Electronic Colloquium on Computational Complexity, Report No. 76 (2005)

Time hierarchies for cryptographic function inversion with advice

Dima Grigoriev[*] Edward A. Hirsch[†] Konstantin Pervyshev[‡]

July 2, 2005

Abstract

We prove a time hierarchy theorem for inverting functions computable in polynomial time with one bit of advice. 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 $\geq 1/p(n)$ 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 $l > k \geq 1$

$$(\mathbf{DTime}[n^k] \cap \mathbf{NTime}[n])/1 \underset{\neq}{\subset} (\mathbf{DTime}[n^l] \cap \mathbf{NTime}[n])/1.$$

1 Introduction

Motivation. One of the most challenging problems in cryptography is the complexity of inverting functions: public-key cryptography could exist only if there is a function that is easy to compute but hard to invert. At a first glance, it seems natural that adversaries that are allowed more time can invert more functions. This is a classic problem of *time hierarchies.*

For simplicity, consider first the worst-case (rather than more complicated cryptographic) setting and a natural assumption that $\mathbf{P} \neq \mathbf{NP}$. Then the hierarchy problem can be formulated as

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

(where n^w is the complexity of computing the functions we are trying to invert, and $l > k$), because $\mathbf{P} \neq \mathbf{NP}$ implies the existence of worst-case one-way functions based on $\mathbf{NP}$-complete problems, and inverting such a function is equivalent to solving the corresponding decision problem.

However, the hierarchy (1) is not easy to prove since the corresponding class $\mathbf{DTime}[n^k] \cap \mathbf{NTime}[n^w]$ is not a *syntactic* class, i.e., given a deterministic and a non-deterministic machine it is impossible to check that they accept the same language. There was a recent progress in proving

[*]IRMAR, Université de Rennes, Campus de Beaulieu, 35042 Rennes, cedex France. Web: `http://name.math.univ-rennes1.fr/dimitri.grigoriev/` .

[†]St.Petersburg Department of Steklov Institute of Mathematics, 27 Fontanka, 191011 St.Petersburg, Russia. Web: `http://logic.pdmi.ras.ru/~hirsch/` . Supported in part by INTAS project 04-77-7173 and Russian Science Support Foundation.

[‡]Mathematics and Mechanics Dept., St.Petersburg State University, St.Petersburg, Russia. Web: `http://logic.pdmi.ras.ru/~pervyshev/` .

ISSN 1433-8092

time hierarchies for such *semantic* (or *promise*) classes. These recent advances motivated this work and allowed to prove hierarchy theorems in the worst-case as well as in the cryptographic setting (see below for details). We give a brief survey of the history of time hierarchies for syntactic and semantic complexity classes and review our techniques in the next paragraphs.

History of time hierarchies. In the 1960s Hartmanis and Stearns [HS65] showed that for any constants k and l such that $1 < k < l$, $\mathbf{DTime}[n^k] \subsetneq \mathbf{DTime}[n^l]$ where $\mathbf{DTime}[n^d]$ is the class of the languages decidable by multi-tape deterministic Turing machines operating within $O(n^d)$ steps. A decade later, Cook [Coo73] proved a time hierarchy for nondeterministic computations: $\mathbf{NTime}[n^k] \subsetneq \mathbf{NTime}[n^l]$. Both these results can be proved using the *diagonalization*, a technique proposed by Cantor (cf. [For00]).

However, the diagonalization fails to prove hierarchy theorems for probabilistic classes, in particular, for **BPTime** (the languages recognized by randomized algorithms with two-sided bounded error), **RPTime** (the languages recognized by randomized algorithms with one-sided bounded error), and **ZPTime** (the languages recognized by randomized algorithms with correct answers only). The main obstacle is that these classes impose some restrictions on the error probability of the machines, and the set of machines satisfying these restrictions is not known to be recursively enumerable. Therefore, a straightforward diagonalization argument may produce a machine that does not define such language.

The lack of different techniques resulted in many long-standing open questions in the area including $\mathbf{BPTime}[n]$ vs $\mathbf{BPP} = \bigcup_d \mathbf{BPTime}[n^d]$ and similar questions for **RPTime** and **ZPTime**.

In 2002 Barak [Bar02] suggested a new technique for proving time hierarchies that uses an *optimal algorithm* introduced by Levin [Lev73]. This technique was developed later by Fortnow and Santhanam [FS04] and Goldreich, Sudan and Trevisan [GST04] who showed that there exists a polynomial time hierarchy for **BPTime** with one-bit advice, that is for the class of the languages recognizable by probabilistic algorithms with two-sided error that use one bit of advice for every input length (only algorithm equipped with *right* advice needs to satisfy the requirements of the class). Formally, it was proved that $\mathbf{BPTime}[n^k]/1 \subsetneq \mathbf{BPTime}[n^l]/1$. The same result holds for **RPTime** [FST05] as well.

In 2005 Fortnow, Santhanam and Trevisan [FST05] came up with an idea that it is possible to prove a time hierarchy with short advice for a wide range of computations at once. Essentially, their technique gives a hierarchy for any reasonable complexity class defined using nondeterministic (including probabilistic) polynomial-time machines with advice function $a(n) = O(\log n \cdot \log\log n)$, for example, for $\mathbf{ZPTime}/a(n)$, $\mathbf{RPTime}/a(n)$, $(\mathbf{NTime} \cap \mathbf{coNTime})/a(n)$, $\mathbf{UTime}/a(n)$ (nondeterministic machines with unambiguous accepting path), $\mathbf{MATime}/a(n)$ (Merlin-Arthur games with time-bounded Arthur), $\mathbf{AMTime}/a(n)$ (Arthur-Merlin games with time-bounded Arthur) (see [Aar] for formal definitions).

Recently, van Melkebeek and Pervyshev [vMP05, Per05] proved that a polynomial time hierarchy with *one* bit of advice exists for the same broad range of computations. For example, it was proved that

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

Our results. In this paper we prove two hierarchy theorems for inverting one-way functions with one bit of advice. Namely, we prove a version of (1) in the worst-case setting: for every positive constant $w \geq 1$ and for all $l > k \geq w$,

$$\mathbf{DTime}[n^k] \cap \mathbf{NTime}[n^w] \subsetneq \mathbf{DTime}[n^l] \cap \mathbf{NTime}[n^w],$$

and we prove a similar theorem in the cryptographic setting. The technique in the worst-case setting is similar to the one used in [Per05] (however, our result is not implied by the general result of [Per05, vMP05]); the cryptographic setting requires several more tricks, which are informally discussed below.

A strongly one-way function is a function that is easy to compute but hard to invert. More specifically, (1) it is computable in polynomial time, and (2) no polynomial-time randomized algorithm inverts it on infinitely many input lengths with significant probability. This probability of success depends on the input length n. *Significant probability* is the one that is greater than $1/r(n)$ for some polynomial $r(n)$.

We say that a (polynomial-time randomized) adversary M $r(n)$*-breaks* a function f iff M inverts f with probability at least $1/r(n)$ on infinitely many input lengths. Evidently, if there is a polynomial r and an adversary that $r(n)$-breaks f, then f is not a strongly one-way function. A natural question arises when we consider the exact time that adversaries use for breaking a function: Is it possible to break more functions using more time?

Under the assumption that strongly one-way functions exist, we answer this question affirmatively by proving, in particular, that for any polynomial r and every $k > 1$ there is a length-preserving function G computable in linear time with one bit of advice that has a polynomial-time randomized adversary that $r(n)$-breaks it, but has no such $O(n^k)$-time adversaries (even with one bit of advice). (See Theorem 1 for the exact time bound of the successful adversary and for a stronger formulation that allows this adversary to break the function with a higher probability than the weaker adversaries.) In other words, we prove a *time hierarchy theorem* for adversaries to strongly one-way functions with one bit of advice. Note that the successful adversary itself does not use any advice.

Our technique. Starting with any strongly one-way function f, we construct a function F that is easier to invert than f at some input lengths. The difficulty of inverting F gradually decreases on the (infinite) sequence of lengths π, $l(\pi)$, $l(l(\pi))$, ... for a prime π and an appropriate padding function l; the function F equals f on inputs of length π and is easily invertible on higher input lengths in this sequence. Now consider any adversary M that $r(n)$-breaks the function F in time $O(n^k)$. It appears that for some number $n = l(l(\ldots(\pi)\ldots))$, M breaks the function F with probability $1/r(l(n))$ on length $l(n)$ but not on length n with probability $1/r(n)$. Our construction of F allows to compute (and invert) F at length n using queries to F (or to an adversary) at length $l(n)$. We can thus construct another adversary N that would invoke M on inputs of length $l(n)$ to invert function F at length n.

However, the probability of breaking function F at length n by our adversary is still $1/r(l(n))$, which is less than the desired probability $1/r(n)$. To overcome this difficulty we employ the hardness amplification technique (cf. [Gol01]). Going into more details, our function F is obtained by concatenating several instances of the original function f for a smaller length of input. Now invoking the adversary M several times on inputs of length $l(n)$ by substituting our query for different instances of f increases the probability of success to $1/r(n)$ (and even higher, if needed). Of course, the price for it is an increase in the running time.

It is straightforward to modify F in such a way that N $r(n)$-breaks it and M does not (just use one bit of advice to wipe out "non-interesting" inputs lengths from our sequence). However, our goal is that *any* $O(n^k)$-time adversary M_i fails to $r(n)$-break F. To achieve it, our adversary N invokes all the $O(n^k)$-adversaries M_1, M_2 and so on in the way that is very similar to the construction of Levin's optimal algorithm [Lev73].

The theorem in the worst-case setting is proved a bit differently (and much simpler).

Paper organization. In the subsequent sections we give definitions (Section 2), prove our main theorem in the cryptographic setting (Section 3) and prove its worst-case analogue (Section 4). Section 5 lists remaining open questions.

2 Definitions

Definition 1. A function is in $\mathbf{FTime}[t(n)]$ if it can be computed within time $O(t(n))$ on a multi-tape deterministic Turing machine. Also define $\mathbf{FP} = \bigcup_k \mathbf{FTime}[n^k]$. Notations $\mathbf{DTime}[t(n)]$ and $\mathbf{P}$ stand for predicates (i.e., languages) in $\mathbf{FTime}[t(n)]$ and $\mathbf{FP}$; $\mathbf{NTime}[t(n)]$ and $\mathbf{NP}$ stand for predicates computed by appropriate multi-tape non-deterministic Turing machines.

Let us denote the class of all multi-tape probabilistic Turing machines computing functions within (worst-case) time $O(t(n))$ by $\mathbf{PFTime}[t(n)]$. Note that $\mathbf{PFTime}[\cdot]$ classes are classes of machines and not of functions, and these machines may return different results depending on their coin tosses (later we will be interested in the probability that such a machine outputs the value of some particular function of its input). We also assume that all our polynomial-time machines are supplied with a specific time bound of the form $C \cdot n^k$ and an alarm clock, i.e., they actually terminate within the declared time bound. Finally, define $\mathbf{PFP} = \bigcup_k \mathbf{PFTime}[n^k]$.

For a complexity class $\mathcal{C}$ (of languages, functions, or machines), we write $\mathcal{C}/a(n)$ if Turing machines employed in the definition of $\mathcal{C}$ are supplied with length $a(n)$ advice on one of their tapes.

For each of the above classes $\mathbf{DTime}[n^k]$, $\mathbf{DTime}[n^k]/a(n)$, $\mathbf{PFTime}[n^k]$, and $\mathbf{PFTime}[n^k]/a(n)$, we fix an effective (and easily computable) enumeration $M_1, M_2, \ldots$ of the machines (but not of their advice strings) such that M_i's alarm clock interrupts it after $T_{M_i}(n) = C_{M_i} \cdot n^k$ steps for $0 < C_{M_i} \le i$.

Definition 2. $f : \{0,1\}^* \to \{0,1\}^*$ is a strongly one-way function secure against a uniform probabilistic adversary (or just *strongly one-way function*) if $f \in \mathbf{FP}$, f is honest (i.e., for every $y \in \operatorname{Im} f$ there is a string in $f^{-1}(y)$ of length bounded by a polynomial of $|y|$), and for every polynomial p and every $A \in \mathbf{PFP}$, $\exists N \; \forall n > N$

$$\Pr\{A(f(x)) \in f^{-1}(f(x))\} < \frac{1}{p(n)},$$

where the probability is taken over x uniformly distributed on $\{0,1\}^n$ and over the internal coin tosses of A.

Definition 3. $A \in \mathbf{PFP}$ is a *successful* $r(n)$*-adversary* for $f : \{0,1\}^* \to \{0,1\}^*$ if it serves as a counterexample in Definition 2, i.e., for infinitely many n,

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

When (2) holds for a particular length n, we say that A $r(n)$*-breaks* f at this length.

Remark 1. *An honest* $\mathbf{FP}$ *function is strongly one-way iff it has no successful* $p(n)$*-adversaries for any polynomial* p.

Definition 4. We define strongly one-way functions and adversaries with advice similarly.

Definition 5. A function T from non-negative integers to non-negative integers is called *proper* (cf. [Pap91, Section 7.1]; note that we need neither monotonicity nor space restrictions of [Pap91]) if the string $0^{T(n)}$ (i.e., $T(n)$ in unary) is computable in time $O(n + T(n))$ when given any string of length n as its input.

3 More time means inverting more functions with greater probability

We first need a technical lemma that gives a construction of a padding function with certain properties.

Lemma 1. Let $\{\pi_1, \pi_2, \ldots\}$ be the set of all odd primes. Let $r(n) = R \cdot n^\rho$ (where $R > 0$ and $\rho \geq 0$ are (computable) constants) and r' be a proper function such that $1 < b \leq r'(n) \leq r(n)$ (where b is a constant). Then there exists a function $l : \mathbb{N} \to \mathbb{N}$ such that

1. $\exists n_0 \forall n \geq n_0\ (r'(n))^{l(n)/n} > 2 \cdot r(l(n))$.
2. $\forall n\ n|l(n)$.
3. Let $l^s(n)$ denote $\underbrace{l(\ldots(l(}_{s}n))\ldots)$. Then $\forall s, s', j, j'\ ((s,j) \neq (s',j') \Rightarrow l^s(\pi_j) \neq l^{s'}(\pi_{j'}))$.
4. l is proper.
5. Given $l^s(\pi_j)$ in unary, the number π_j can be computed in time $O(l^s(\pi_j))$.
6. $l(n) = O\left(\frac{n \log n}{\log r'(n)}\right)$.

Proof. First of all, if $\rho = 0$, then an appropriate linear function $l(n) = n \cdot \text{const}$ satisfies all these requirements. Thus we assume that r is non-constant.

We first construct a function $m(n)$ that satisfies the requirement 1 and then replace it with a slightly larger $l(n)$ that satisfies other requirements. Taking the binary logarithm of the inequality in requirement 1, we transform it into

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

i.e.,

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

Let

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

It is easy to see that $\mu(n) = \Theta(\frac{n}{\log r'(n)})$ and $m(n) = \Theta(\frac{n \log n}{\log r'(n)})$. Also,

$$\mu(n) \geq \left(1 + \frac{1}{\log b}\right) \frac{n\rho}{\log r'(n) + 1} \cdot \frac{\log r'(n)}{\log r'(n)} = \frac{1 + \frac{1}{\log b}}{1 + \frac{1}{\log r'(n)}} \cdot \frac{n\rho}{\log r'(n)} \geq \frac{n\rho}{\log r'(n)}$$

and

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

for sufficiently large numbers n. Therefore inequality (4) holds for all sufficiently large numbers n.

Finally, let $l(n) = n \cdot t^2$, where $(t-1)^2 \leq \max\{\frac{m(n)}{n}, 1\} < t^2$. It is easy to see that it still satisfies requirement 1. Requirement 2 is trivially satisfied; requirement 3 holds, because π_j and $\pi_{j'}$ are the only prime factors of $l^s(\pi_j)$ and $l^{s'}(\pi_{j'})$ of odd multiplicity; it is also clear that l is proper (requirement 4). To verify 5, note that to figure out π_j it is enough to try at most $\sqrt{l^s(\pi_j)}$ possible factors and perform division by each of them at most logarithmic number of times to check the multiplicity. Finally, it is easy to see that

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

Indeed, if $\frac{m(n)}{n} \geq 16$, we have $t > 4$ and $2(t-1)^2 > t^2$. Therefore, $l(n) = n \cdot t^2 < 2n \cdot (t-1)^2 \leq 2n \cdot \frac{m(n)}{n} = O(m(n))$. Otherwise, $\frac{m(n)}{n} < 16$. Thus, we have $t^2 \leq 16$ and $l(n) = O(n)$. Recall that $m(n) = \Theta(\frac{n \log n}{\log r'(n)})$ and $\log r'(n) = O(\log n)$. Hence $m(n) = \Omega(n)$ and $l(n) = O(m(n))$.

□

The following lemma states that if strongly one-way functions exist, then there is a strongly one-way length-preserving function computable in *linear* time. We also formulate a version of it where the original function is a length-preserving permutation; then the newly constructed linear-time function inherits this property. To make our paper self-contained, we prove these lemmas in the appendix (they are based on folklore results; however, we do not know whether they were ever published together with the linear-time requirement).

Lemma 2. Assume there is a strongly one-way function. Then there exists a length-preserving strongly one-way function computable in linear time.

Lemma 3. Assume there is a strongly one-way permutation. Then there exists a strongly one-way permutation computable in linear time. Moreover, there is a family of trapdoor permutations, the functions constructed by this lemma form a family of trapdoor permutations as well.

Theorem 1. Assume that strongly one-way functions exist. Let $r(n) = R \cdot n^\rho$ (where $R > 0$ and $\rho \geq 0$ are (computable) constants) and r' be a proper function such that $1 < \text{const} \leq r'(n) \leq r(n)$. Let $p(n) = \frac{n \log n}{\log r'(n)}$. Consider any proper slowly growing non-decreasing function $\zeta(n)$.

Then for every $k \geq 1$ and every proper function $a(n) = O(\log n)$ there is a length-preserving function G computable in $\mathbf{FTime}[n]/1$ that has a successful $r'(n)$-adversary in

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

but no successful $r(n)$-adversary in $\mathbf{PFTime}[n^k]/a(n)$.

Remark 2. *Note that Theorem 1 allows to make the probability of success of the more powerful adversary an arbitrary constant (in fact, it is possible even to replace it by a function tending to one, which requires a slight modification of Lemma 1). However, the price for it is a polylogarithmic increase in running time; the corollary below considers the case when the probabilities of success of both adversaries are equal (and thus the time bounds for the two adversaries are closer). Note that even in this simple case the gap in the hierarchy is proportional to the inverse of success probability of adversaries.*

Corollary 1. Assume strongly one-way functions exist. Let $r(n) = R \cdot n^\rho$ (where $R > 0$ and $\rho \geq 0$ are (computable) constants). Consider any proper slowly growing non-decreasing function $\zeta(n)$. Then for every $k \geq 1$ there is a length-preserving function G computable in $\mathbf{FTime}[n]/1$ that has a successful $r(n)$-adversary in $\mathbf{PFTime}[r(n) \cdot n^k \log n \cdot \zeta(n)]$ but no successful $r(n)$-adversary in $\mathbf{PFTime}[n^k]/1$.

Proof of Theorem 1. By Lemma 2 there is a length-preserving linear-time computable strongly one-way function f. We now define a new length-preserving function G that combines the original f (employed at certain input lengths) with "padded" (i.e., weakened) f (employed at other input lengths). The informal idea behind it is that for a certain amount of padding the function remains secure against weaker adversaries while becomes insecure against slightly stronger ones.

Using $l(n)$ given by Lemma 1, we define F as the collection of functions $F_n : \{0,1\}^n \to \{0,1\}^n$ for all input lengths n, where

$$F_n(x) = \begin{cases} f(x_{1..\pi_i}) \circ \ldots \circ f(x_{(n-\pi_i+1)..x_n}), & \text{if } \exists! i, s :\ n = l^s(\pi_i), \\ \text{undefined}, & \text{otherwise.} \end{cases}$$

Note that $F_{l(n)}(x) = F_n(\ldots) \circ \ldots \circ F_n(\ldots)$ if F_n is defined, i.e., $F_{l(n)}$ splits its input into $l(n)/n$ parts of length n and applies F_n to each of them separately.

Assume there is an adversary M that $r(l(n))$-breaks the function F at length $l(n)$. Following, e.g., [Gol01, Theorem 2.3.2], we construct a procedure I that uses the adversary M and inverts F_n with some substantial probability.

```
Procedure I (accepts input y of length n; uses M as an oracle):
  • For $i := 1$ to $l(n)/n$ do
      – Pick $u^{(1)}, \ldots, u^{(l(n)/n)}$ uniformly and independently from $\{0,1\}^n$.
      – Compute $z^{(1)} \circ \ldots \circ z^{(l(n)/n)} = M(F_n(u^{(1)}) \circ \ldots \circ F_n(u^{(i-1)}) \circ y \circ F_n(u^{(i+1)}) \circ \ldots \circ F_n(u^{(l(n)/n)}))$.
      – If $F_n(z^{(i)}) = y$, then output $z^{(i)}$ and stop.
```

We now estimate the probability of success of I. Let S_n be the set of all strings x of length n for which I inverts the function F_n (i.e., computes an element of $F_n^{-1}(F_n(x))$) with probability at least $1/r_1(n)$ over internal coin tosses of I and M (we will choose $r_1(n)$ later):

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

We will show that $|S_n|$ is big enough, namely,

$$\frac{|S_n|}{2^n} > \frac{1}{r_2(n)}, \tag{5}$$

where $r_2(n)$ will be also chosen later.

Assume to the contrary that (5) does not hold. Let

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

where the probability is taken over u uniformly distributed on $\{0,1\}^{l(n)}$ and over internal coin tosses of M. Since M $r(l(n))$-breaks $F_{l(n)}$, we have

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

Let us split $u \in \{0,1\}^{l(n)}$ into $u^{(1)} \circ \cdots \circ u^{(l(n)/n)}$, each $u^{(i)}$ being of length n (clearly, all $u^{(i)}$'s are independent and uniformly distributed whenever u is). Then $s(n)$ is the sum of $s_1(n)$ and $s_2(n)$

defined by

$$s_1(n) = \Pr\left\{M(F_{l(n)}(u)) \in F_{l(n)}^{-1}(F_{l(n)}(u)) \ \wedge\ \exists i\ u^{(i)} \notin S_n\right\},$$
$$s_2(n) = \Pr\left\{M(F_{l(n)}(u)) \in F_{l(n)}^{-1}(F_{l(n)}(u)) \ \wedge\ \forall i\ u^{(i)} \in S_n\right\}.$$

Recall that $F_{l(n)}(u) = F_n(u^{(1)}) \circ \ldots \circ F_n(u^{(l(n)/n)})$. We can estimate $s_1(n)$ as

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

and $s_2(n)$ as

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

Hence

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

Letting

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

(for a small constant $\varepsilon > 0$) we come to a contradiction with our assumption. We thus have a "success set" S_n of the desired density $> \frac{1}{r_2(n)}$; it remains to amplify the success rate on S_n by repeating I $C \cdot r_1(n)$ times for a sufficiently large constant C:

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

(the last inequality is by the choice of C).

Thus if we ask I to simulate M itself and repeat I for $C \cdot r_1(n)$ times, we get an adversary $A(M)$ that runs in time $O(r_1(n) \cdot \frac{l(n)}{n} \cdot (T_M(l(n)) \cdot \log T_M(l(n))) = O(\frac{l(n)^2}{n^2} \cdot r(l(n)) \cdot (T_M(l(n)) \cdot \log T_M(l(n)))$ (recall from Sect. 2 that T_M denotes M's time bound; the logarithmic overhead is spent for simulation [HS66]) and inverts F_n with probability at least

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

Consider any proper slowly growing non-decreasing function $\zeta(n)$. Let B be the set of such numbers n that at least one of the $\mathbf{PFTime}[n^k]/a(n)$ machines $M_1, \ldots, M_{\sqrt{\zeta(n)}}$ (recall our effective enumeration of machines in Sect. 2) with some $a(n)$-size advice $r(n)$-breaks F at length n. Let

$E = \{n \in \mathbb{N} \mid l(n) \in B \wedge n \notin B \wedge \exists!\, l, s : n = l^s(\pi_i)\}$. We now define our function G as a collection of functions $G_n : \{0,1\}^n \to \{0,1\}^n$:

$$G_n(x) = \begin{cases} F_n(x), & \text{if } n \in E \\ f(x), & \text{otherwise} \end{cases}. \tag{6}$$

We claim that G satisfies the statement of the theorem:

1. The function G is computable in $\mathbf{FTime}[n]/1$.

   Given an advice for the set E, we know whether to compute $f \in \mathbf{FTime}[n]$ or F_n. Given an input of length $n = l^s(\pi_i)$, the number π_i can be computed in time $O(n)$ due to the condition 5 of Lemma 1. Given this number, the input can be easily split into pieces of size π_i each and f can be applied to each of them for a time cost of $O(\frac{n}{\pi_i} \cdot \pi_i) = O(n)$ in total.

2. The function G does not have a successful $r(n)$-adversary in $\mathbf{PFTime}[n^k]/a(n)$.

   If M_i is a successful $r(n)$-adversary in $\mathbf{PFTime}[n^k]/a(n)$ for G, then it is a successful adversary on an infinite subset of E (because f is one-way), which contradicts the definition of E.

3. The function G has a successful $r'(n)$-adversary in

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

   on an infinite set of input lengths, namely, on E. (When we choose specific l below, it will be clear that this time bound suffices.)

   Enumerate the machines $M = M_1, \dots, M_{\sqrt{\zeta(n)}}$ (recall the enumeration from Sect. 2) and all their possible $a(n)$-size advices α. For each M and α, compute $A(M/\alpha)(y)$, verify its answer by computing F_n, and output this answer if it is correct. It can be done within

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

   time (note that $T_{M_i}(m) \leq \sqrt{\zeta(m)} \cdot m^k$ for $i \leq \sqrt{\zeta(m)}$). It remains to substitute the bound $O\left(\frac{n \log n}{\log r'(n)}\right)$ for $l(n)$.

   To verify that E is indeed infinite, note that there are infinitely many numbers π_i such that $\pi_i \notin B$ (otherwise it is straightforward to construct a **PFP** $r(n)$-adversary for the original one-way function f: just simulate the machines $M_1, \dots, M_{\sqrt{\zeta(n)}}$ with all possible advices). On the other hand, for every prime π_i there is a number s such that $l^s(\pi_i) \in B$ (since the function $\underbrace{f \circ \dots \circ f}_{l^k(\pi_i)/\pi_i}$ can be inverted deterministically in time $l^k(\pi_i)/\pi_i \cdot 2^{\pi_i} \cdot \pi_i^{k_0}$).

□

Remark 3. *It is easy to see from the proof of Theorem 1 that if length-preserving permutations exist, then one can start directly with a function provided by Lemma 3 (instead of Lemma 2); then the newly constructed function G becomes a length-preserving permutation as well. Also if the original one-way function has a trapdoor, G inherits it.*

4 More time and one bit of advice means computing more functions inside $\mathbf{NTime}[n]/1$

Theorem 2. If $\mathbf{P} \neq \mathbf{NP}$, then for every $l > k \geq 1$ and every proper function $a(n) = O(\log n)$,

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

Corollary 2. If $\mathbf{P} \neq \mathbf{NP}$, then for every $l > k \geq 1$, $(\mathbf{DTime}[n^k] \cap \mathbf{NTime}[n])/1 \underset{\neq}{\subset} (\mathbf{DTime}[n^l] \cap \mathbf{NTime}[n])/1$.

Proof of Theorem 2. Let $\varepsilon = l - k$. W.l.o.g, $\varepsilon < 1$. Let SAT* be an ($\mathbf{NP}$-complete) version of SAT consisting of formulas in conjunctive normal form padded by $|F|^{2/\varepsilon} - |F|$ zeroes, where $|F|$ denotes the length of the binary representation of formula F. Evidently, SAT* is solvable in linear time in any reasonable model of nondeterministic computations.

Consider language A such that

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

In other words, suffixes of the words from $A \cap \{0,1\}^{n+1}$ contain all satisfiable formulas of length $n^{\varepsilon/2}$ *or less*, which means that if we are able to solve all inputs of length $n+1$, then (for an overhead of $\cdot n^{\varepsilon/2}$ in time) we are able to produce satisfying assignments for positive answers by *self-reducibility*: for a formula F and a variable v, consider two formulas $F|_{v=0}$ and $F|_{v=1}$, and if one of them is satisfiable (i.e., its properly padded version belongs to $A \cap \{0,1\}^{n+1}$), apply this procedure to it recursively; continue the recursion until the formula trivializes. It is also clear that A is $\mathbf{NP}$-complete.

The following language B is obtained from A using padding (cf. [FS04]):

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

Clearly, A reduces to B; hence, B is $\mathbf{NP}$-complete. It is also clear that all words of B of the same length correspond to words of A of equal length. Let $\text{pad}(x)$ denote the next x corresponding to the same y, i.e., $\text{pad}(x) = 0^{2^{i+1}} y$ whenever $x = 0^{2^i} y$ and $|y| < 2^i$. Denote also $\text{first}(m) =$ the smallest length of x's corresponding to y's of length m, and $\text{next}(n) = |\text{pad}(|x|)|$ for any x of length n. Let $\text{next}^i(n) = \underbrace{\text{next}(\ldots \text{next}}_{i}(n))$.

Recall from Sect. 2 that we fixed an effective enumeration of $\mathbf{DTime}[n^k]/a(n)$ machines M_i (but not of their advices). Assume $\zeta(n)$ is any slowly growing non-decreasing proper function.

Claim: If there is a $\mathbf{DTime}[n^k]/a(n)$-machine M_i and advice α for it such that M_i/α solves B correctly for all inputs of length $\text{next}(n)$ (where $\zeta(\mathit{next}(n)) \geq i$), then the following algorithm OPT running in time $O(n^{k+\varepsilon/2} \log n \cdot \zeta(n) \cdot 2^{a(n)})$ solves B correctly for all inputs of length n (the factor $n^{\varepsilon/2}$ in the time bound is for the self-reducibility procedure, and the factor $\log n$ is for simulation [HS66]).

Algorithm OPT (accepts input x of length n)**.**

```
For all machines $M = M_1, M_2, \ldots, M_{\sqrt{\zeta(n)}}$ and all advice strings $a$ of length $a(n)$ do
    • Run $M(\mathrm{pad}(x))$ for $T_M(\mathrm{next}(n))$ steps.
    • If the answer is positive, verify it by self-reducibility.
    • If the answer is correct, then output "Yes" and stop.
Output "No".
```

Select now such lengths for language C:

$$x \in C \iff \text{OPT solves } B \text{ correctly on all words of length } |x|.$$

Clearly,

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

where the advice bit says whether OPT indeed solves B correctly on all words of the corresponding length. It is also clear that for each m, $C \cap (\{0,1\}^{\mathrm{first}(m)} \cup \{0,1\}^{\mathrm{next}(\mathrm{first}(m))} \cup \ldots)$ is infinite, because almost all words of it are very "easy" (because they are padded). It remains to show that $C \notin \mathbf{DTime}[n^k]/a(n)$.

Assume that there is a $\mathbf{DTime}[n^k]/a(n)$ machine M with advice sequence $\mathfrak{A} = (\alpha_1, \alpha_2, \ldots)$ that solves C. For any m, consider the first i such that $C \cap \{0,1\}^{\mathrm{next}^i(\mathrm{first}(m))}$ is non-empty. Since M/α solves C and we have chosen the first such i, the claim above implies that $i = 0$. Since this happens for every m, M/α and hence OPT itself solve an $\mathbf{NP}$-complete problem B, which contradicts the assumption $\mathbf{P} \neq \mathbf{NP}$. □

Remark 4. *Note that a factor of $n^{\varepsilon/2} = n^{(l-k)/2}$ in the running time of the more powerful deterministic machine is due to the self-reducibility procedure. A similar theorem formulated in terms of worst-case one-way functions would not have this factor since the output of an adversary inverting f can be verified by applying f to it; thus, the time complexity of the successful adversary would be $O(n^k \log n \cdot \zeta(n) \cdot 2^{a(n)})$.*

5 Open questions

1. Our results are conditional on the existence of strongly one-way functions in the cryptographic setting and on $\mathbf{P} \neq \mathbf{NP}$ in the worse-case setting. Prove these results unconditionally.
2. Following cryptography tradition, we treat an adversary as successful if it breaks the function on an infinite number of input lengths. It would be interesting to consider a more natural formulation in terms of average-case complexity.

References

[Aar] Scott Aaronson. Complexity Zoo. `http://www.complexityzoo.com`.

[BFL91] László Babai, Lance Fortnow, and Carsten Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3-40, 1991.

[Bar02] Boaz Barak. A Probabilistic-Time Hierarchy Theorem for "Slightly Non-Uniform" Algorithms. *Lecture Notes in Computer Science*, 2483:194-208, 2002.

[Coo73] S. Cook. A hierarchy theorem for nondeterministic time complexity. *Journal of Computer and System Sciences*, 7:343–353, 1973.

[For00] L. Fortnow. Diagonalization. *Bulletin of the European Association for Theoretical Computer Science*, 71:102-112, June 2000.

[FS04] Lance Fortnow and Rahul Santhanam. Hierarchy Theorems for Probabilistic Polynomial Time. In *Proceedings of the 45th IEEE Symposium on Foundations of Computer Science*, 2004.

[FST05] Lance Fortnow, Rahul Santhanam, and Luca Trevisan. Promise Hierarchies. In *Proceedings of the 37th ACM Symposium on the Theory of Computing*, 2005.

[Gol01] Oded Goldreich. Foundations of Cryptography : Volume 1, Basic Tools. *Cambridge University Press*, 2001.

[GST04] 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.

[HS65] Juris Hartmanis and Richard E. Stearns. On the computational complexity of algorithms. *Trans. Amer. Math. Soc. (AMS)*, 117:285-306, 1965.

[HS66] F. C. Hennie and R. E. Stearns. Two-tape simulation of multitape Turing machines. *J.ACM*, 13(4):533–546, 1966.

[Lev73] Leonid Levin. Universal search problems (in Russian). *Problemy Peredachi Informatsii*, 9(3):265-266, 1973.

[vMP05] Dieter van Melkebeek and Konstantin Pervyshev. Manuscript in preparation, 2005.

[Pap91] Christos Papadimitriou. Computational Complexity. *Addison-Wesley*, 1991.

[Per05] Konstantin Pervyshev. Time Hierarchies for Computations with a Bit of Advice. In *Electronic Colloquium on Computatioinal Complexity*, TR05-054, 2005.

[TV02] Luca Trevisan and Salil Vadhan. Pseudorandomness and Average-case Complexity via Uniform Reductions. In *Proceedings of the 17th Annual IEEE Conference on Computational Complexity*, volume 17, 2002.

[Yao82] Andrew C. Yao. Theory and application of trapdoor functions. In *Proceedings of the 23th Annual IEEE Symposium on Foundations of Computer Science*, pages 80-91, 1982.

A A linear-time length-preserving one-way function

Lemma 2. Assume there is a strongly one-way function. Then there exists a length-preserving strongly one-way function computable in linear time.

Proof. Assume f is a strongly one-way function computable in time n^k where $k \geq 1$ without loss of generality. Let us define an auxiliary function t that we will use in the construction of length-preserving one-way function g to handle the variable length of images of the original function f:

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

Our new one-way function g is given by

$$g(x) = t(f(\bar{x})) \circ 0^{n-|t(f(\bar{x}))|} \tag{7}$$

where $n = |x|$ and $\bar{x} = x_{1..\bar{n}}$, a prefix of x of length $\bar{n} = \lfloor (n/2)^{1/k} \rfloor$. Function t helps us to trace the output of $f(\bar{x})$ inside $g(x)$.

Clearly, g is a length preserving function (note that $|f(\bar{x})| \leq \bar{n}^k \leq n/2$). Also, g is computable in linear time. It remains to prove that g is hard to invert. For the sake of obtaining a contradiction, assume that an adversary M inverts g with significant probability, that is for some polynomial $p(n)$ for infinitely many numbers n we have

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

where probability is taken over x uniformly distributed on $\{0,1\}^n$ and over the internal coin tosses of M.

Let us construct an adversary N that would invert f with some significant probability. The adversary is given $\bar{z} = f(\bar{x})$. Since f is honest, there are at most $\text{poly}(|\bar{z}|)$ possibilities for the length $\bar{n}$ of the shortest argument $\bar{x}'$ such that $f(\bar{x}') = \bar{z}$. Our adversary N also has to guess a number n such that $\bar{n} = \lfloor (n/2)^{1/k} \rfloor$ is a valid value for $\bar{n}$ and the adversary M breaks g at length n. This can be done with some significant probability $1/\text{poly}(\bar{n})$ as well. The adversary N performs as follows:

1. Guess n and let $\bar{n} = \lfloor (n/2)^{1/k} \rfloor$.
2. Let $z = t(\bar{z}) \circ 0^{n-|t(\bar{z})|}$.
3. Print $M(z)_{1..\bar{n}}$.

Let us denote $x = \bar{x} \circ \tilde{x}$, where $\tilde{x}$ is uniformly distributed on $\{0,1\}^{n-\bar{n}}$. Then we have

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

where the probability is taken over $\bar{x}$ uniformly distributed on $\{0,1\}^{\bar{n}}$ and over the internal coin tosses of adversary N. (Note that the second equality holds because the use of the function t guarantees that if $g(M(g(x))) = g(x)$, then $\bar{n}$ is defined unambigiously.) □

Lemma 3. Assume there is a strongly one-way permutation. Then there exists a strongly one-way permutation computable in linear time. Moreover, there is a family of trapdoor permutations, the functions constructed by this lemma form a family of trapdoor permutations as well.

Proof. Assume f is a strongly one-way permutation computable in time n^k where $k \geq 1$ without loss of generality. Let us construct a one-way permutation g such that

$$g(x) = f(\bar{x}) \circ \tilde{x} \tag{9}$$

where $n = |x|$ and $\bar{x} = x_{1..\bar{n}}$, a prefix of x of length $\bar{n} = \lfloor n^{1/k} \rfloor$, and $\tilde{x} = x_{(\bar{n}+1)..n}$, the remaining suffix of x.

Clearly, g is a permutation. Also, g is computable in linear time. It remains to prove that g is hard to invert. For the sake of obtaining a contradiction, assume that an adversary M inverts g with significant probability, that is for some polynomial $p(n)$ for infinitely many numbers n we have

$$\Pr\{M(g(x)) \in g^{-1}(g(x))\} > \frac{1}{p(n)}, \tag{10}$$

where the probability is taken over x uniformly distributed on $\{0,1\}^n$ and over the internal coin tosses of M.

Let us construct an adversary N that would invert f with some significant probability. The adversary is given $\bar{z} = f(\bar{x})$. Our adversary N performs as follows:

1. Let $\bar{n} = |\bar{z}|$ and guess n such that $\bar{n} = \lfloor n^{1/k} \rfloor$.
2. Let $z = \bar{z} \circ \tilde{z}$ where $\tilde{z}$ is uniformly distributed on $\{0,1\}^{n-\bar{n}}$.
3. Print $M(z)_{1..\bar{n}}$.

Let us denote $x = \bar{x} \circ \tilde{x}$, where $\tilde{x}$ is uniformly distributed on $\{0,1\}^{n-\bar{n}}$. Then we have

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

where the probability is taken over $\bar{x}$ uniformly distributed on $\{0,1\}^{\bar{n}}$ and over the internal coin tosses of adversary N.

The claim about trapdoor permutations is also easy to see, because a trapdoor for f serves as trapdoor for g as well (after adjusting the input length appropriately). □