Electronic Colloquium on Computational Complexity, Report No. 117 (2007)

An infinitely-often one-way function based on an average-case assumption

Edward A. Hirsch* Dmitry M. Itsykson†

November 8, 2007

Abstract

We assume the existence of a function f that is computable in polynomial time but its inverse function is not computable in randomized average-case polynomial time. The cryptographic setting is, however, different: even for a weak one-way function, every possible 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

One-way functions are one of the main cryptographic primitives. However, no reasonable complexity assumption (such as, say, $\mathbf{P} \neq \mathbf{NP}$ or $\mathbf{AvgP} \neq \mathbf{DistNP}$) is known that would imply the existence of one-way functions. In particular, the three obstacles that prevent using Levin's average-case complexity notions in the cryptographic setting, are

1. A successful cryptographic adversary may err on a polynomial fraction of inputs [Gol01, Definition 2.2.2], while in the average-case setting this is not enough to solve a problem: if one spends exponential time on these inputs, the average-case complexity is not polynomial [Lev86, BT06].

2. The (polynomially samplable) probability distribution for the cryptographic setting is taken over function inputs, while in the average-case setting it is taken over the outputs (i.e., the instances of the search problem of computing the inverse function): see, e.g., [Imp95, Lev03]. Note that a polynomially samplable distribution on the outputs is not necessarily dominated by the distribution induced by a polynomially samplable distribution on the inputs.

*Steklov Institute of Mathematics at St. Petersburg. Web: `http://logic.pdmi.ras.ru/~hirsch/`. Partially supported by grant RFBR 05-01-00932, INTAS grant 04-77-7173, the president of Russia grant NSh-8464.2006.1, and the Dynasty foundation fellowship.

†Steklov Institute of Mathematics at St. Petersburg. Web: `http://logic.pdmi.ras.ru/~dmitrits/`. Partially supported by INTAS grant 04-77-7173, the president of Russia grant NSh-8464.2006.1, and St. Petersburg government grant.

ISSN 1433-8092

3. To solve a problem in the average-case, one should solve it on all input lengths, while in the cryptographic case a successful adversary is allowed to solve it just on an infinite number of input lengths. We do not know if this definitional discrepancy can be overcome, and follow the average-case tradition in this work. Thus the function that we obtain is hard to invert only on an infinite number of (rather than on almost all) input lengths.

In this paper we address item 1 of the above list: we prove that average-case hardness of inverting a function implies cryptographic hardness of inverting a related function on an infinite number of input lengths. Namely, we show how to pad any function so that we can use any polynomial-time algorithm that inverts the padded function with any noticeable probability of success for inverting the padded function (as well as the original non-padded function) in polynomial-time on the average. The reduction essentially uses the fact that the two concepts use a similarly defined set of input lengths, thus we do not resolve item 3. We do not attempt to resolve item 2 either.

Our method uses the following simple idea of the proof of [Imp95, Proposition 3] for an algorithm that fails on an $o(1)$ fraction of inputs of the same length, if one pads the input, the failure probability decreases. Bogdanov and Trevisan use this idea to prove that the existence of an algorithm that fails on a $\frac{1}{n}$ fraction of inputs for a version of the bounded halting problem implies the average-case easiness of all **NP** languages on polynomially samplable distributions. We adapt this method for the problem of inverting a function. Instead of taking a particular function we show how to modify *any* function to fit it. Note that a straightforward approach of applying the argument to a (**DistNP**-hard) search version of the bounded halting problem fails because for the (cryptographic) problem of inverting a function one needs a (polynomially samplable) probability distribution on inputs of this function and not on its outputs.

Organization of the paper. In Sect. 2 we define rigorously the notions we use. In Sect. 3 we prove the main result.

2 Preliminaries

2.1 Average-case complexity

In this subsection we define the core notions of the average-case complexity. We basically follow [BT06] (techinically, [BT06] allows distributions that are defined not on every input length, but it does not make any difference for us).

Definition 2.1. 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 inputs of length n (i.e., $\sum_{a \in \{0,1\}^n} D_n(a) = 1$).

Definition 2.2. A function $f \colon \{0,1\}^* \to 2^{\{0,1\}^*}$ is called polynomially verifiable if every string in its output is polynomially bounded 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 2.3. A distributed search problem (f, D) consists of a polynomially verifiable function $f\colon \{0,1\}^* \to 2^{\{0,1\}^*}$ and an ensemble of distributions D.

Remark 2.1. Bogdanov and Trevisan [BT06] consider search algorithms for **NP** languages instead of formally defining distributed search problems, though these approaches are obviously equivalent.

We follow Impagliazzo [Imp95] and Bogdanov and Trevisan [BT06] in defining average-case polynomial-time algorithms as polynomial-time algorithms that are allowed to "give up" (by outputting a special symbol $\perp$).

Definition 2.4 (cf. [BT06, Definition 4.2])**.** A distributed 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 $f(x) \neq \emptyset$, then $A(x, \delta) \in f(x) \cup \{\perp\}$;
- $\Pr_{x \leftarrow D_n}\{A(x, \delta) = \perp\} \leq \delta$.

Definition 2.5. Complexity class **FAvgP** consists of all distributed search problems that can be solved in polynomial time on the average.

Definition 2.6 (cf. [BT06, Definition 4.3])**.** A distributed 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 $f(x) \neq \emptyset$, 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 2.7. Complexity class **FAvgBPP** consists of all distributed search problems that can be solved in randomized polynomial time.

The following definition of a (deterministic) reduction is a special case of randomized heuristic search reduction [BT06, 5.1.1]. While **FAvgBPP** might not be closed under these randomized reductions, it is closed under the deterministic ones. In this paper we use only deterministic reductions.

Definition 2.8. Consider two distributed search problems (f, D) and (f', D'). We say that (f, D) reduces to (f', D'), if there are polynomial-time computable functions h, g such that the two following statements hold:

- $f(x) \neq \emptyset \implies f'(h(x)) \neq \emptyset$;
- $y \in f'(h(x)) \implies g(y) \in f(x)$ 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.) We now formally verify that both **FAvgP** and **FAvgBPP** are closed under such reductions.

Lemma 2.1. If a problem (f, D) is reducible to a problem (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 the problem (f', D'). Define $A(x, \delta) = g(A'(h(x), \frac{\delta}{p(|x|)}))$ (we assume $g(\perp) = \perp$). Clearly, the algorithm A is polynomial in $|x|$ and in $\frac{1}{\delta}$ and does not output wrong answers when $f(x) \neq \emptyset$. Let q be a polynomial such that $\max_{x \in \{0,1\}^n} |h(x)| \leq q(n)$. The probability of the "give up" answer can be estimated as

$$\Pr_{x \leftarrow D_n}\{A(x, \delta) = \perp\} = \sum_{A'(h(x), \frac{\delta}{p(n)q(n)}) = \perp} D_n(x) \leq \sum_{A'(y, \frac{\delta}{p(n)q(n)}) = \perp} p(n) D'_{|y|}(y) \leq$$

$$p(n)q(n)\frac{\delta}{p(n)q(n)} = \delta.$$

□

Lemma 2.2. If a problem (f, D) is reducible to a problem (f', D') then $(f', D') \in$ **FAvgBPP** implies $(f, D) \in$ **FAvgBPP**.

Proof. The construction of the new algorithm A and the verification of the probability of the "give up" answer is similar to the deterministic case (Lemma 2.1):

$$\Pr_{x \leftarrow D_n}\{\Pr\{A(x, \delta) = \perp\} \geq \frac{1}{4}\} = \sum_{\Pr\{A'(h(x), \frac{\delta}{p(n)q(n)}) = \perp\} \geq \frac{1}{4}} D_n(x)$$

$$\leq \sum_{\Pr\{A'(y, \frac{\delta}{p(n)q(n)}) \geq \frac{1}{4}\} = \perp} p(n) D'_{|y|}(y) \leq p(n)q(n)\frac{\delta}{p(n)q(n)} = \delta.$$

The additional condition can also be easily verified:

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

□

2.2 Infinitely-often one-way functions

In this section we define infinitely-often one-way functions[1] that differ from "standard" one-way functions in that they are hard only on an infinite number of (rather than on almost all) input lengths. In other words, in a contrast to, e.g., [Gol01, Definition 2.2.1] we require the adversary to invert the function on all sufficiently large input lengths to violate the one-wayness condition, while in the "classical" definition it is enough to invert it on an infinite number of input lengths.

Definition 2.9. A polynomial-time computable function $f\colon \{0,1\}^* \to \{0,1\}^*$ is a strong 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.

Remark 2.2. We adjust the definition of weak one-way functions similarly (cf. [Gol01, Definition 2.2.2] for "ordinary" one-way functions).

Definition 2.10. A polynomial-time computable function $f\colon \{0,1\}^* \to \{0,1\}^*$ is a weak i.o.-one-way function if there exists 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 2.1. The existence of weak i.o.-one-way functions implies the existence of strong i.o.-one-way functions.

Proof. The proof repeats the proof of [Gol01, Theorem 2.3.2] (for "ordinary" one-way functions) literally. □

3 Main result

3.1 Proof strategy

We assume the existence of a length-preserving function $f\colon \{0,1\}^* \to \{0,1\}^*$ such that the search problem of inverting f on the distribution resulting from the uniform distribution on the inputs of f cannot be solved in randomized polynomial time on average. We will show how to modify f so that it becomes i.o.-one-way.

[1] The term was suggested to us by an anonymous referee of ECCC.

The main idea of the proof is to supply the original function with *padding*. Namely, we define a new function $f_p(x, y)$ on pairs of strings that applies f to its 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 padding (the probability is exactly $2^{-|f(x)|}$). By the definition of the randomized average-case polynomial algorithm it is required to solve much more instances at higher lengths (the probability of error is a constant), and padding allows us to put the instances of smaller lengths into higher lengths.

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

Suppose that there is a randomized polynomial-time algorithm B that inverts f_p with $1/n$ error: $\Pr\{B(f_p(x)) \in f_p^{-1}(f_p(x))\} \geq 1 - \frac{1}{n}$ where the probability is taken both by x and by the random choices of B. We now define a randomized average-case polynomial-time algorithm $A(x, \delta)$ that inverts f_p. The paradigm is: increase the padding of the input $x \mapsto x1^{\lceil \frac{1}{\delta} \rceil}$ and then use B. Since the probability of the input does not depend on the length of padding, the probability of error of A is at most $\frac{1}{n+\frac{1}{\delta}} \leq \delta$. Thus, inverting f_p, and, by the reduction above, inverting f, is randomized average-case tractable, which contradicts the assumption.

3.2 Proof details

Throughout this section log denotes the binary logarithm. For a string x, we denote the binary representation of its length $|x|$ (as a string) by $|x|_2$.

Definition 3.1. 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 $x_{k+1} \ldots x_{2k-1}$ is the binary representation of the number l such that $|x| \geq 2k + l - 1$.

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

Definition 3.2. Let $\pi\colon \{0,1\}^* \to \{0,1\}^*$ be a 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 3.1. Note that π is injective.

Definition 3.3. Let $f\colon \{0,1\}^* \to \{0,1\}^*$ be a length-preserving function. We then 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 3.4. For any length-preserving function g, the distribution U^g is generated as the output of the function g whose inputs are sampled according to the uniform distribution U, i.e.,

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

Lemma 3.1. 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 zeroes, 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 has to sum up the probabilities for different original paddings. The other two cases are trivial. □

Lemma 3.2. 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 2.8, assume

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

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 3.1

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

The last equality holds since π is injective. (If x' cannot be represented as $\pi(x)$, the domination condition is trivially satisfied.)

□

Theorem 3.1. 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 $\Pr_{x \leftarrow U_n^{f_p}, r \leftarrow U_{s(n)}} \{B(x) \in f_p^{-1}(x)\} \geq 1 - \frac{1}{n^c}$, where r is the string of random bits used by the algorithm B, $s(n)$ 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 either B correctly inverts f_p or gives up. We also assume that when B returns an element of $f_p^{-1}(x)$, it chooses one without zeroes in its padding.

We first prove that

$$\Pr_{x \leftarrow U_n^{f_p}} \left\{ \Pr_{r \leftarrow U_{s(n)}} \{B(x) \notin f_p^{-1}(x)\} \geq \frac{1}{4} \right\} \leq \frac{4}{n^c}. \tag{3.1}$$

Indeed, if this inequality is incorrect, then

$$\Pr_{x\leftarrow U_n^{f_p}, r\leftarrow U_{s(n)}}\{B(x)\notin f_p^{-1}(x)\} = \sum_{x\in\{0,1\}^n} U_n^{f_p}(x)\cdot \Pr_{r\leftarrow U_{s(n)}}\{B(x)\notin f_p^{-1}(x)\} \geq$$

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

which contradicts the assumption about 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 with padding without 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 strips Δ trailing 1's of B's answer and outputs the result.

Then

$$\Pr_{x\leftarrow U_n^{f_p}}\left\{\Pr_{r\leftarrow U_{s(N)}}\{A(x,\delta)\notin f_p^{-1}(x)\}\geq\frac{1}{4}\right\}\leq$$

$$\Pr_{x\leftarrow U_n^{f_p}}\left\{\Pr_{r\leftarrow U_{s(N)}}\{B(\sigma(x))\notin f_p^{-1}(\sigma(x))\}\geq\frac{1}{4}\right\} \overset{\text{Lemma 3.1}}{=}$$

$$\Pr_{y\leftarrow U_N^{f_p}}\left\{\exists x(x\in\{0,1\}^n\wedge y=\sigma(x))\wedge\Pr_{r\leftarrow U_{s(N)}}\{B(y)\notin f_p^{-1}(y)\}\geq\frac{1}{4}\right\}\leq$$

$$\Pr_{y\leftarrow U_N^{f_p}}\left\{\Pr_{r\leftarrow U_{s(N)}}\{B(y)\notin f_p^{-1}(y)\}\geq\frac{1}{4}\right\}\overset{(3.1)}{\leq}\frac{4}{N^c}<\delta.$$

□

Corollary 3.1. Let f be a length-preserving polynomial-time computable function. If the problem $(f^{-1}, U_n^f)\notin$ **FAvgBPP**, then for any randomized polynomial-time algorithm B and for 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. By Theorem 3.1, Lemma 3.2, and Lemma 2.2. □

Using Theorem 2.1 one gets

Corollary 3.2. 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.

4 Acknowledgement

The authors are very grateful to Dima Grigoriev, Arist Kojevnikov, Sergey Nikolenko, and Alexander Shen for helpful discussions, and to an anonymous referee of ECCC for valuable comments that improved the presentation of the paper.

References

[BT06] Andrej Bogdanov and Luca Trevisan. Average-case complexity. *Foundation and Trends in Theoretical Computer Science*, 2(1):1–106, 2006.

[Gol01] Oded Goldreich. *Foundations of Cryptography*, volume 1. Cambridge University Press, 2001.

[Imp95] Russell Impagliazzo. A personal view of average-case complexity. In *Proceedings of the 10th Annual Structure in Complexity Theory Conference (SCT'95)*, pages 134–147, 1995.

[Lev86] L. Levin. Average case complete problems. *SIAM Journal on Computing*, 15(1):285–286, 1986.

[Lev03] L. A. Levin. The tale of one-way functions. *Problems of Information Transmission*, 39(1):92–103, January 2003.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de/