On Pseudorandomness with respect to Deterministic Observers

Oded Goldreich*
Department of Computer Science
Weizmann Institute of Science
Rehovot, Israel.
oded@wisdom.weizmann.ac.il

Avi Wigderson
The Hebrew University, Jerusalem
and
The Institute for Advanced Study, Princeton
avi@math.ias.edu

May 4, 2000

Abstract

In the theory of pseudorandomness, potential (uniform) observers are modeled as *probabilistic* polynomial-time machines. In fact many of the central results in that theory are proven via probabilistic polynomial-time reductions. In this paper we show that analogous deterministic reductions are unlikely to hold. We conclude that randomness of the observer is essential to the theory of pseudorandomness.

What we actually prove is that the hypotheses of two central theorems (in the theory of pseudorandomness) hold unconditionally when stated with respect to *deterministic* polynomial-time algorithms. Thus, if these theorems were true for deterministic observers, then their conclusions would hold unconditionally, which we consider unlikely. For example, it would imply (unconditionally) that any unary language in $\mathcal{BPP}$ is in $\mathcal{P}$.

The results are proven using diagonalization and pairwise independent sample spaces.

Keywords: Pseudorandomness, Computational Difficulty, Derandomization, Unary Languages, Diagonalization, Pairwise Independent Sample Spaces.

*Supported by MINERVA Foundation, Germany.

Introduction

The theory of pseudorandomness, initiated by Blum, Goldwasser, Micali, and Yao [6, 2, 12], is one of the fundamental achievements of complexity theory. The pivot of this approach is the suggestion to view objects as equal if they cannot be told apart by any *efficient procedure* (i.e., polynomial-time ones). In particular, a distribution that cannot be *efficiently distinguished* from the uniform distribution will be considered as being "random" (or rather "random for all practical purposes", or "pseudorandom"). Some fundamental results in this theory are sufficient conditions for effective pseudorandomness, discussed below.

We probe the meaning of "efficient procedures" in the definition of allowed observers; specifically, whether the observers are allowed to be probabilistic or not. In the case observers are nonuniform circuits, the distinction makes no difference. However, when observers are uniform machines, only the (natural) case in which they are *probabilistic* polynomial time machines was considered so far. A natural question that arises is why not to identify efficient procedures with *deterministic* polynomial-time algorithms.

Putting aside the philosophical (or conceptual) question of which choice is "correct", one may ask whether the choice is important from a technical point of view. That is, do the known results, which are stated in terms of *probabilistic* observers, hold also with respect to *deterministic* observers. We stress that the known results are of the form *if efficient procedures cannot do A then efficient procedures cannot do B*, and they hold when efficient procedures are identified with *probabilistic* polynomial-time algorithms. We ask whether analogous results hold when efficient procedures are identified with *deterministic* polynomial-time algorithms.

Two such results (for probabilistic observers) that we focus on, are stated informally below.

Thm. A: unpredictability implies pseudorandomness [12][1]. *If each next-bit in a distribution is unpredictable by efficient procedures then the distribution is pseudorandom* (i.e., indistinguishable from random by efficient procedures).

Thm. B: hardness implies pseudorandomness [8][2]. *If there exists an exponential-time computable predicate that is in-approximable by efficient procedures then every language in $\mathcal{BPP}$ has a deterministic sub-exponential simulation that is correct "on average".*

The current proofs of these results make essential use of randomized reductions, and thus fail to establish analoguous statements for *deterministic* observers.

In this paper we provide some evidence towards the essential role of randomness in these proofs by establishing, (without use of any intractability assumptions), the hypotheses of these two theorems. Loosely speaking, without using any intractability assumption, we

1. construct distributions with polynomial-size support that are (next-bit) unpredictability by deterministic polynomial-time algorithms;

2. present an exponential-time computable predicate that is in-approximable by determinstic polynomial-time algorithms.

[1] The theorem does not appear in [12], but was stated by Yao in oral presentations of his work.
[2] We refer to a specific incarnation of the general paradigm (cf., for example, [2, 12, 9]).

Thus, we establish the hypotheses of Thm. A and B for the case in which efficient is understood as *deterministic* polynomial-time. If the analogous theorems would hold in such a case then we would obtain (without use of any intractability assumptions) non-trivial pseudorandom generators and derandomization results. For example, as the reader can verify, it would follow that every unary language in $\mathcal{BPP}$ has a non-trivial derandomization[3]. We mention that $\mathcal{BPP}$ contains some unary languages that are not known to be in $\mathcal{P}$: for example, consider the set $\{1^n : \mathrm{PrimePar}(n)\}$, where $\mathrm{PrimePar}(n) = 1$ if the number of primes in the interval $[2^n, 2^n + n^3]$ is odd (and $\mathrm{PrimePar}(n) = 0$ otherwise).[4]

Caveat: In both cases we allow the construct (distribution or predicate) to be more complex than the adversary. This is in agreement with the approach of Nisan and Wigderson [9], but not with the approach of Blum, Micali and Yao [2, 12]. Recall that the former approach suffices for derandomization of probabilistic complexity classes, which is our main focus here.

Comment: Derandomization of randomized complexity classes is typically acheived by pseudorandomness w.r.t nonuniform circuits (cf., for example, [12, 9, 7]). In such a case one can (non-trivially) simulate the randomized algorithm by a deterministic one so that the latter yields the correct output on every input. When using pseudorandomness w.r.t uniform probabilistic machines, the resulting simulation is only correct "on the average" (cf., for example, [4, Prop. 3.3] and [8]). Specifically, no probabilistic polynomial-time algorithm may output (with non-negligible probability) an instance on which the simulation fails.[5] In our context, when using pseudorandomness w.r.t uniform deterministic machines, the resulting simulation is only correct in the sense that no deterministic polynomial-time algorithm may output an instance on which the simulation fails. Thus, for unary languages, the simulation is always correct.

Techniques: The results are proven using diagonalization and pairwise independent sample spaces. The same technique was used (for a similar purpose) in [8].

1 Hard predicates

For technical reasons we start with the result mentioned second in the introduction.

Theorem 1 *There exists an exponential-time computable predicate P so that for every deterministic polynomial-time machine M and all sufficiently large n's, it holds that*

$$\Pr[M(U_n) = P(U_n)] < \frac{1}{2} + 2^{-n/3}$$

[3] Indeed, this paper came about when naively trying to prove $\mathcal{UBPP} = \mathcal{P}$ by establishing (1) above, and attempting to use the deterministic analog of Theorem A.

[4] Recall the famous Number Theoretic conjecture by which, for all large x, there exists a prime in the interval $[x, x + O(\log^2 x)]$. Thus, it seems appealing to conjecture that for infinitely many n's $\mathrm{PrimePar}(n) = 1$ (resp., $\mathrm{PrimePar}(n) = 0$). Deciding membership in the set $\{1^n : \mathrm{PrimePar}(n)\}$ is Cook-reducible to testing primality, and thus the former set is in $\mathcal{ZPP} \subseteq \mathcal{BPP}$ (see [1, 5, 10, 11]). On the other hand, we know of no other way of deciding membership in the former set.

[5] Here and below, the machine trying to fail the simulator is given 1^n as input and is required to generate an instance of length n.

where U_n denotes the uniform distribution over $\{0,1\}^n$.

Recall that such a result for probabilistic polynomial-time machines would imply that $\mathcal{BPP}$ can be simulated, correctly on the average, in deterministic sub-exponential time [8].

Proof: We actually prove a stronger result: we construct a predicate that is inapproximable even by exponential-time machines, where this exponent is smaller than the one allowed to evaluate of the predicate. Specifically, for inputs of length n, the predicate will be inapproximable in time 2^n but computable in time $2^{O(n)}$.

Consider an enumeration of (modified)[6] deterministic machines running within time 2^n on input of length n, and suppose that for input length n we wish to fool $m = m(n) < 2^{n/3}$ machines upto error $\epsilon = 2^{-n/3}$. (As usual (when applying idagonalization), it suffices to have $m : \mathsf{N} \to \mathsf{N}$ be an arbitrary unbounded function; the condition $m(n) < 2^{n/3}$ will be used at the very end of our proof.)

Let M be one of the abovementioned m machines, and $N = 2^n$. Consider the vector $v_M \in \{0,1\}^N$ representing the output of M on each of the possible n-bit long strings; that is, $v_M[x] \stackrel{\text{def}}{=} M(x)$, for $x \in \{0,1\}^n$. Our aim is to find a vector u that has low correlation with all the v_M's. That is, find $u \in \{0,1\}^N$ so that, for all M's,

$$\Pr[v_M[U_n] = u[U_n]] < \frac{1}{2} + 2^{-n/3} \tag{1}$$

Once such a u is found, we set $P(x) \stackrel{\text{def}}{=} u[x]$, and the theorem follows.

We will select u from a small sample space of pairwise-independent N-bit sequences. We use the fact that there exists such sample spaces of size N^2 that can be constructed in poly(N)-time (see [3]). Going over all elements of the sample space, we may check whether each element u satisfies Eq. (1), for each of the m machines, within poly(N)-time.[7] Thus, it is only left to show that a pairwise-independent N-bit long sequence satisfies Eq. (1) with probability strictly greater than $1 - m^{-1}$.

Claim: Let $v \in \{0,1\}^N$ be arbitrary, and u be a sequence of N uniformly-distributed pairwise-independent bits. Then the probability that u and v agree on more than $(0.5+\epsilon)\cdot N$ entries is strictly less than $\frac{1}{\epsilon^2 N}$.

Proof: For each $x \in \{0,1\}^n$, we define a 0-1 random variable η_x so that $\eta_x \stackrel{\text{def}}{=} 1$ if $v[x] = u[x]$ and $\eta_x \stackrel{\text{def}}{=} 0$ otherwise. Since v is fixed and u is a sequence of $N = 2^n$ uniformly-distributed pairwise-independent bits, it follows that the η_x's are pairwise-independent and $\mathrm{E}(\eta_x) = 0.5$ for each η_x. Using Chebyshev's Inequality, we have

$$\begin{aligned} \Pr\left[\left|\sum_{x\in\{0,1\}^n} \eta_x - 0.5\cdot N\right| \;\geq\; \epsilon\cdot N\right] &\leq \frac{\mathrm{Var}(\sum_x \eta_x)}{(\epsilon N)^2} \\ &< \frac{1}{\epsilon^2 N} \end{aligned}$$

[6]Recall that one cannot effectively enumerate machines running within some given time bound. Yet, one can enumerate all machines, and modify each machine in the enumeration so that the running-time of the modified machine respects the given time bound and so that the modified machine maintains the functionality of the original one in case the original respects the time bound. This is done by incorporating a time-out mechanism.

[7]Recall that the running time of each machine is bounded by $2^n = N$.

and the claim follows. $\square$

Using $\epsilon^2 N = 2^{n/3} > m$ in the above claim, we conclude that a pairwise-independent sample space contains an N-bit long sequence u that satisfies Eq. (1) for each of the m machines, and the theorem follows. $\blacksquare$

2 Next-bit unpredicability

We say that the set $S = \cup_{n \in \mathbb{N}} S_n$, where $S_n \subset \{0,1\}^n$, is polynomial-time constructible if there exists a polynomial-time algorithm that on input 1^n outputs the list of all strings in S_n.

Theorem 2 *For every polynomial p, there exists a polynomial-time constructible set $S = \cup_{n \in \mathbb{N}} S_n$ so that for every deterministic algorithm of running-time p, the following holds: on input 1^n and a prefix of a string uniformly selected in S_n, the algorithm predicts the next bit in the string with probability at most $\frac{1}{2} + \frac{1}{p(n)}$.*

Recall that such a result for probabilistic polynomial-time machines would imply a deterministic polynomial-time program that, on input 1^n and a logarithmically long random seed, produces n-bit sequences that cannot be distinguished from random ones by probabilistic machines running in time $p(n)$. This would in turn imply that $\mathcal{BPP}$ can be simulated, correctly on the average, in deterministic polynomial-time.

Proof: Consider an enumeration of (modified) deterministic machines running within time $p(n)$ on input of length n, and suppose that we wish to fool $m = m(n) < p(n)$ machines from guessing the next bit better than $0.5 + \epsilon$, where $\epsilon = 1/p(n)$. Let M be one of the machines we wish to fool.

We construct S_n in (roughly) n iterations, so that in iteration i we construct $S_{n,i} \subseteq \{0,1\}^i$. We start with $S_{n,\ell} = \{0,1\}^\ell$, where $\ell = 2\log_2(m/\epsilon)$ and $L = 2^\ell$. In the $i+1^{\text{st}}$ iteration, we consider the vector $v_M \in \{0,1\}^L$ representing the output of M on each of the L possible i-bit long strings; that is, $v_M[j] = M[x^{(j)}]$, where $x^{(j)}$ is the j^{th} string in $S_{n,i} \subseteq \{0,1\}^i$. (This represents M's guess of the $i+1^{\text{st}}$ bit of a string uniformly selected in S_n, based on the i-bit long prefix of that string.) Our aim is to find a vector $u \in \{0,1\}^L$ that has low correlation with all the v_M's (i.e., u agrees with each v_M on at most an $0.5 + \epsilon$ fraction of the entries). Once such a vector u is found, we extend $S_{n,i}$ into $S_{n,i+1}$ in the natural manner; that is,

$$S_{n,i+1} \stackrel{\text{def}}{=} \{x^{(j)}u[j] : \text{ where } x^{(j)} \text{ is the } j^{\text{th}} \text{ string in } S_{n,i}\} \subset \{0,1\}^{i+1} \tag{2}$$

It follows that $S_n \stackrel{\text{def}}{=} S_{n,n}$ satisfies the unpredictability requirement; that is, for each of the above M's, the probability that M correctly predicts the next bit in a sequence uniformly selected in S_n is at most $0.5 + \epsilon$.

It remains to specify how a suitable vector $u \in \{0,1\}^L$ is found, in each iteration. This is done analogously to the way described in the proof of Theorem 1. We stress two points:

1. To prove an analogous claim regarding the density of suitable vectors u in the sample space, we need $\epsilon^2 L > m$ to hold. This is indeed the case, by our choice of $L = 2^\ell = 2^{2\log_2(m/\epsilon)} = \frac{m^2}{\epsilon^2}$.

2. The vector u can be found in time $\mathrm{poly}(L) \cdot m \cdot p(n)$, where the first factor accounts for the construction and the size of the sample space, and the last for the running time of each machine M. By our choice of parameters, $\mathrm{poly}(L) \cdot m \cdot p(n)$ is a polynomial in $p(n)$, and so is polynomial in n.

Thus, our entire construction operates in time polynomial in n, and the theorem follows. ■

3 Open Problems

In contradiction to the feelings expressed above, it may be that the distribution ensemble constructed in the proof of Theorem 2 has stronger pseudorandom features than stated. In particular, we propose the following open problem.

Open Problem 3 *Is the distribution ensemble constructed in the proof of Theorem 2 pseudorandom with respect to deterministic algorithms running in time p. That is, is it the case that for every deterministic algorithm A of running-time p, and for all sufficiently large n's it holds that*

$$|\Pr[A(U_n) = 1] - \Pr[A(X_n) = 1]| < \frac{1}{p(n)}$$

where U_n and X_n denote the uniform distribution over $\{0,1\}^n$ and S_n, respectively.

Indeed, an affirmative answer to this question will be a of great interest, but an negative answer will be interesting too. In the latter vein it may be even interesting to show that ensemble constructed in the proof of Theorem 2 is easy to predict via probabilistic algorithms running in time p. Indeed, it would be more interesting (to say the very least) to prove the opposite, but we don't expect this to be doable.[8]

Acknowledgements

We thank Luca Trevisan for a helpful discussion.

References

[1] L.M. Adleman and M. Huang. *Primality Testing and Abelian Varieties Over Finite Fields.* Springer-Verlag LNCS (Vol. 1512), 1992.

[2] M. Blum and S. Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SIAM J. on Comput.*, Vol. 13, pages 850–864, 1984.

[3] B. Chor and O. Goldreich. On the Power of Two–Point Based Sampling. *Jour. of Complexity*, Vol 5, 1989, pages 96–106.

[8]Recall that the ensemble is unpredictable via probabilistic polynomial-time algorithms iff it is pseudorandom with respect to such algorithms.

[4] O. Goldreich. *Modern Cryptography, Probabilistic Proofs and Pseudorandomness. Algorithms and Combinatorics series* (*Vol. 17*), *Springer*, 1999.

[5] S. Goldwasser and J. Kilian. Primality Testing Using Elliptic Curves. *J. of the ACM*, Vol. 46, pages 450–472, 1999.

[6] S. Goldwasser and S. Micali. Probabilistic Encryption. *J. of Comp. and Sys. Sci.*, Vol. 28, No. 2, pages 270–299, 1984.

[7] R. Impagliazzo and A. Wigderson. P=BPP if E requires exponential circuits: Derandomizing the XOR Lemma. In *29th STOC*, pages 220–229, 1997.

[8] R. Impagliazzo and A. Wigderson. Randomness vs. Time: De-randomization under a uniform assumption. In *39th FOCS*, pages 734–743, 1998.

[9] N. Nisan and A. Wigderson. Hardness vs Randomness. *J. of Comp. and Sys. Sci.*, Vol. 49, No. 2, pages 149–167, 1994.

[10] M.O. Rabin. Probabilistic Algorithm for Testing Primality. *Journal of Number Theory*, Vol. 12, pages 128–138, 1980.

[11] R. Solovay and V. Strassen. A Fast Monte-Carlo Test for Primality. *SIAM J. on Comput.*, Vol. 6, pages 84–85, 1977. Addendum in *SIAM J. on Comput.*, Vol. 7, page 118, 1978.

[12] A.C. Yao. Theory and Application of Trapdoor Functions. In *23rd FOCS*, pages 80–91, 1982.