Алгебра и анализ
Том 21 (2009), № 3

St. Petersburg Math. J.
Vol. 21 (2010), No. 3, Pages 459–468
S 1061-0022(10)01103-9
Article electronically published on February 26, 2010

AN INFINITELY-OFTEN ONE-WAY FUNCTION BASED ON AN AVERAGE-CASE ASSUMPTION

E. A. HIRSCH AND D. M. ITSYKSON

Abstract. We assume the existence of a function f that is computable in polynomial time but cannot be inverted by a randomized average-case polynomial algorithm. The cryptographic setting is, however, different: even for a weak one-way function, a successful adversary should fail on a polynomial fraction of inputs. Nevertheless, we show how to construct an *infinitely-often* one-way function based on f.

§1. Introduction

Let Σ be a finite alphabet. We consider functions acting from Σ^* to Σ^*, where Σ^* is the set of finite words in the alphabet Σ. We say that a function f is *one-way* if it is easy to compute but hard to invert. It is commonly assumed that a function is easy to compute if it can be computed in polynomial time. There are several approaches to defining the hardness of inverting a function.

- *Worst-case one-way functions.* A polynomial-time computable function is said to be *worst-case one-way* if its inverse cannot be computed in polynomial time. The main drawback of this definition is that there may be very few inputs difficult for inverting, so that the function may be in fact easily inverted for almost all inputs. The existence of worst-case one-way functions is equivalent to $\mathbf{P} \neq \mathbf{NP}$.
- *Cryptographic one-way functions.* A polynomial-time computable function is said to be *cryptographic one-way* if for some positive constant c every randomized polynomial-time algorithm for inverting it fails with probability at least $\frac{1}{n^c}$ for all inputs of sufficiently large length n. No reasonable complexity assumption is known that would imply the existence of cryptographic one-way functions.
- *Average-case one-way functions.* In average-case complexity theory, one considers a computational problem together with a distribution on the inputs (instances). A problem equipped with a distribution on the inputs is called a *distributional problem*. We consider only polynomial-time samplable distributions, i.e., distributions that can be generated in polynomial time from a uniform distribution. The first definition of an average-case polynomial algorithm is due to Levin [6]. According to Levin, an algorithm runs in polynomial time on the average if there exists a positive number ϵ such that the expectation of $T^\epsilon(x)$ over all inputs x of length n is $O(n)$, where $T(x)$ is the running time of the algorithm on the input x. An equivalent definition was suggested by Impagliazzo [4]; in accordance with that definition, a problem can be solved in polynomial

2000 *Mathematics Subject Classification.* Primary 68Q15.

Key words and phrases. One-way function, average-case complexity.

Partially supported by RFBR grant 08-01-00640 and the President of Russian Federation grant for support of leading scientific schools NSh-4392.2008. The first author was also supported by the Dynasty Foundation Fellowship, and the second author was also supported by the Russian Science Support Foundation.

time on the average if there exists an algorithm for solving this problem with two parameters, x (input) and δ ("give up" probability), such that its running time is bounded by a polynomial in $\frac{|x|}{\delta}$ and it "gives up" with probability at most δ (otherwise giving the correct answer). There are two ways to define the hardness of inverting a function f in the average-case complexity setting:

- The problem of inverting f with a polynomial-time samplable distribution on the *inputs* of f cannot be solved by any average-case polynomial-time randomized algorithm with bounded error. In this case, f is called an average-case one-way function.
- The problem of inverting f with a polynomial-time samplable distribution on the *outputs* of f cannot be solved by any average-case polynomial-time randomized algorithm with bounded error. In this case, one says that the problem of inverting f is hard on the average. The existence of problems hard on the average is equivalent to $(\mathbf{NP}, \mathbf{PSamp}) \not\subseteq \mathbf{AvgBPP}$, where $(\mathbf{NP}, \mathbf{PSamp})$ is the set of $\mathbf{NP}$ problems with polynomial-time samplable distributions and $\mathbf{AvgBPP}$ is the set of distributional problems that can be solved in polynomial time on the average by randomized algorithms with bounded error (this follows from the fact that any distributional $\mathbf{NP}$ search problem with a polynomial-time samplable distribution can be reduced to a search problem with a uniform distribution [5], and any search problem with a uniform distribution can be reduced to some decision problem [1]).

The existence of average-case one-way functions implies the existence of average-case hard problems, since a polynomial-time samplable distribution on the inputs of a function generates a polynomial-time samplable distribution on the outputs of this function.

The existence of cryptographic one-way functions implies the existence of average-case one-way functions (this is an easy consequence of the fact that the existence of weak one-way functions is equivalent to the existence of strong one-way functions [3]); this implies the existence of problems hard on the average, which in its turn implies $(\mathbf{NP}, \mathbf{PSamp}) \not\subseteq \mathbf{AvgBPP}$. Whether $(\mathbf{NP}, \mathbf{PSamp}) \not\subseteq \mathbf{AvgBPP}$ implies the existence of cryptographic one-way functions is a most important open problem. Whether $(\mathbf{NP}, \mathbf{PSamp}) \not\subseteq \mathbf{AvgBPP}$ implies the existence of average-case one-way functions is also an open problem. Levin [7] showed that if there exists a hard-on-the-average problem of inverting a length-preserving function with a uniform distribution, then there also exists an average-case one-way function (these two conditions are too strong, one cannot satisfy them using any known complete problem in the class $(\mathbf{NP}, \mathbf{PSamp})$).

The existence of problems hard on the average implies the existence of worst-case one-way functions, i.e., $\mathbf{P} \neq \mathbf{NP}$. The converse implication is an open problem.

There are three main differences between the approaches to the problem of inverting functions in cryptography and average-case complexity theory.

(1) A successful cryptographic adversary may err on a polynomial fraction of inputs [3, Definition 2.2.2], while an average-case polynomial-time algorithm cannot spend exponential time on a polynomial fraction of inputs [2, 6].
(2) A (polynomial-time samplable) probability distribution in the cryptographic setting is taken over the inputs of a function, while in the average-case setting, it is usually taken over the outputs (i.e., over the instances of the problem of inverting the function); see, e.g., [4, 7]. Note that it is not known whether a polynomial-time samplable distribution on the outputs is necessarily dominated by a distribution induced by a polynomial-time samplable distribution on the inputs.

(3) To solve a problem in polynomial time on the average, one must solve it for all input lengths, while in the cryptographic case, a successful adversary is allowed to solve it for infinitely many input lengths. We do not know whether this discrepancy in the definitions can be overcome, and we follow the average-case tradition. Thus, the function we obtain is hard to invert only for infinitely many input lengths.

In this paper, we overcome the difficulty described in item (1) above: we prove that the average-case hardness of inverting a function implies the cryptographic hardness of inverting a related function for infinitely many input lengths. Namely, we show how to pad any function in such a way that we can use any polynomial-time algorithm that inverts the padded function for a noticeable fraction of inputs for inverting the original function in polynomial time on the average. The reduction essentially uses the fact that the algorithm for inverting the padded function works for all sufficiently large input lengths, so that we do not overcome the difficulty described in item (3). We do not attempt to resolve the difficulty from item (2) either.

Our method employs the following simple idea of Impagliazzo's proof of [4, Proposition 3], which allows one to obtain an average-case polynomial-time algorithm from an algorithm that works on a fixed fraction $(1-\frac{1}{n^k})$ of inputs. Bogdanov and Trevisan used this idea to prove that the existence of an algorithm for a version of the bounded halting problem that fails on a fraction $\frac{1}{n}$ of inputs implies the average-case tractability of all **NP** languages with polynomial-time samplable distributions ([1, Proposition 3.5]). The trick is that if one pads the input, the probability of failure decreases. We adapt this method to the problem of inverting a function. However, instead of taking a particular function, we show how to modify *any* function.

Our main result is the following theorem.

Theorem 3. *If there exists a length-preserving polynomial-time computable function f that cannot be inverted in bounded error randomized average-case polynomial time with a polynomial-time samplable distribution on its inputs, then there exists a length-preserving function that is cryptographic one-way for infinitely many input lengths.*

1.1. Organization of the paper. In §2 we rigorously define the notions we use. In §3 we prove the main result (Theorem 3).

§2. Preliminaries

2.1. Average-case complexity. In this subsection, we define the key notions of the average-case complexity theory. Essentially, we follow [2].

We restrict ourselves to the two-letter alphabet $\{0,1\}$; the set of all words in this alphabet is denoted by $\{0,1\}^*$. By c^n we denote the string consisting of n symbols c, where $c \in \{0,1\}$.

Definition 1. An *ensemble of distributions* is a collection $D = \{D_n\}_{n=1}^{\infty}$, where $D_n\colon \{0,1\}^n \to \mathbb{R}$ is a distribution on the inputs of length n (i.e., $\sum_{a\in\{0,1\}^n} D_n(a) = 1$). An ensemble of distributions is said to be *polynomial-time samplable* if there exists a polynomial-time randomized algorithm obtaining 1^n (the string consisting of n ones) as the input whose outputs are distributed according to D_n.

In [1], D_n is defined not only on strings of length n, but possibly on strings of close lengths. In the case of polynomial-time samplable distributions, these approaches are equivalent. The uniform distribution U is determined by the ensemble of distributions U_n, where U_n is the uniform distribution on $\{0,1\}^n$.

Definition 2. A function $f : \{0,1\}^* \to 2^{\{0,1\}^*}$ is *polynomial-time verifiable* if the length of every string in its output is bounded by a polynomial in the length of the input and there exists a polynomial-time computable function v such that

$$\forall x, y \in \{0,1\}^* \quad v(x,y) = 1 \iff y \in f(x).$$

Definition 3. A distributional search problem (f, D) consists of a polynomial-time verifiable function $f\colon \{0,1\}^* \to 2^{\{0,1\}^*}$ and an ensemble of distributions D.

Remark 1. Bogdanov and Trevisan [2] considered distributional search problems for **NP** languages instead of multivalued polynomial-time verifiable functions. These approaches are obviously equivalent.

Following Impagliazzo [4] and Bogdanov and Trevisan [2], we define average-case polynomial-time algorithms as polynomial-time algorithms that are allowed to "give up" (by outputting a special symbol $\perp$).

Definition 4 (cf. [2, Definition 4.2])**.** A distributional search problem (f, D) *can be solved in polynomial time on the average* if there exists an algorithm $A(x, \delta)$ such that

- A runs in time polynomial in $|x|$ and $\frac{1}{\delta}$ for any x in the support of D and any positive δ;
- if $f(x) \neq \varnothing$, then $A(x, \delta) \in f(x) \cup \{\perp\}$;
- $\Pr_{x \leftarrow D_n}\{A(x, \delta) = \perp\} \leq \delta$.

Definition 5. The complexity class **FAvgP** consists of all distributional search problems that can be solved in polynomial time on the average.

Definition 6 (cf. [2, Definition 4.3])**.** A distributional search problem (f, D) can be solved in randomized polynomial time on the average if there exists a randomized algorithm $A(x, \delta)$ such that

- A runs in time polynomial in $|x|$ and $\frac{1}{\delta}$ for any x in the support of D and any positive δ;
- if $f(x) \neq \varnothing$, then $\Pr\{A(x, \delta) \notin f(x) \cup \{\perp\}\} \leq \frac{1}{4}$, where the probability is taken over the random bits of A;
- $\Pr_{x \leftarrow D_n}\{\Pr\{A(x, \delta) = \perp\} \geq \frac{1}{4}\} \leq \delta$, where the inner probability is taken over the random bits of A.

Definition 7. The complexity class **FAvgBPP** consists of all distributional search problems that can be solved in randomized polynomial time on the average.

The following definition of (deterministic) reductions is a special case of randomized heuristic search reductions [2, 5.1.1]. It is not known whether **FAvgBPP** is closed under these randomized reductions, but it is closed under the deterministic ones. In this paper, we use only deterministic reductions.

Definition 8. Consider two distributional search problems (f, D) and (f', D'). We say that (f, D) *reduces to* (f', D') if there are polynomial-time computable functions h and g such that the following three assertions hold true:

- $f(x) \neq \varnothing \implies f'(h(x)) \neq \varnothing$;
- $y \in f'(h(x)) \implies g(y) \in f(x)$ for any y and x with $D_{|x|}(x) > 0$;
- there is a polynomial $p(n)$ such that

$$\sum_{h(x)=x',\ |x|=n} D_n(x) \leq p(n) \cdot D'_{|x'|}(x')$$

for any x'.

(The last condition is called the *domination* condition.)

Now we formally verify that both **FAvgP** and **FAvgBPP** are closed under such reductions.

Lemma 1. *If a problem (f, D) is reducible to (f', D'), then $(f', D') \in$ **FAvgP** implies $(f, D) \in$ **FAvgP**.*

Proof. Let $A'(y, \delta)$ be an average-case polynomial-time algorithm for (f', D'). Let q be a polynomial such that $\max_{x \in \{0,1\}^n} |h(x)| \le q(n)$. Define

$$A(x, \delta) = g\left(A'\left(h(x), \frac{\delta}{p(|x|)q(|x|)}\right)\right)$$

(we assume that $g(\perp) = \perp$). Clearly, the algorithm A is polynomial in $|x|$ and $\frac{1}{\delta}$ and does not output wrong answers when $f(x) \neq \varnothing$. The probability of the "give up" answer can be estimated as

$$\begin{aligned}\Pr_{x \leftarrow D_n}\{A(x, \delta) = \perp\} &= \sum_{A'(h(x), \frac{\delta}{p(n)q(n)}) = \perp} D_n(x) \\ &\le \sum_{A'(y, \frac{\delta}{p(n)q(n)}) = \perp} p(n) D'_{|y|}(y) \le p(n)q(n)\frac{\delta}{p(n)q(n)} = \delta. \qquad \square\end{aligned}$$

Lemma 2. *If a problem (f, D) is reducible to (f', D'), then $(f', D') \in$ **FAvgBPP** implies $(f, D) \in$ **FAvgBPP**.*

Proof. The construction of the new algorithm A and the estimation of the probability of the "give up" answer are similar to the deterministic case (see Lemma 1):

$$\begin{aligned}\Pr_{x \leftarrow D_n}\left\{\Pr\{A(x, \delta) = \perp\} \ge \frac{1}{4}\right\} &= \sum_{\Pr\{A'(h(x), \frac{\delta}{p(n)q(n)}) = \perp\} \ge \frac{1}{4}} D_n(x) \\ &\le \sum_{\Pr\{A'(y, \frac{\delta}{p(n)q(n)}) = \perp\} \ge \frac{1}{4}} p(n) D'_{|y|}(y) \le p(n)q(n)\frac{\delta}{p(n)q(n)} = \delta.\end{aligned}$$

The additional condition for randomized algorithms can also be verified easily:

$$\Pr\{A(x, \delta) \notin f(x) \cup \{\perp\}\} \le \Pr\left\{A'(h(x), \frac{\delta}{p(n)q(n)}) \notin f'(h(x)) \cup \{\perp\}\right\} \le \frac{1}{4}. \qquad \square$$

2.2. Infinitely-often one-way functions. In this section, we define infinitely-often one-way functions, which differ from "standard" one-way functions in that they are hard only for infinitely many (rather than for almost all) input lengths. In other words, in contrast to, e.g., [3, Definition 2.2.1], a successful adversary must invert the function for all sufficiently large input lengths, while in the "classical" definition, it suffices to invert the function for infinitely many input lengths.

Definition 9. A polynomial-time computable function $f\colon \{0,1\}^* \to \{0,1\}^*$ is a *strong infinitely-often one-way* (i.o. one-way) function if for any polynomial $p(n)$ and any randomized polynomial-time algorithm B,

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

where the probability is taken over x uniformly distributed on $\{0,1\}^n$ and over the random bits used by B.

As in the "usual" case, we also define weak infinitely-often one-way functions (cf. [3, Definition 2.2.2]).

Definition 10. A polynomial-time computable function $f\colon \{0,1\}^* \to \{0,1\}^*$ is a weak i.o. one-way function if there exists a polynomial $p(n)$ such that for any randomized polynomial-time algorithm B,

$$\forall N\ \exists n > N\ \Pr\{B(f(x)) \notin f^{-1}(f(x))\} > \frac{1}{p(n)},$$

where the probability is taken over x uniformly distributed on $\{0,1\}^n$ and over the random bits used by B.

Theorem 1. *The existence of weak i.o. one-way functions implies the existence of strong i.o. one-way functions.*

Proof. The proof reproduces the proof of [3, Theorem 2.3.2] (for "usual" one-way functions) word for word. □

§3. Main result

3.1. Strategy of the proof. We assume the existence of a length-preserving function $f\colon \{0,1\}^* \to \{0,1\}^*$ such that the problem of inverting f with the uniform[1] distribution on the inputs of f cannot be solved in randomized polynomial time on the average. Note that f is not necessarily a weak i.o. one-way function. Indeed, suppose that f is polynomial-time invertible on a set of probability $(1 - \frac{1}{2^{\sqrt{n}}})$, and invertible in time $\Omega(2^n)$ on the other inputs. In this case, f may be hard to invert in polynomial time on the average, but it is not weakly i.o. one-way. We shall show how to modify f so as to make it weakly i.o. one-way.

The main idea of the proof is to supply the original function with a *padding*. Namely, we define a new function $f_p(x,y)$ on pairs of strings; it applies f to the first argument and replaces the second argument by $1^{|y|}$. Note that the probability of an input $f_p(x,y)$ does not depend on the length of the padding (i.e., the length of y): this probability is exactly $2^{-|f(x)|}$. By definition, a randomized average-case polynomial-time algorithm must solve much more instances for greater lengths (the error probability is a constant), and padding allows us to put instances of smaller lengths among instances of higher lengths.

We show that the problem of inverting f is average-case reducible to that of inverting f_p. Indeed, in order to invert f on a string y, it suffices to invert f_p on the pair $(y,1)$. To verify the domination condition, we must specify an economical encoding of pairs (to satisfy this condition, we are allowed to increase the length of the input only by a logarithmic number). The details of the encoding are described in the next section.

Assume that there is a randomized polynomial-time algorithm B that inverts f_p with error $1/n$: $\Pr\{B(f_p(z)) \in f_p^{-1}(f_p(z))\} \geq 1 - \frac{1}{n}$, where the probability is taken both over z and over the random choices of B. Now we define a randomized average-case polynomial-time algorithm $A(z,\delta)$ that inverts f_p. The paradigm is as follows: increase the padding of the input $z \mapsto z1^{\lceil \frac{1}{\delta} \rceil}$ and then use B.

Since the probability of the input does not depend on the length of the padding, the error probability of A is at most $\frac{1}{n+\frac{1}{\delta}} \leq \delta$. Thus, inverting f_p and, by the above reduction, inverting f are randomized average-case tractable, contradicting the assumption.

3.2. Details of the proof. Throughout this section, log denotes the binary logarithm. Given a number n, we denote its binary representation by n_2 (in particular, for a string x, we denote the binary representation of its length by $|x|_2$).

[1]Later we shall show that here one can take any polynomial-time samplable distribution.

Definition 11. We say that a string $x \in \{0,1\}^*$ is *correct* if

- x contains at least one occurrence of 0; let $x_k = 0$ be the first such occurrence;
- $|x| \geq 2k - 1$;
- the substring $x_{k+1} \dots x_{2k-1}$ is the binary representation of a number l such that $|x| \geq 2k + l - 1$.

Given a correct string x, its *main part* is the substring $x_{2k} \dots x_{2k+l-1}$, and its *padding* is the suffix $x_{2k+l} \dots x_{|x|}$.

Definition 12. Let $\pi\colon \{0,1\}^* \to \{0,1\}^*$ be the function that maps a string x to the correct string $\pi(x)$ with the main part x and the empty padding, i.e., $\pi(x) = 1^{\lceil \log |x| \rceil} 0 |x|_2 x$.

Remark 2. Note that π is injective.

Definition 13. Let $f\colon \{0,1\}^* \to \{0,1\}^*$ be a length-preserving function. We define a new (also length-preserving) function f_p as follows: if there are y and z such that $x = 1^{\lceil \log |z| \rceil} 0 |z|_2 zy$, then $f_p(x) = 1^{\lceil \log |z| \rceil} 0 |z|_2 f(z) 1^{|y|}$; otherwise $f_p(x) = x$.

Definition 14. Given a length-preserving function g and a distribution D, let D^g be the distribution of the outputs of g whose inputs are sampled according to D. In particular,

$$U^g(y) = \sum_{g(x)=y} 2^{-|x|} = |g^{-1}(y)| \cdot 2^{-|y|}.$$

Lemma 3. *If $x = 1^{\lceil \log |z| \rceil} 0 |z|_2 z 1^t$, then*

$$U^{f_p}(x) = |f^{-1}(z)| \cdot 2^{-|z| - 2\lceil \log |z| \rceil - 1} = U^{f_p}(x1^s)$$

for any $s \geq 0$. If x is a correct string whose padding contains zeros, then $U^{f_p}(x) = 0$. If a string x is incorrect, then $U^{f_p}(x) = 2^{-|x|}$.

Proof. In the first case, one should sum the probabilities for all the different paddings of a given length. The other two cases are trivial. □

Lemma 4. *For any length-preserving function f, the problem (f^{-1}, U^f) is reducible to (f_p^{-1}, U^{f_p}).*

Proof. To satisfy Definition 8, we put

$$h(x) = \pi(x),$$
$$g(y) = \begin{cases} x & \text{if } \ y = \pi(x), \\ y & \text{otherwise,} \end{cases}$$
$$p(n) = 2n^3.$$

If f is invertible on x, then f_p is invertible on $\pi(x)$. If f_p is invertible on $\pi(x)$, then $g(f_p^{-1}(\pi(x))) \in f^{-1}(x)$. If f is not invertible on x, then trivially $U^f(x) = 0$. Let $n = |x|$. Finally, if $x' = \pi(x)$, then, by Lemma 3,

$$\begin{aligned} U^{f_p}(x') = |f^{-1}(x)| \cdot 2^{-n - 2\lceil \log n \rceil - 1} &\geq \frac{1}{2n^3} \cdot |f^{-1}(x)| 2^{-n} \\ &= \frac{1}{p(n)} U_n^f(x) = \frac{1}{p(n)} \sum_{\pi(y)=x'} U_n^f(y). \end{aligned}$$

The last identity is valid because π is injective. (If x' cannot be represented as $\pi(x)$, then the domination condition is satisfied trivially.) □

Theorem 2. *Let f be a length-preserving polynomial-time computable function. If there exists a randomized polynomial-time algorithm B such that, for a constant $c > 0$ and every integer n, we have*

$$\Pr_{x\leftarrow U_n^{f_p}, r\leftarrow U_{s(n)}}\{B(x) \in f_p^{-1}(x)\} \geq 1 - \frac{1}{n^c},$$

where r is the string of random bits used by the algorithm B and $s(n)$ is a polynomial, then $(f_p^{-1}, U_n^{f_p}) \in \mathbf{FAvgBPP}$.

Proof. Since one can verify the answer of B, we assume that B either correctly inverts f_p or gives up. We also assume that B returns an element of $f_p^{-1}(x)$ whose padding does not contain zeros.

First, we prove that

$$\Pr_{x\leftarrow U_n^{f_p}}\left\{\Pr_{r\leftarrow U_{s(n)}}\{B(x) \notin f_p^{-1}(x)\} \geq \frac{1}{4}\right\} \leq \frac{4}{n^c}. \tag{1}$$

Indeed, if this inequality fails, then

$$\begin{aligned}\Pr_{x\leftarrow U_n^{f_p}, r\leftarrow U_{s(n)}}\{B(x) \notin f_p^{-1}(x)\} &= \sum_{x\in\{0,1\}^n} U_n^{f_p}(x)\cdot \Pr_{r\leftarrow U_{s(n)}}\{B(x) \notin f_p^{-1}(x)\}\\ &\geq \sum_{\substack{x\in\{0,1\}^n\\ \Pr_{r\leftarrow U_{s(n)}}\{B(x)\notin f_p^{-1}(x)\}\geq\frac14}} U_n^{f_p}(x)\cdot \Pr_{r\leftarrow U_{s(n)}}\{B(x) \notin f_p^{-1}(x)\} > \frac14\cdot\frac{4}{n^c} = \frac{1}{n^c},\end{aligned}$$

contradicting the assumption on B.

The new algorithm $A(x,\delta)$ performs as follows. If x is an incorrect string, then $A(x,\delta) = x$. If x is a correct string whose padding does not contain zeros (note that if the padding contains zeros, then $U^{f_p}(x) = 0$), then we pad x and run B. More precisely, let $|x| = n$, $\Delta = \lceil(\frac{4}{\delta})^{1/c}\rceil$, $N = n+\Delta$. Define $\sigma(x) = x1^\Delta$. If $B(\sigma(x)) = \perp$, then $A(x,\delta) = \perp$; otherwise $A(x,\delta)1^\Delta = B(\sigma(x))$, i.e., A discards Δ trailing 1's of the answer of B and outputs the result.

Then

$$\begin{aligned}&\Pr_{x\leftarrow U_n^{f_p}}\left\{\Pr_{r\leftarrow U_{s(N)}}\{A(x,\delta) \notin f_p^{-1}(x)\} \geq \frac14\right\}\\ &\quad\leq \Pr_{x\leftarrow U_n^{f_p}}\left\{\Pr_{r\leftarrow U_{s(N)}}\{B(\sigma(x)) \notin f_p^{-1}(\sigma(x))\} \geq \frac14\right\}\\ &\quad\overset{\text{Lemma 3}}{=} \Pr_{y\leftarrow U_N^{f_p}}\left\{\exists x(x\in\{0,1\}^n \wedge y = \sigma(x)) \wedge \Pr_{r\leftarrow U_{s(N)}}\{B(y) \notin f_p^{-1}(y)\} \geq \frac14\right\}\\ &\quad\leq \Pr_{y\leftarrow U_N^{f_p}}\left\{\Pr_{r\leftarrow U_{s(N)}}\{B(y) \notin f_p^{-1}(y)\} \geq \frac14\right\} \overset{(1)}{\leq} \frac{4}{N^c} < \delta.\end{aligned}$$

□

Corollary 1. *Let f be a length-preserving polynomial-time computable function. If $(f^{-1}, U_n^f) \notin \mathbf{FAvgBPP}$, then for any randomized polynomial-time algorithm B and any constant $c > 0$, there exist infinitely many $n \in \mathbb{N}$ such that $\Pr_{x\leftarrow U_n, r\leftarrow U_{s(n)}}\{B(f_p(x)) \in f_p^{-1}(f_p(x))\} < 1 - \frac{1}{n^c}$.*

Proof. This follows from Theorem 2, Lemma 4, and Lemma 2. □

Using Theorem 1, we obtain the following assertion.

Corollary 2. *If there exists a length-preserving polynomial-time computable function f that cannot be inverted in randomized average-case polynomial time (i.e., $(f^{-1}, U^f) \notin$ **FAvgBPP**), then there exists a length-preserving strong i.o. one-way function.*

Corollary 2 formally requires that the function used in the assumption have a uniform distribution on its inputs. However, the assertion is true for any other polynomial-time samplable distribution on the inputs.

Theorem 3. *If there exists a length-preserving polynomial-time computable function f that cannot be inverted in randomized average-case polynomial time on a polynomial-time samplable distribution D (i.e., $(f^{-1}, D^f) \notin$ **FAvgBPP**), then there exists a length-preserving strong i.o. one-way function.*

Proof. Let $s\colon \{0,1\}^{m(n)} \to \{0,1\}^n$ be a polynomial-time sampler for D, i.e., $D = U^s$. Without loss of generality we may assume that $m(n) \geq n$ and $m(n)$ is a strictly monotone increasing polynomial function. We define a function f^s as follows:

$$f^s(x) = \begin{cases} f(s(x))1^{m(n)-n} & \text{if } \exists n(m(n) = |x|), \\ x & \text{otherwise.} \end{cases}$$

We reduce (f^{-1}, D^f) to $((f^s)^{-1}, U^{f^s})$. To satisfy Definition 8, put

$$\begin{aligned} h(y) &= y1^{m(n)-n}, \\ g(x) &= s(x), \\ p(n) &= 1. \end{aligned}$$

The domination condition is satisfied because $D^f(y) = U^{f^s}(h(y))$.

Since **FAvgBPP** is closed under reductions, we have $((f^s)^{-1}, U^{f^s}) \notin$ **FAvgBPP**. Now the theorem follows from Corollary 2. □

Acknowledgments

The authors are very grateful to Dima Grigoriev, Arist Kojevnikov, Sergey Nikolenko, and Alexander Shen for helpful discussions, and to the anonymous referee of ECCC for the term "i.o. one-way function" and valuable comments that improved the presentation of the paper.

References

[1] Shai Ben-David, Benny Chor, Oded Goldreich, and Michael Luby, *On the theory of average case complexity*, J. Comput. System Sci. **44** (1992), no. 2, 193–219; http://dx.doi.org/10.1016/0022-0000(92)90019-F. MR1160461 (93d:68020)

[2] A. Bogdanov and L. Trevisan, *Average-case complexity*, Found. Trends Theor. Comput. Sci. **2** (2006), no. 1, 1–106. MR2453146

[3] O. Goldreich, *Foundations of cryptography. Basic tools*, Cambridge Univ. Press, Cambridge, 2001. MR1881185 (2004a:94042)

[4] R. Impagliazzo, *A personal view of average-case complexity*, 10th Annual Conference. Structure in Complexity Theory (SCT'95) (Minneapolis, Minnesota, 1995), IEEE Comput. Soc. Press, Los Alamitos, CA, 1995, pp. 134–147.

[5] R. Impagliazzo and L. Levin, *No better ways to generate hard NP instances than picking uniformly at random*, 31st Annual Symposium on Foundations of Computer Science, Vol. II (St. Louis, MO, 1990), IEEE Comput. Soc. Press, Los Alamitos, CA, 1990, pp. 812–821.

[6] L. Levin, *Average case complete problems*, SIAM J. Comput. **15** (1986), no. 1, 285–286. MR0822205 (87b:68053)
[7] ———, *One-way functions*, Probl. Peredachi Inf. **39** (2003), no. 1, 103–117; English transl., Probl. Inf. Transm. **39** (2003), no. 1, 92–103. DOI 10.1023/A:1023634616182. MR2101668 (2005g:94080)

St. Petersburg Branch, Steklov Institute of Mathematics, 27 Fontanka, St. Petersburg 191023, Russia
E-mail address: hirsch@pdmi.ras.ru

St. Petersburg Branch, Steklov Institute of Mathematics, 27 Fontanka, St. Petersburg 191023, Russia
E-mail address: dmitrits@pdmi.ras.ru

Received 29/MAY/2008

Translated by THE AUTHORS