\bar{n}}$ – префикс строки x длины $\bar{n} = \lfloor (n/2)^{1/k} \rfloor$. Функция t позволяет определить, где находится результат $f(\bar{x})$ в строке $g(x)$.

Ясно, что функция g сохраняет длину (заметим, что $|f(\bar{x})| \leqslant \bar{n}^k \leqslant n/2$). Ясно также, что g вычислима за линейное время. Остаётся доказать, что g трудно обратить. Мы получим противоречие, предположив, что противник M обращает g с существенной вероятностью, т.е. для некоторого полинома $p(n)$ для бесконечно многих n выполняется неравенство

$$\Pr\{M(g(x)) \in g^{-1}(g(x))\} > \frac{1}{p(n)}, \tag{6}$$

где вероятность берётся по входам x, равномерно распределённым на $\{0,1\}^n$, и по случайным числам, используемым машиной M.

Построим противника N, который обращает f с существенной вероятностью на бесконечно многих длинах входов. Противнику на вход подаётся $\bar{z} = f(\bar{x})$. Поскольку функция f честная, имеется не более $\mathrm{poly}(|\bar{z}|)$ различных вариантов длины $\bar{n}$ самой короткой из подаваемых строк $\bar{x}'$, такой, что $f(\bar{x}') = \bar{z}$. Наш противник N также должен выбрать (при помощи случайного выбора) число n, такое, что $\bar{n} = \lfloor (n/2)^{1/k} \rfloor$ – корректное значение для $\bar{n}$ и противник M взламывает g на входе n. Это также можно сделать с некоторой существенной вероятностью $1/\mathrm{poly}(\bar{n})$. Противник N ведёт себя следующим образом:

1. Выбрать n и положить $\bar{n} = \lfloor (n/2)^{1/k} \rfloor$.
2. Положить $z = t(\bar{z}) \circ 0^{n-|t(\bar{z})|}$.
3. Выдать $M(z)_{1..\bar{n}}$.

Обозначим $x = \bar{x} \circ \tilde{x}$, где $\tilde{x}$ равномерно распределено на $\{0,1\}^{n-\bar{n}}$. Имеем

$$\begin{aligned}
&\Pr\{N(f(\bar{x})) \in f^{-1}(f(\bar{x}))\} \\
&\qquad \geqslant \frac{1}{\mathrm{poly}(\bar{n})} \Pr\{M(t(f(\bar{x})) \circ 0^{n-|t(f(\bar{x}))|})_{1..\bar{n}} \in f^{-1}(f(\bar{x}))\} \\
&\qquad = \frac{1}{\mathrm{poly}(\bar{n})} \Pr\{f(M(g(x))_{1..\bar{n}}) = f(\bar{x})\} \\
&\qquad = \frac{1}{\mathrm{poly}(\bar{n})} \Pr\{g(M(g(x))) = g(x)\} \\
&\qquad \geqslant \frac{1}{\mathrm{poly}(\bar{n})} \cdot \frac{1}{p(n)} = \frac{1}{\mathrm{poly}(\bar{n})},
\end{aligned}$$

где вероятность берётся по $\bar{x}$, равномерно распределённым на $\{0,1\}^{\bar{n}}$, и по случайным числам, которые использует противник N. (Заметим, что второе неравенство выполняется, поскольку использование функции t гарантирует, что если $g(M(g(x))) = g(x)$, то $\bar{n}$ определено однозначно.)

Лемма 3. *Пусть существует сильно односторонняя перестановка. Тогда существует сильно односторонняя перестановка, вычислимая за линейное время. Более того, если существует семейство перестановок с секретом, функции, построенные согласно этой лемме, также образуют семейство перестановок с секретом.*

Доказательство. Пусть f – сильно односторонняя перестановка, вычислимая за время n^k, где, не умаляя общности, $k \geqslant 1$. Построим одностороннюю перестановку g, такую, что

$$g(x) = f(\bar{x}) \circ \tilde{x}, \tag{7}$$

где $n = |x|$ и $\bar{x} = x_{1..\bar{n}}$ – префикс строки x длины $\bar{n} = \lfloor n^{1/k} \rfloor$, а $\tilde{x} = x_{(\bar{n}+1)..n}$ – остающийся от него суффикс строки x.

Ясно, что g – перестановка. Также ясно, что g вычислима за линейное время. Остаётся доказать, что функцию g трудно обратить. Чтобы прийти к противоречию, предположим, что противник M обращает g с существенной вероятностью, т.е. для некоторого многочлена $p(n)$ для бесконечно многих n выполняется неравенство

$$\Pr\{M(g(x)) \in g^{-1}(g(x))\} > \frac{1}{p(n)}, \tag{8}$$

где вероятность берётся по x, равномерно распределённым на $\{0,1\}^n$, и по случайным числам, используемым M.

Построим противника N, который обращает f с существенной вероятностью на бесконечно многих длинах входов. Противнику на вход подаётся $\bar{z} = f(\bar{x})$. Противник N ведёт себя следующим образом:

1. Положить $\bar{n} = |\bar{z}|$ и выбрать n, такое, что $\bar{n} = \lfloor n^{1/k} \rfloor$.
2. Положить $z = \bar{z} \circ \tilde{z}$, где $\tilde{z}$ равномерно распределено на $\{0,1\}^{n-\bar{n}}$.
3. Выдать $M(z)_{1..\bar{n}}$.

Обозначим $x = \bar{x} \circ \tilde{x}$, где $\tilde{x}$ равномерно распределено на $\{0,1\}^{n-\bar{n}}$.

Имеем

$$\begin{aligned}\Pr\{N(f(\bar{x})) \in f^{-1}(f(\bar{x}))\} &\geqslant \frac{1}{\mathrm{poly}(\bar{n})} \Pr\{M(f(\bar{x}) \circ \tilde{z}) \in f^{-1}(f(\bar{x}))\} \\ &\geqslant \frac{1}{\mathrm{poly}(\bar{n})} \Pr\{f(M(g(\bar{x} \circ \tilde{x}))_{1..\bar{n}}) = f(\bar{x})\} \\ &\geqslant \frac{1}{\mathrm{poly}(\bar{n})} \Pr\{g(M(g(x))) = g(x)\} \\ &\geqslant \frac{1}{\mathrm{poly}(\bar{n})} \cdot \frac{1}{p(n)} = \frac{1}{\mathrm{poly}(\bar{n})},\end{aligned}$$

где вероятность берётся по $\bar{x}$, равномерно распределённым на $\{0,1\}^{\bar{n}}$, и по случайным числам, используемым противником N.

Утверждение о перестановках с секретом также легко проверяется, поскольку секрет для f годится и в качестве секрета для g (надо только соответствующим образом поменять длину входа).

Теорема 1. *Предположим, что сильно односторонние функции существуют. Пусть $r(n) = R \cdot n^{\rho}$ (где $R > 0$ и $\rho \geqslant 0$ – вычислимые константы), а r' – допустимая функция, такая, что $1 < \mathrm{const} \leqslant r'(n) \leqslant r(n)$. Пусть $p(n) = \frac{n \log n}{\log r'(n)}$. Пусть также $\zeta(n)$ – сколь угодно слабо растущая неубывающая допустимая функция.*

Тогда для всякого $k \geqslant 1$ и всякой допустимой функции $a(n) = O(\log n)$ существует сохраняющая длину функция G, вычислимая в классе $\mathbf{FTime}[n]/1$, *имеющая успешного $r'(n)$-противника в классе*

$$\mathbf{PFTime}[(p(n))^k \cdot r(p(n)) \cdot (\log n)^3 \cdot (\log r'(n))^{-2} \cdot \zeta(n) \cdot 2^{a(n)}] \ (\subseteq \mathbf{PFP}),$$

но не имеющая успешных $r(n)$-противников в классе $\mathbf{PFTime}[n^k]/a(n)$.

Замечание 3. Заметим, что теорема 1 позволяет довести вероятность успеха более сильного противника до произвольной константы (на самом деле даже до функции, стремящейся к единице, но для этого надо слегка поменять лемму 1). Однако это делается за счёт полилогарифмического увеличения времени работы; ниже мы приводим следствие, в котором рассматривается случай, когда вероятности успеха обоих противников совпадают (и тогда оценки времени их работы ближе). Заметим, что даже в этом простом случае “шаг” иерархии обратно пропорционален вероятности успеха противников.

Следствие 1. *Пусть сильно односторонние функции существуют. Пусть* $r(n) = R \cdot n^{\rho}$ *(где* $R > 0$ *и* $\rho \geqslant 0$ *– (вычислимые) константы), а* $\zeta(n)$ *– сколь угодно медленно растущая неубывающая допустимая функция. Тогда для всякого* $k \geqslant 1$ *имеется сохраняющая длину функция* G*, вычислимая в классе* $\mathbf{FTime}[n]/1$*, имеющая успешного* $r(n)$*-противника в классе* $\mathbf{PFTime}[r(n) \cdot n^k \log n \cdot \zeta(n)]$*, но не имеющая* $r(n)$*-противников в классе* $\mathbf{PFTime}[n^k]/1$.

Доказательство теоремы 1. По лемме 2 имеется сохраняющая длину сильно односторонняя функция f, вычислимая за линейное время. Определим новую сохраняющую длину функцию G, комбинирующую исходную f (используемую на некоторых длинах входов) с "раздутой" (т.е. ослабленной) f (используемой на других длинах входов). Неформально, мы используем то, что для некоторой степени раздутия функция остаётся надёжной относительно более слабых противников, но становится ненадёжной относительно более сильных.

При помощи функции $l(n)$ из леммы 1 мы определяем F как семейство функций $F_n : \{0,1\}^n \to \{0,1\}^n$ для всех длин входов n, где

$$F_n(x) = \begin{cases} f(x_{1..\pi_i}) \circ \ldots \circ f(x_{(n-\pi_i+1)..x_n}), & \text{если } \exists! i, s: \ n = l^s(\pi_i), \\ \text{не определено} & \text{в противном случае.} \end{cases}$$

Заметим, что $F_{l(n)}(x) = F_n(\ldots) \circ \ldots \circ F_n(\ldots)$ в случае, если F_n определена, т.е. $F_{l(n)}$ делит свой вход на $l(n)/n$ частей длины n и применяет F_n к каждой из них по отдельности.

Предположим, что существует противник M, который $r(l(n))$-взламывает функцию F на длине $l(n)$. Аналогично доказательству [6, 2.3.2], построим процедуру I, использующую противника M и обращающую F_n с некоторой существенной вероятностью.

Процедура I (вход: строка y длины n; оракул: M):

Для всех i от 1 до $l(n)/n$ повторить:

- Выбрать $u^{(1)}, \ldots, u^{(l(n)/n)}$ независимо согласно равномерному распределению на $\{0,1\}^n$.
- Вычислить $z^{(1)} \circ \ldots \circ z^{(l(n)/n)} = M(F_n(u^{(1)}) \circ \ldots \circ F_n(u^{(i-1)}) \circ y \circ F_n(u^{(i+1)}) \circ \ldots \circ F_n(u^{(l(n)/n)}))$.
- Если $F_n(z^{(i)}) = y$, то выдать $z^{(i)}$ и остановиться.

Оценим вероятность успеха процедуры I. Пусть S_n – множество всех строк x длины n, для которых I обращает функцию F_n (т.е. вычисляет элемент $F_n^{-1}(F_n(x))$) с вероятностью не менее $1/r_1(n)$, взятой по случайным числам, используемым I и M (мы выберем $r_1(n)$ позже):

$$S_n = \left\{ x \in \{0,1\}^n \;\middle|\; \Pr\left\{I(F_n(x)) \in F_n^{-1}(F_n(x))\right\} \geqslant \frac{1}{r_1(n)} \right\}.$$

Мы покажем, что мощность множества S_n достаточно велика, а именно,

$$\frac{|S_n|}{2^n} > \frac{1}{r_2(n)}, \tag{9}$$

где полином $r_2(n)$ также будет выбран позже.

Предположим, что, напротив, неравенство (9) не выполняется. Обозначим

$$s(n) = \Pr\left\{M\left(F_{l(n)}(u)\right) \in F_{l(n)}^{-1}(F_{l(n)}(u))\right\},$$

где вероятность берётся по u, равномерно распределённому на $\{0,1\}^{l(n)}$, и по случайным числам, используемым M. Поскольку M $r(l(n))$-взламывает $F_{l(n)}$, справедливо неравенство

$$s(n) \geqslant \frac{1}{r(l(n))}.$$

Представим строку $u \in \{0,1\}^{l(n)}$ в виде $u^{(1)} \circ \cdots \circ u^{(l(n)/n)}$, где каждая подстрока $u^{(i)}$ имеет длину n (ясно, что все подстроки $u^{(i)}$ независимы и равномерно распределены, если u равномерно распределена). Тогда величина $s(n)$ – сумма величин $s_1(n)$ и $s_2(n)$, определяемых как

$$s_1(n) = \Pr\left\{M\left(F_{l(n)}(u)\right) \in F_{l(n)}^{-1}(F_{l(n)}(u)) \;\wedge\; \exists i\; u^{(i)} \notin S_n\right\},$$

$$s_2(n) = \Pr\left\{M\left(F_{l(n)}(u)\right) \in F_{l(n)}^{-1}(F_{l(n)}(u)) \;\wedge\; \forall i\; u^{(i)} \in S_n\right\}.$$

Вспомним, что $F_{l(n)}(u) = F_n(u^{(1)}) \circ \ldots \circ F_n(u^{(l(n)/n)})$. Величина $s_1(n)$ оценивается как

$$\begin{aligned} s_1(n) &\leqslant \sum_{i=1}^{l(n)/n} \Pr\left\{M\left(F_{l(n)}(u)\right) \in F_{l(n)}^{-1}(F_{l(n)}(u)) \;\wedge\; u^{(i)} \notin S_n\right\} \\ &\leqslant \sum_{i=1}^{l(n)/n} \Pr\left\{I(F_n(u^{(i)})) \in F_n^{-1}(F_n(u^{(i)})) \;\wedge\; u^{(i)} \notin S_n\right\} \\ &< \frac{l(n)}{n \cdot r_1(n)}, \end{aligned}$$

а $s_2(n)$ – как

$$s_2(n) \leqslant \Pr\{\forall i\ u^{(i)} \in S_n\} \leqslant \left(\frac{1}{r_2(n)}\right)^{l(n)/n}.$$

Следовательно,

$$\frac{l(n)}{n \cdot r_1(n)} + \left(\frac{1}{r_2(n)}\right)^{l(n)/n} > s_1(n) + s_2(n) = s(n) \geqslant \frac{1}{r(l(n))}.$$

Положив

$$r_1(n) = 2 \cdot r(l(n)) \cdot \frac{l(n)}{n} \cdot \frac{1}{1-2\varepsilon},$$
$$r_2(n) = r'(n) \cdot \frac{1}{1+\varepsilon}$$

(для некоторой малой константы $\varepsilon > 0$), мы получаем противоречие с нашим предположением. Таким образом, "успешное множество" S_n имеет требуемую плотность $> \frac{1}{r_2(n)}$; остаётся увеличить вероятность успеха на S_n.

Увеличение вероятности успеха. Для того, чтобы увеличить вероятность успеха на множестве S_n, достаточно повторить процедуру I $C \cdot r_1(n)$ раз и, если какой-то из полученных ответов – правильный, вернуть его. Мы выберем константу C достаточно большой для того, чтобы выполнялось неравенство

$$1 - \left(1 - \frac{1}{r_1(n)}\right)^{C \cdot r_1(n)} > 1 - e^{-C} \geqslant \frac{1}{1+\varepsilon}$$

(выбор C позволяет сделать так, чтобы выполнилось последнее неравенство).

Таким образом, если процедуру I мы повторяем $C \cdot r_1(n)$ раз, полученный противник $A(M)$, работающий время

$$O\left(r_1(n) \cdot \frac{l(n)}{n} \cdot T_M(l(n)) \cdot \log T_M(l(n))\right)$$
$$= O\left(\frac{l(n)^2}{n^2} \cdot r(l(n)) \cdot T_M(l(n)) \cdot \log T_M(l(n))\right)$$

(вспомним из раздела 2, что T_M обозначает максимально допустимое время работы машины M; логарифмический сомножитель тратится на моделирование машины [10]), обращает F_n с вероятностью не менее

$$\frac{1}{r_2(n)} \cdot (1 - e^{-C}) \geqslant \frac{1}{r'(n)}.$$

Окончательная конструкция. Рассмотрим сколь угодно слабо растущую монотонно неубывающую функцию $\zeta(n)$. Пусть B – множество таких чисел n, что хотя бы одна из машин $M_1, \ldots, M_{\sqrt{\zeta(n)}}$ класса $\mathbf{PFTime}[n^k]/a(n)$ (вспомним, что в разделе 2 было зафиксировано эффективное перечисление машин) с некоторой подсказкой длины $a(n)$ $r(n)$-взламывает F на длине n. Пусть

$$E = \{n \in \mathbb{N} \mid l(n) \in B \wedge n \notin B \wedge \exists!\, l, s : n = l^s(\pi_i)\}.$$

Заметим, что множество E бесконечно. В самом деле, имеется бесконечно много чисел π_i, таких, что $\pi_i \notin B$ (в противном случае ясно, как построить $r(n)$-противника из $\mathbf{PFP}$ для исходной односторонней функции f: достаточно промоделировать машины $M_1, \ldots, M_{\sqrt{\zeta(n)}}$ со всеми возможными подсказками). С другой стороны, для каждого простого числа π_i имеется число s, такое, что $l^s(\pi_i) \in B$, поскольку функция

$$\underbrace{f \circ \ldots \circ f}_{l^k(\pi_i)/\pi_i}$$

может быть обращена детерминированным образом за время

$$l^k(\pi_i)/\pi_i \cdot 2^{\pi_i} \cdot \pi_i^{k_0}.$$

Определим нашу функцию G как семейство функций

$$G_n : \{0,1\}^n \to \{0,1\}^n,$$
$$G_n(x) = \begin{cases} F_n(x), & \text{если } n \in E, \\ f(x) & \text{в противном случае.} \end{cases} \tag{10}$$

Утверждается, что G удовлетворяет утверждению нашей теоремы.

1. Функция G принадлежит классу $\mathbf{FTime}[n]/1$.

Имея подсказку, сообщающую, верно ли, что $n \in E$, мы знаем,

какую из функций надо вычислять: $f \in \mathbf{FTime}[n]$ или F_n. Имея вход длины $n = l^s(\pi_i)$, по условию 5 леммы 1 мы можем вычислить π_i за время $O(n)$. Имея это число, вход можно разбить на части длины π_i, и f можно применить к каждой из частей за общее время $O(\frac{n}{\pi_i} \cdot \pi_i) = O(n)$.

2. В классе $\mathbf{PFTime}[n^k]/a(n)$ не существует успешного $r(n)$-противника для функции G.

Если M_i – успешный $r(n)$-противник класса $\mathbf{PFTime}[n^k]/a(n)$ для функции G, то он успешен на бесконечном подмножестве множества E (поскольку f – односторонняя), что противоречит определению множества E.

3. Для функции G существует $r'(n)$-противник класса

$$\mathbf{PFTime}[(p(n))^k \cdot r(p(n)) \cdot (\log n)^3 \cdot (\log r'(n))^{-2} \cdot \zeta(n) \cdot 2^{a(n)}],$$

являющийся успешным на бесконечном количестве длин входов, а именно на множестве E. (Ниже мы выберем конкретное l, и тогда будет ясно, что этой оценки на время работы достаточно.)

Перечислим машины $M = M_1, \ldots, M_{\sqrt{\zeta(n)}}$ (согласно перечислению из раздела 2) и все возможные подсказки α длины $a(n)$ для каждой из этих машин. Для каждого M и α вычислим $A(M/\alpha)(y)$, проверим ответ при помощи F_n и выдадим его, если он верен. Это можно сделать за время

$$O\left(\frac{l(n)^2}{n^2} \cdot r(l(n)) \cdot T_M(l(n)) \cdot \log T_M(l(n)) \cdot \sqrt{\zeta(n)} \cdot 2^{a(n)}\right)$$
$$= O\left(\frac{l(n)^{k+2}}{n^2} \cdot r(l(n)) \cdot \log l(n) \cdot \zeta(n) \cdot 2^{a(n)}\right)$$

(заметим, что $T_{M_i}(m) \leq \sqrt{\zeta(m)} \cdot m^k$ для $i \leq \sqrt{\zeta(m)}$). Остаётся подставить оценку $O\left(\frac{n \log n}{\log r'(n)}\right)$ для $l(n)$.

Замечание 4. Глядя на доказательство теоремы 1, легко заметить, что если существуют сохраняющие длину односторонние перестановки, то можно сразу воспользоваться функцией из леммы 3 (вместо леммы 2); тогда построенная нами функция G будет также сохраняющей длину перестановкой. И если исходная односторонная функция была функцией с секретом (trapdoor), то этот секрет подходит и для G.

4. Большее время и один бит подсказки позволяют вычислить больше языков класса $\mathbf{NTime}[n]/1$

Докажем результат, аналогичный теореме 1, для сложности в наихудшем случае.

Теорема 2. *Если* $\mathbf{P} \neq \mathbf{NP}$, *то для всяких* $l > k \geqslant 1$ *и всякой допустимой функции* $a(n) = O(\log n)$ *справедливо*

$$\Big(\mathbf{DTime}[n^l \cdot 2^{a(n)}] \cap \mathbf{NTime}[n]\Big) \Big/ 1 \ \not\subseteq \ \mathbf{DTime}[n^k]/a(n).$$

Следствие 2. *Если* $\mathbf{P} \neq \mathbf{NP}$, *то для всяких* $l > k \geqslant 1$ *справедливо* $(\mathbf{DTime}[n^k] \cap \mathbf{NTime}[n])/1 \subsetneq (\mathbf{DTime}[n^l] \cap \mathbf{NTime}[n])/1$.

Доказательство теоремы 2. Пусть $\varepsilon = l - k$. Не умаляя общности, $\varepsilon < 1$. Обозначим через SAT^* (**NP**-полный) вариант языка выполнимых булевых формул, состоящий из формул в конъюнктивной нормальной форме, дополненных $|F|^{2/\varepsilon} - |F|$ нулями, где $|F|$ – длина двоичного представления формулы F. Ясно, что SAT^* распознаётся за линейное время в любой разумной модели недетерминированных вычислений.

Рассмотрим язык A, определённый следующим образом:

$$x \in A \iff ((x = 1y \ \wedge \ y \in \mathrm{SAT}^*) \vee (x = 0y \ \wedge \ y \in A)).$$

Иными словами, суффиксы слов из $A \cap \{0,1\}^{n+1}$ содержат все выполнимые формулы длины $n^{\varepsilon/2}$ *или менее*, что означает, что если мы научимся решать все входы длины $n+1$, то мы научимся и выдавать выполняющие наборы (время работы домножится на $n^{\varepsilon/2}$) для выполнимых формул – это можно сделать, используя *самосводимость*: для формулы F и переменной v рассмотрим две формулы $F|_{v=0}$ и $F|_{v=1}$ и, если одна из них выполнима (т.е. после подходящего "раздутия" она принадлежит $A \cap \{0,1\}^{n+1}$), применим к ней эту процедуру рекурсивно; рекурсия продолжается, пока формула не станет тривиальной. Ясно также, что язык A – **NP**-полный.

Следующий язык B получается из языка A раздуванием (ср. [4]):

$$x \in B \iff \exists i \in \mathbb{N} \ (x = 0^{2^i} y \wedge y \in A \wedge 2^i > |y|).$$

Ясно, что A сводится к B; следовательно, B – **NP**-полный. Также ясно, что все слова языка B одной и той же длины соответствуют

словам языка A этой же длины. Пусть $\mathrm{pad}(x)$ обозначает следующую по длине строку, соответствующую той же самой строке y, т.е. $\mathrm{pad}(x) = 0^{2^{i+1}}y$ тогда и только тогда, когда $x = 0^{2^i}y$ и $|y| < 2^i$. Обозначим также $\mathrm{first}(m) =$ наименьшая длина x, соответствующая какому-либо y длины m, и пусть $\mathrm{next}(n) = |\mathrm{pad}(x)|$ для любого x длины n. Пусть $\mathrm{next}^i(n) = \underbrace{\mathrm{next}(\ldots\mathrm{next}}_{i}(n))$.

Напомним, что в разделе 2 мы зафиксировали эффективное перечисление машин M_i класса $\mathbf{DTime}[n^k]/a(n)$ (но не их подсказок). Пусть $\zeta(n)$ – сколь угодно слабо растущая монотонно неубывающая допустимая функция.

Утверждение. Если машина M_i класса $\mathbf{DTime}[n^k]/a(n)$ с подсказкой α распознаёт B правильно для всех входов длины $\mathrm{next}(n)$ (где $\zeta(\mathrm{next}(n)) \geqslant i$), то следующий алгоритм OPT, работающий время $O(n^{k+\varepsilon/2}\log n \cdot \zeta(n) \cdot 2^{a(n)})$, распознаёт B правильно для всех входов длины n (сомножитель $n^{\varepsilon/2}$ добавляется во время работы за счёт процедуры самосводимости, а сомножитель $\log n$ – за счёт моделирования [10]).

Алгоритм OPT (вход: строка x длины n).

Для всех машин $M = M_1, M_2, \ldots, M_{\sqrt{\zeta(n)}}$ и подсказок a длины $a(n)$ повторить:

- Промоделировать $M(\mathrm{pad}(x))$ в течение $T_M(\mathrm{next}(n))$ шагов.
- Если ответ положительный, проверить его при помощи самосводимости.
- Если ответ верный, ответить “Да” и остановиться.

Ответить “Нет”.

Выберем эти длины входов, чтобы включить их в язык C, который демонстрирует нашу иерархию:

$x \in C \iff x \in B \wedge$ OPT решает B правильно на всех словах длины $|x|$.

Ясно, что

$$C \in (\mathbf{NTime}[n] \cap \mathbf{DTime}[n^{k+\varepsilon/2} \cdot \log n \cdot \zeta(n) \cdot 2^{a(n)}])/1$$
$$\subseteq (\mathbf{NTime}[n] \cap \mathbf{DTime}[n^{k+\varepsilon} \cdot 2^{a(n)}])/1,$$

где бит подсказки говорит, верно ли, что OPT в самом деле решает B правильно на всех словах соответствующей длины. Также ясно, что для всякого m язык $C \cap (\{0,1\}^{\mathrm{first}(m)} \cup \{0,1\}^{\mathrm{next}(\mathrm{first}(m))} \cup \dots)$ бесконечен, поскольку почти все его слова очень "просты" (так как "раздуты"). Остаётся показать, что $C \notin \mathbf{DTime}[n^k]/a(n)$.

Предположим, что имеется машина M класса $\mathbf{DTime}[n^k]/a(n)$ с последовательностью подсказок $\mathfrak{A} = (\alpha_1, \alpha_2, \dots)$, которая распознаёт язык C. Для каждого m рассмотрим первое i, такое, что множество $C \cap \{0,1\}^{\mathrm{next}^i(\mathrm{first}(m))}$ непусто. Поскольку $M/\mathfrak{A}$ распознаёт C и мы выбрали первое такое i, из приведённого выше утверждения следует, что $i = 0$. Поскольку это верно для каждого m, получается, что $M/\mathfrak{A}$, и, следовательно, сам алгоритм OPT также решает **NP**-полную задачу B, что противоречит предположению $\mathbf{P} \neq \mathbf{NP}$.

Замечание 5. Заметим, что сомножитель $n^{\varepsilon/2} = n^{(l-k)/2}$ во времени работы более сильной машины отражает время работы процедуры самосводимости. Если сформулировать аналогичную теорему в терминах функций, односторонних в наихудшем случае, этого сомножителя не будет, поскольку выход противника, обращающего функцию f, можно проверить, применив к нему саму f; таким образом, время работы успешного противника будет составлять $O(n^k \log n \cdot \zeta(n) \cdot 2^{a(n)})$.

Литература

1. S. Aaronson, *Complexity Zoo*. — http://www.complexityzoo.com.
2. B. Barak, *A probabilistic time hierarchy theorem for slightly nonuniform algorithms*. — In: *Proceedings of the 6th International Workshop on Randomization and Approximation Techniques in Computer Science*, volume 2483 of Lecture Notes in Computer Science, pages 194–208. Springer-Verlag (2002).
3. S. Cook, *A hierarchy theorem for nondeterministic time complexity*. — Journal of Computer and System Sciences **7** (1973), 343–353.
4. L. Fortnow and R. Santhanam, *Hierarchy theorems for probabilistic polynomial time*. — In: *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science* (2004), pages 316–324, IEEE.
5. L. Fortnow, R. Santhanam, and L. Trevisan, *Hierarchies for semantic classes*. — In: *Proceedings of the 37th ACM Symposium on the Theory of Computing* ACM (2005), 348–355.
6. O. Goldreich, *Foundations of Cryptography*: Volume 1, Basic Tools. *Cambridge University Press* (2001).

7. O. Goldreich, M. Sudan, and L. Trevisan, *From logarithmic advice to single-bit advice.* — Technical Report TR-04-093, Electronic Colloquium on Computational Complexity (2004).
8. L. A. Hemaspaandra and M. Ogihara, *The Complexity Theory Companion.* Springer (2002).
9. J. Hartmanis and R. Stearns, *On the computational complexity of algorithms.* — Transactions of the American Mathematical Society **117** (1965), 285–306.
10. F. Hennie and R. Stearns, *Two-tape simulation of multitape Turing machines.* Journal of ACM **13 (4)** (1966) 533–546.
11. L. Levin, *Universal search problems.* — Problems of Information Transmission **9(3)** (1973), 265–266. In Russian.
12. D. van Melkebeek and K. Pervyshev, *A generic time hierarchy with one bit of advice.* — Computational Complexity **16(2)** (2007), 139–179.
13. C. Papadimitriou, *Computational Complexity.* — Addison-Wesley (1991).
14. K. Pervyshev, *Time hierarchies for computations with a bit of advice.* — Technical Report TR-05-054, Electronic Colloquium on Computational Complexity (2005).
15. J. Seiferas, M. Fischer, and A. Meyer, *Separating nondeterministic time complexity classes.* — Journal of the ACM **25** (1978).
16. S. Žák, *A Turing machine time hierarchy.* — Theoretical Computer Science (1983).

Hirsch E. A., Grigoriev D. Yu., Pervyshev K. V. Time hierarchies for cryptographic function inversion with advice.

We prove a time hierarchy theorem for inverting functions computable in a slightly non-uniform polynomial time. In particular, we prove that if there is a strongly one-way function then for any k and for any polynomial p, there is a function f computable in linear time with one bit of advice such that there is a polynomial-time probabilistic adversary that inverts f with probability $\geqslant 1/p(n)$ on infinitely many lengths of input while all probabilistic $O(n^k)$-time adversaries with logarithmic advice invert f with probability less than $1/p(n)$ on almost all lengths of input.

We also prove a similar theorem in the worst-case setting, i.e., if $\mathbf{P} \neq \mathbf{NP}$, then for every $l > k \geqslant 1$

$$(\mathbf{DTime}n^k \cap \mathbf{NTime}n)/1 \subsetneq (\mathbf{DTime}n^l \cap \mathbf{NTime}n)/1.$$

С.-Петербургское отделение
Математического института
им. В. А. Стеклова РАН
Web: http://logic.pdmi.ras.ru/~hirsch/

IRMAR, Université de Rennes,
Web: http://perso.univ-rennes1.fr/dmitry.grigoryev

Санкт-Петербургский государственный университет и
University of California, San Diego,
Department of Computer Science and Engineering

Поступило 11 мая 2007 г.