# Hardness and Hierarchy Theorems for Probabilistic Quasi-polynomial Time

JIN-YI CAI * AJAY P. NERURKAR † D. SIVAKUMAR ‡

(extended abstract)

**Abstract**

We prove tight hierarchy theorems for bounded error probabilistic quasi-polynomial time classes, under several hardness assumptions. We show that assuming either (1) the Permanent does not have a subexponential time BP algorithm, or (2) some function in EXPTIME does not have subexponential size circuits, then for every $1 \leq \alpha < \beta$,

$$\mathrm{BPTIME}(2^{(\log n)^{\alpha}}) \subsetneq \mathrm{BPTIME}(2^{(\log n)^{\beta}}).$$


*Department of Computer Science, State University of New York at Buffalo, Buffalo, NY 14260. Email: cai@cs.buffalo.edu. Research supported in part by NSF grant CCR-9634665 and a J. S. Guggenheim Fellowship.

†Department of Computer Science, State University of New York at Buffalo, Buffalo, NY 14260. Email: apn@cs.buffalo.edu. Research supported in part by NSF grant CCR-9634665.

‡Department of Computer Science, University of Houston, Houston, TX 77204. Email: siva@cs.uh.edu. Research supported in part by NSF CAREER award CCR-9734164.

# 1 Introduction

It has been observed for some time now that the hardness of a function, far from being something undesirable, could actually be very useful. Early work on this idea [BM84, Yao82] dealt with the hardness of inverting one-way functions. Nisan and Wigderson [NW94] constructed a pseudo-random generator which was based on the hardness of approximating any given boolean function. They showed that if there is a boolean function $f$ in EXP such that any polynomial-size circuit errs in computing $f$ on at least an inverse polynomial fraction of inputs, then BPP has a simulation in $\bigcap_{\epsilon>0} \mathrm{DTIME}(2^{n^\epsilon})$ which works for an infinite sequence of input lengths. This was strengthened by Babai *et al.* [BFNW93], who achieved the same result assuming only that there is a function in EXP that does not have polynomial-size circuits.

In a recent paper [IW98], Impagliazzo and Wigderson were able to derandomize BPP algorithms under a uniform assumption for the first time. They use the random self-reducibility and the downward self-reducibility of the Permanent function in conjunction with a result of Karp and Lipton [KL80] and Toda's theorem [Tod91] to show the following result: if EXP $\neq$ BPP, then every language in BPP has a simulation that works correctly on all but an inverse polynomial fraction of inputs of length $n$, for infinitely many input lengths $n$.

In this paper, we show that hardness of functions can be used to prove a separation result for probabilistic quasi-polynomial time, $\mathrm{BPQP} = \mathrm{BPTIME}(2^{(\log n)^{O(1)}})$. Specifically, we prove that BPQP has a tight hierarchy, if either the Permanent cannot be computed in sub-exponential probabilistic time or there is a function in EXP that cannot be computed by sub-exponential size circuits.

Hierarchy theorems have a long history in complexity theory. Some of the first results in the field were of this kind. For deterministic time, Hartmanis and Stearns [HS65] proved that for any time-constructible functions $t(n)$ and $T(n)$, with $t(n)^2 = o(T(n)), \mathrm{DTIME}(t(n))$ is properly contained in $\mathrm{DTIME}(T(n))$. Later, Hennie and Stearns [HS66] proved the same result under the weaker assumption, $t(n)\log t(n) = o(T(n))$. A hierarchy theorem for non-deterministic time was proved by Cook [Coo73] and improved by Seiferas *et al.* [SFM78]. They separate $\mathrm{NTIME}(t(n))$ and $\mathrm{NTIME}(T(n))$ when $t, T$ are time-constructible and $t(n+1) = o(T(n))$.

Unlike the cases of deterministic time and non-deterministic time, the existence of tight hierarchies for probabilistic time classes BPTIME is a major open question in complexity theory; in fact, this question remains open even in the presence of oracles. An oracle result showing a collapse of BPP to $\mathrm{BPTIME}(O(n))$ was claimed in [FS89], but was retracted in [FS97]. There have been since some further claims of oracle results of a similar nature, but it is still not clear whether the claimed proof works. Berg and Håstad [BH97] discuss further the difficulties and pitfalls of this problem, even for relativized results. One of the very few results known in this area is due to Karpinski and Verbeek [KV87], who showed, for example, that for every $\epsilon, c > 0$, $\mathrm{BPTIME}(2^{(\log n)^c})$ is strictly contained in $\mathrm{BPTIME}(2^{n^\epsilon})$. Allender *et al.* [ABHH93] obtain a mild strengthening of their result as follows: For time-constructible functions $T, t$ with $t^{-1}$ computable in linear time, if for some constant $k$, $(T \cdot t^{-1})^{(k)}(n) \neq 2^{O(n)}$, then there is a set in $\mathrm{BPTIME}(T(n))$ such that any BP machine that correctly decides this set takes more than $t(n)$ time for all but finitely many inputs. We note that the gap between $t$ and $T$ is still essentially exponential. This is the best known unconditional result about a hierarchy for BPTIME classes.

In sharp contrast, not only are tight hierarchies known for deterministic and non-deterministic time, but also substantially stronger versions of them. For example, Geske *et al.* [GHS91] showed, under the same assumptions as in [HS66], that there is a set in $\mathrm{DTIME}(T(n))$ such that any deterministic machine that correctly decides this set takes more than $t(n)$ time for all but finitely many inputs. Such results are called *almost-everywhere hierarchy* theorems, or a.e. hierarchy theorems. (The theorems of [HS65, HS66, Coo73, SFM78] are of the i.o. (infinitely often) kind, that is, they only assert the existence of sets in $T(n)$-time for which any machine requires more than $t(n)$-time on *infinitely many* inputs.) An a.e. hierarchy theorem for non-deterministic time was proved by Allender *et al.* [ABHH93], who also proved their result for BPTIME in the a.e. setting as well.

To prove our hierarchy theorems we first show that languages in BPQP $=$ BPTIME$(2^{(\log n)^{O(1)}})$ can be simulated in DTIME$(2^{(\log n)^{O(1)}})$, in a certain technical sense, under either of the hardness assumptions. We do so by employing techniques from [Yao82, NW94, IW98, IW97] to construct a pseudo-random generator which is used to perform the deterministic simulation. We then use a tight hierarchy-type argument (similar to the a.e. kind) for deterministic time to show the existence of a tight hierarchy for BPQP.

This paper is organized as follows. In Section 2 we give some preliminary definitions. In Section 3 we state our main technical theorems and apply them to prove the hierarchy theorems. Section 4 contains the proof of the first technical theorem, which uses a uniform hardness assumption on the Permanent function to deterministically simulate BPQP. Section 5 has the proof of our second technical theorem which uses a non-uniform hardness assumption on EXP.

## 2 Preliminaries

In this section, we define a version of the Permanent function that is still hard for #P, and some other functions that amplify its computational hardness.

We first define an encoding for elements of the field $Z_p$ for an odd prime $p$. Any $p$-bit string $w$ is considered to represent the element ($w \mod p$) of $Z_p$. For purposes of uniformity, we disallow the largest $2^p - \lfloor \frac{2^p}{p} \rfloor p$ strings as valid representations. Therefore, every element of $Z_p$ has exactly $\lfloor \frac{2^p}{p} \rfloor$ representations. Note that the fraction of $p$-bit strings that are not valid representations is $< \frac{p}{2^p}$.

Let $M$ be a $t \times t$ matrix over $Z_p$, where $p \geq t+2$, and $s \in Z_p$. Let $n = 2pt^2$. A string $x$ of length $n$ encodes the pair $(M, s)$ if the first $p(t^2+1)$ bits of $x$ represent the $t^2$ entries of $M$ and $s$ (with redundancy) as defined earlier. Note that the fraction of $n$-bit vectors that are not the encoding of any $(M, s)$ pair is $< \frac{p(t^2+1)}{2^p} < \frac{n}{2^{n^{1/4}}}$.

Because $n = 2pt^2$ and $p$ is the only odd prime that occurs an odd number of times in the prime factorization of $n$, given a string $x$ of length $n$, it is easy to recover the $M$ and $s$ that $x$ encodes. Also, all strings of the same length correspond to the *same* $p$ and $t$. The language *Perm* is defined as follows:

$$Perm = \{x \mid |x| = 2pt^2, \text{ for some } p \geq t+2, x \text{ encodes } M \text{ and } s, M \in Z_p^{t\times t}, \mathrm{per}(M) \bmod p = s\},$$

In this paper, we always use $n$ to denote the length of an input to *Perm*. The function obtained by evaluating *Perm* on many inputs is called $g$. Formally, $g$ takes $k$ arguments, $x_1, \ldots, x_k$ and $g(x_1, \ldots, x_k)$ is defined as the tuple $(Perm(x_1), \ldots, Perm(x_k))$. In this paper, we always use $k$ to denote the number of arguments to $g$. The value of $k$ will be $o(n^5)$. The tuple $(x_1, \ldots, x_k)$ is denoted $y$. Clearly, $|y| = nk$ and $|g(y)| = k$. Let $h$ be the Goldreich-Levin predicate [GL89] for $g$, i.e., $h(y, r) = \langle g(y), r \rangle$, where $\langle ., . \rangle$ denotes inner product mod 2. The lengths of the two inputs to $h$ are $nk$ and $k$ respectively.

All these functions will be indexed by $n$. $Perm_n$ denotes the "slice" of *Perm* that takes inputs of length $n$, $g_n$ denotes the slice of $g$ that takes $k$ inputs of $n$ bits each ($k$ depends on $n$ in a manner to be specified later), and $h_n$ is the slice of $h$ where $|y| = nk$ and $|r| = k$.

# 3 Main Results

In this section, we establish the connection between hardness assumptions and tight hierarchy theorems for BPQP $=$ BPTIME$(2^{(\log n)^{O(1)}})$. We do this by proving two intermediate theorems, which we will call our main technical theorems.

Our first technical theorem shows that if *Perm* $\notin$ BPTIME$(2^{n^{\epsilon_0}})$ for some $\epsilon_0 > 0$, then for every language $L$ in BPQP, there is an algorithm that runs in deterministic time $2^{(\log n)^{O(1)}}$ that solves $L$ correctly on almost all instances of infinitely many input lengths. Note that we use the letter $d$ to denote lengths of inputs to $L$.

**Theorem 1** *For all constants $c \geq 1, c_1 > 0$ and all $\epsilon_0$, where $0 < \epsilon_0 < 1$, there exists $c' \geq c$ such that if Perm cannot be decided in* BPTIME$(2^{n^{\epsilon_0}})$*, then every $L \in$* BPTIME$(2^{(\log d)^c})$ *has a deterministic simulation in time $2^{(\log d)^{c'}}$, such that for infinitely many lengths $d$, the deterministic simulation is correct for at least a $1 - \frac{1}{2^{(\log d)^{c_1}}}$ fraction of $\Sigma^d$.*

We call such a simulation an io-deterministic simulation. Theorem 1 is proved in Section 4. Below we use it to obtain a hierarchy theorem for BPQP. We begin with the following lemma.

**Lemma 1** *Let $1 \leq \alpha < \beta$ be such that* BPTIME$(2^{(\log d)^\alpha}) =$ BPTIME$(2^{(\log d)^\beta})$*. Then* BPQP $=$ BPTIME$(2^{(\log d)^\alpha})$.

**Proof.** Let $L \in$ BPTIME$(2^{(\log d)^{\beta^2/\alpha}})$. Define $L' = \{x0^{d'-d} \mid x \in L\}$, where $d = |x|$ and $d' = 2^{(\log d)^{\beta/\alpha}}$. Because $\alpha \geq 1$, $L$ reduces to $L'$ in deterministic time $O(2^{(\log d)^\beta})$. Also, since $2^{(\log d')^\beta} = 2^{(\log d)^{\beta^2/\alpha}}$, $L' \in$ BPTIME$(2^{(\log d')^\beta}) =$ BPTIME$(2^{(\log d')^\alpha})$. But, since $2^{(\log d')^\alpha} = 2^{(\log d)^\beta}$ we have $L \in$ BPTIME$(2^{(\log d)^\beta})$. Thus BPTIME$(2^{(\log d)^{\beta^2/\alpha}}) =$ BPTIME$(2^{(\log d)^\alpha})$. This argument can be extended indefinitely, so we obtain BPQP $=$ BPTIME$(2^{(\log d)^\alpha})$. $\square$

**Lemma 2** *If $\forall c \geq 1,\ c_1 > 0\ \exists c' \geq c$ such that every $L \in$* BPTIME$(2^{(\log d)^c})$ *has a deterministic simulation in time $2^{(\log d)^{c'}}$ for infinitely many lengths $d$ and for all such $d$, the deterministic simulation is correct for at least a $1 - \frac{1}{2^{(\log d)^{c_1}}}$ fraction of $\Sigma^d$, then* BPQP *has a hierarchy, that is, $\forall \alpha, \beta, 1 \leq \alpha < \beta$,*

$$\text{BPTIME}(2^{(\log d)^\alpha}) \subsetneq \text{BPTIME}(2^{(\log d)^\beta}).$$

**Proof.** If not, then by Lemma 1, BPQP $=$ BPTIME$(2^{(\log d)^\alpha})$, for some $\alpha \geq 1$. By hypothesis any language in BPTIME$(2^{(\log d)^\alpha})$ has an io-deterministic simulation in DTIME$(2^{(\log d)^{\alpha'}})$ for some $\alpha' \geq \alpha$. Let $\alpha_1 > \alpha'$. We will exhibit a language $L \in$ BPTIME$(2^{(\log d)^{\alpha_1}}) \subseteq$ BPQP $=$ BPTIME$(2^{(\log d)^\alpha})$ which has no io-deterministic simulation in time DTIME$(2^{(\log d)^{\alpha'}})$.

Let $T_1(d) = 2^{(\log d)^{\alpha_1}}$ and $T_2(d) = 2^{(\log d)^{\alpha'}}$. The $2^d$ strings of length $d$ are divided in lex order into $\log d$ segments of equal size. Here is a deterministic machine running in time $T_1(d)$ that diagonalizes against $T_2(d)$ time machines.

On input $x$ of length $d$, $D$ calculates which segment of $\Sigma^d$ $x$ lies in. Let $x$ be in the $i$-th segment. Then, if $i$ is a valid code of a one-tape TM $M_i$, $D$ simulates $M_i$ on $x$ for $\frac{T_1(d)}{d}$ steps. If the simulation finishes in so many steps, $D$ accepts iff $M_i$ rejects $x$. If $i$ is not a valid code of such a TM or if $M_i$ does not finish its computation in $\frac{T_1(d)}{d}$ steps, $D$ rejects $x$. Clearly, $D$ runs in time at most $T_1(d)$.

Let $M$ be a machine running in DTIME$(T_2(d))$. Let $M'$ be a one-tape machine that accepts the same language as $M$ and runs in time $O(T_2^2(d))$. For all large enough $d$, $M'$ runs in time $< \frac{T_1(d)}{d}$. Therefore, for all large $d$, $D$ will simulate $M'$ (to completion) on a fraction $\frac{1}{\log d}$ of the inputs and accept iff $M'$ rejects. Define $L = L(D)$. Then, $L \in \mathrm{DTIME}(2^{(\log d)^{\alpha_1}})$ and no machine in $\mathrm{DTIME}(2^{(\log d)^{\alpha'}})$ can simulate $L$ on more than a $1 - \frac{1}{\log d}$ ($< 1 - \frac{1}{2^{(\log d)^{c_1}}}$) fraction of $\Sigma^d$ for infinitely many $d$. This is a contradiction. $\square$

From Theorem 1 and Lemma 2 we get,

**Theorem 2 (Hierarchy Theorem from Hardness of Permanent)** *If Perm* $\notin \mathrm{BPTIME}(2^{n^{\epsilon_0}})$ *for some* $\epsilon_0 > 0$*, then* BPQP *has a tight hierarchy, that is, for all* $\alpha, \beta$, $1 \leq \alpha < \beta$,

$$\mathrm{BPTIME}(2^{(\log d)^{\alpha}}) \subsetneq \mathrm{BPTIME}(2^{(\log d)^{\beta}}).$$

Our second technical theorem shows that if exponential time functions cannot be computed by sub-exponential size circuits, then for every language $L$ in BPQP, there is an algorithm that runs in deterministic time $2^{(\log n)^{O(1)}}$ that solves $L$ correctly on infinitely many input lengths.

**Theorem 3** *If there is a language in deterministic exponential time* $\mathrm{EXP} = \mathrm{DTIME}(2^{n^{O(1)}})$ *that cannot be computed by circuits of size* $2^{n^{\epsilon_1}}$ *for some* $\epsilon_1 > 0$*, then for every* $c \geq 1$*, there is a* $c' \geq c$ *such that every language* $L \in \mathrm{BPTIME}(2^{(\log d)^{c}})$ *can be deterministically simulated in time* $2^{(\log d)^{c'}}$ *for infinitely many input lengths* $d$.

We prove Theorem 3 in Section 5. From Theorem 3 and Lemma 2 we get,

**Theorem 4 (Hierarchy Theorem from Hardness of EXP)** *If* EXP *does not have circuits of size* $2^{n^{\epsilon_1}}$ *for some* $\epsilon_1 > 0$*, then* BPQP *has a tight hierarchy, that is, for all* $1 \leq \alpha < \beta$,

$$\mathrm{BPTIME}(2^{(\log d)^{\alpha}}) \subsetneq \mathrm{BPTIME}(2^{(\log d)^{\beta}}).$$

## 4 Derandomizing BPQP if *Perm* is Hard

This section is devoted to the proof of Theorem 1. Let $c, c_1, \epsilon_0$ be given as in Theorem 1, and $L \in \mathrm{BPTIME}(2^{(\log d)^c})$. We will follow the construction of [NW94] to obtain a pseudo-random generator $G$ which will be used to simulate $L$ deterministically.

THE GENERATOR. Let $m = (\log d)^{c_2}$, where $c_2 \geq \max(\frac{54c_1}{\epsilon_0}, \frac{150c}{\epsilon_0}, \frac{c}{2})$, and $c, c_1, \epsilon_0$ are as given in Theorem 1. Let $l = m^2$. Choose an $(l, m, (\log d)^c, 2^{(\log d)^c})$-design, as in [NW94]. This is a $2^{(\log d)^c}$ by $l$ 0-1 matrix over a field of size $m$. (We assume that $m$ is a prime power. If not, we can choose the smallest power of 2 bigger than it. This will only introduce an additional constant factor into our analysis.) Each column of the design is indexed by a pair of field elements and each row is indexed by a polynomial of degree $(\log d)^c$. The entry $(f, (a, b))$ is 1 iff $f(a) = b$. The generator, $G$, is a function that maps $l$ bits to $2^{(\log d)^c}$ bits. The $i^{\text{th}}$ bit of $G(x)$, where $|x| = l$, is computed by evaluating $h$ on the $m$ bits of $x$ that correspond to the positions in which the $i^{\text{th}}$ row of the design has a 1. $G_n$ is the restriction of $G$ where $m = nk + k$.

THE SIMULATION. Let $M$ be a $\mathrm{BPTIME}(2^{(\log d)^c})$ machine for $L$. Because $2c_2 \geq c$, $M$ can be simulated in deterministic time $O(2^{2^{(\log d)^{2c_2}}})$ by running through all the $l$-bit seeds of $G$, running $M$ with each of the outputs of $G$ as the source of randomness and taking the majority outcome.

Assume that for all but finitely many $d$, the simulation fails on a fraction of $z \in \Sigma^d$ that is $\geq \frac{1}{2^{(\log d)^{c_1}}}$. We will show how to decide *Perm* in $\mathrm{BPTIME}(2^{n^{\epsilon_0}})$, for all *large enough* $n$, a contradiction. We will do so by constructing probabilistically, in time $O(2^{n^{3\epsilon_0/4}})$, a deterministic circuit $C_n$ of size $O(2^{n^{\epsilon_0/9}})$ to solve *Perm* on inputs of length $n$. The construction is inductive.

That is, to construct $C_n$, we assume we have circuits $C_1, \ldots, C_{n-1}$ for input lengths $1, \ldots, (n-1)$. If $j$ is not of the form $2pt^2$, $C_j$ is a dummy circuit.

On an input of length $n$, the BPTIME algorithm constructs the circuits, $C_1, \ldots, C_n$ and then simulates $C_n$. The total time required by this process will be at most $nO(2^{n^{3\epsilon_0/4}}) + O(2^{n^{\epsilon_0/9}}) < 2^{n^{\epsilon_0}}$. Since all the constructions are probabilistic, we will have to ensure that all the $C_i$ are correct with a good enough probability (*e.g.* $1 - \frac{1}{n^2}$, which ensures that the total error involved in the construction of the $n$ circuits is at most $\frac{1}{n}$). It will be clear from the proof that this can be done.

Let $d = 2^{(nk+k)^{1/c_2}}$, so that $(nk+k) = (\log d)^{c_2} = m$. A "large enough" $n$ is one that makes this $d$ large enough for the simulation to fail for it. Because $k = o(n^5)$ (to be chosen later), $\log d \leq n^{6/c_2}$. Therefore, $(\log d)^{c_1} \leq n^{\epsilon_0/9}$ and $(\log d)^c \leq n^{\epsilon_0/25}$, by our choice of $c_2$.

The lemmas that follow will show how the circuit $C_n$ can be constructed using circuits for smaller input lengths.

**Lemma 3** *If the above simulation for $L$ fails for a fraction $\geq \frac{1}{2^{(\log d)^{c_1}}}$, then a distinguisher $D_n$ for $G_n$ of size $O(2^{2(\log d)^c})$ can be constructed in time $O(2^{2(\log d)^c})$ such that with probability $\frac{1}{2^{(\log d)^{c_1}}}$, the construction satisfies the following:*

$$\left| \Pr_x[D_n(G_n(x)) = 1] - \Pr_y[D_n(y) = 1] \right| \geq \frac{1}{4}.$$

**Proof.** Sample a point $z$ randomly from $\{0,1\}^d$. With probability $\geq \frac{1}{2^{(\log d)^{c_1}}}$, it is one of the inputs on which the simulation fails. Let $D_n$ be the circuit that represents the computation of $M$ on $z$ with the random bits of $M$ as its input. Wlog, let $z \in L$. Therefore,

$$\Pr_x[D_n(G_n(x)) = 1] \leq \frac{1}{2},$$

and

$$\Pr_y[D_n(y) = 1] \geq \frac{3}{4}.$$

Thus,

$$\left| \Pr_x[D_n(G_n(x)) = 1] - \Pr_y[D_n(y) = 1] \right| \geq \frac{1}{4}.$$

*Time/size analysis for $D_n$*: The time to construct $D_n$ can be seen to be $O(2^{2(\log d)^c})$, because $z$ can be sampled in time $d = 2^{\log d}$ and $c \geq 1$. This is also an upper bound for the size of $D_n$.
*Probability of a good $D_n$*: The distinguishing probability of $D_n$ is $\geq 1/4$ with probability $\frac{1}{2^{(\log d)^{c_1}}}$.
□

**Lemma 4** *Given a distinguisher $D_n$ of size $O(2^{2(\log d)^c})$ for $G_n$ with distinguishing probability $\geq 1/4$, in time $O(2^{2(\log d)^c} \cdot 2^{2n^{\frac{\epsilon_0}{9}}})$ a circuit $E_n$ of size $O(2^{2(\log d)^c})$ to compute $h_n$ can be constructed such that with probability $\geq \frac{1}{4 \cdot 2^{(\log d)^c}}$, the construction satisfies the following:*

$$\Pr_w[E_n(w) = h_n(w)] \geq \frac{1}{2} + \frac{1}{8 \cdot 2^{(\log d)^c}},$$

*where $w$ stands for the pair $(y, r)$ that $h_n$ takes as input.*

The construction exploits the downward self-reducibility of *Perm* as in [IW98], and can be found in the appendix.

□

**Lemma 5** *Given a circuit $E_n$ of size $O(2^{2(\log d)^c})$ that achieves*

$$\Pr_{y,r}[E_n(y,r) = h_n(y,r)] \geq \frac{1}{2} + \frac{1}{8 \cdot 2^{(\log d)^c}},$$

*in time $O(2^{5(\log d)^c})$, a circuit $F_n$ of size $O(2^{5(\log d)^c})$ to compute $g_n$ can be constructed such that with probability $\geq \frac{1}{2^{2(\log d)^{2c}}}$, the construction satisfies the following:*

$$\Pr_{y}[F_n(y) = g_n(y)] \geq \frac{1}{2^{2(\log d)^{2c}}}.$$

**Proof.** This construction is based on the technique of [GL89]. By definition, $h_n(y,r) = \langle g_n(y), r \rangle$, where $|y| = nk$ and $|r| = k$. To probabilistically construct $F_n$,

1. Pick $r_1, \ldots r_s \in_U \{0,1\}^k$, where $s = 3(\log d)^c$.
2. Pick $\sigma_1, \ldots, \sigma_s \in_U \{0,1\}$. $\sigma_i$ is a guess for $\langle g_n(y), r_i \rangle$.
3. For all $S \in \{0,1\}^s, S \neq 0^s$, compute $\tau_S = \bigoplus_{i:S_i=1} \sigma_i$ and for every $j$, $t_S^j$ = the result of flipping the $j^{\text{th}}$ bit of $\bigoplus_{i:S_i=1} r_i$.

All this is stored in $F_n$.

*Working of $F_n$*: $F_n$ computes $g_n(y)$ bit-by-bit. To compute the $j^{\text{th}}$ bit of $g_n(y)$, $F_n$ evaluates $E_n(y, t_S^j) \oplus \tau_S$ for all $S \neq 0^s$ and then takes the majority of the answers.

*Time/size analysis for $F_n$*: Since the size of $E_n$ is $O(2^{2(\log d)^c})$, the time to construct $F_n$ is $O(ks2^s + 2^{2(\log d)^c} 2^s) = O(2^{5(\log d)^c})$. This is also an upper bound for the size of $F_n$.

*Probability of a good $F_n$*: Let $Q_F = \Pr_y[F_n(y) = g_n(y)]$. By hypothesis,

$$\Pr_{y,r}[E_n(y,r) = h_n(y,r)] \geq \frac{1}{2} + \frac{1}{8 \cdot 2^{(\log d)^c}}.$$

We can show that $\text{Exp}_F(Q_F) > \frac{1}{2^{(\log d)^{2c}}}$. Therefore, by applying the Markov inequality, we conclude that $\Pr_F[Q_F > \frac{1}{2^{2(\log d)^{2c}}}] > \frac{1}{2^{2(\log d)^{2c}}}$. Details can be found in the appendix. □

**Lemma 6** *Given a circuit $F_n$ of size $O(2^{5(\log d)^c})$ that achieves*

$$\Pr_{y}[F_n(y) = g_n(y)] \geq \frac{1}{2^{2(\log d)^{2c}}},$$

*in time $O\left(2^{3n^{\epsilon_0/9}}\right)$, a circuit $A_n$ of size $O(2^{3n^{\epsilon_0/10}})$ to compute $Perm_n$ can be constructed such that, with overwhelmingly high probability, the construction satisfies the following:*

$$\Pr_{x}[A_n(x) = Perm_n(x)] \geq 1 - \frac{1}{n^4}.$$

**Proof.** This construction has been adapted from the proof of the following Direct Product Lemma which appeared in [IW97].

**Direct Product Lemma [IW97]** *Let $f : \{0,1\}^n \to \{0,1\}$. $\forall \epsilon, \delta > 0$, there exists $k = O(-\frac{\log \epsilon}{\delta})$ so that, if there is a circuit $C'$ of size $s'$ such that, $\Pr_x(g(x) = C'(x)) > \epsilon$, then there is a circuit $C$ of size $s = \frac{1}{(\epsilon\delta)^{O(1)}} s'$ such that, $\Pr_x(f(x) = C(x)) > 1 - \delta$, where $g(x_1, \ldots, x_k) = (f(x_1), \ldots, f(x_k))$.*

We put $f = Perm_n, C' = F_n, \delta = \frac{1}{n^4}, s' = O(2^{5(\log d)^c})$, and $\epsilon = 2^{-n^{\epsilon_0/10}}$. Since $c_2$ has been chosen large enough compared to $c$, $2^{n^{\epsilon_0/10}} > 2^{2(\log d)^{2c}}$, *i.e.* $\epsilon < \frac{1}{2^{2(\log d)^{2c}}}$. The details of the construction are deferred to the appendix. $\square$

So far, starting from the assumption that the deterministic simulation failed for a sizeable fraction of inputs for all sufficiently large lengths $d$, and assuming the availability of circuits $C_j$ of size $O(2^{n^{\epsilon_0/9}})$ for $j < n$, we have probabilistically constructed a circuit that computes $Perm_n$ on a $1 - \frac{1}{n^4}$ fraction of the inputs. But the probability that such a circuit is obtained is the product of the probabilities for each stage. Here are the success probabilities for the various stages:

1. *Construction of* $D_n$: $\frac{1}{2^{(\log d)^{c_1}}}$
2. *Construction of* $E_n$: $\frac{1}{4 \cdot 2^{(\log d)^c}}$
3. *Construction of* $F_n$: $\frac{1}{2^{2(\log d)^{2c}}}$
4. *Construction of* $A_n$: Exponentially close to 1

Substituting $(\log d)^{c_1} \leq n^{\epsilon_0/9}$ and $(\log d)^c \leq n^{\epsilon_0/25}$ in the probability estimates, we find that the total probability of constructing a good $A_n$, which is a circuit to compute $Perm_n$ on a $1 - \frac{1}{n^4}$ fraction of the inputs, is $> \frac{1}{2^{n^{\epsilon_0/8}}}$. Therefore, if we repeat this process $2^{n^{\epsilon_0/4}}$ times, with very high probability we will get a circuit that is good. But to know which circuit is good, we have to test them. Here is a test that can be shown to work with all but exponentially small probability. Given a circuit $C$, we uniformly sample poly$(n)$ many $x_i$, and run $C$ on each of them. If $C$ computes $Perm_n$ correctly on more than a $1 - \frac{1}{n^3}$ fraction of them, we pass the circuit. To carry out this test, we need to use circuits for permanents of lower order. By using Chernoff bounds, it can be shown that $A_n$, constructed in Lemma 6, passes this test with exponentially high probability and that any circuit that passes this test, computes $Perm_n$ correctly on at least a $1 - \frac{1}{n^2}$ fraction of the inputs. This test takes at most poly$(n)$ $2^{2n^{\epsilon_0/9}}$ time for each circuit tested, a total time of at most poly$(n)$ $2^{2n^{\epsilon_0/9}} 2^{n^{\epsilon_0/4}}$.

**Lemma 7** *Given a circuit $A_n$ of size $O(2^{3n^{\epsilon_0/10}})$ for $Perm_n$ such that,*

$$\Pr_x[A_n(x) = Perm_n(x)] \geq 1 - \frac{1}{n^2},$$

*in time $O(2^{n^{\epsilon_0/9}})$, a circuit $C_n$ of size $O(2^{n^{\epsilon_0/9}})$ can be constructed such that, with overwhelmingly high probability the construction satisfies the following:*

$$\forall x \; C_n(x) = Perm_n(x).$$

**Proof.** Since at most a fraction $\frac{n}{2^{n^{1/4}}}$ of the $n$-bit strings are invalid encodings of a $(M, s)$ pair, the probability that $A_n$ succeeds on a randomly chosen $x$, given that is a valid encoding, is $\geq 1 - \frac{1}{n^2} - \frac{n}{2^{n^{1/4}}} > 1 - \frac{2}{n^2}$.

Here is a probabilistic circuit $C'_n$ which decides $Perm_n$ correctly on every input, with high probability. On input $x \in \{0,1\}^n$, compute the $p$ and $t$ that $n$ corresponds to. If $x$ is not a valid encoding of a pair $(M, s)$, reject $x$. Otherwise, let $(M, s)$ be the pair $x$ encodes. Randomly pick a $t \times t$ matrix $R$ over $Z_p$. This can be done, for example, as follows. To pick $R_{ij}$, uniformly pick a number $r \in \{0, \ldots, 2^p - 1\}$ by tossing $p$ coins. If $r \geq \lfloor \frac{2^p}{p} \rfloor p$, output "fail". Otherwise, let $R_{ij} = r \mod p$. This procedure fails with probability at most $\frac{p}{2^p}$ and outputs a uniformly generated member of $Z_p$ otherwise.

Now, we use the fact that $q(z) = \text{Permanent}(M + zR)$ is a degree $t$ polynomial. For $i = 1, \ldots, t+1$, construct the matrix $M_i = M + iR$, where all the operations are done in $Z_p$. Generate a uniformly random representation for this matrix to present it as an input to $A_n$. Since every member of $Z_p$ has exactly $\lfloor \frac{2^p}{p} \rfloor$ different representations as a $p$-bit string, randomly choose one of these, independently for each entry of $M_i$. Denote this representation of $M_i$ by $x_i$. Clearly, $|x_i| = pt^2$.

For each matrix $M_i$, now represented by a $pt^2$ bit string $x_i$, and for each $j \in \{1, \ldots, 2pt\}$, uniformly pick a $p$-bit string $s_{i,j}$ and a $p(t^2-1)$-bit string $r_{i,j}$. With probability more than $1 - \frac{2p^2(t+1)t}{2^p}$, every $s_{i,j}$ is a valid representation of a uniformly distributed member of $Z_p$. If $x_i$ is a uniformly chosen valid representation of a matrix, $s_{i,j}$ is a valid representation of a uniformly chosen member of $Z_p$ and $r_{i,j}$ is a uniformly chosen $p(t^2-1)$-bit string, then with probability $> 1 - \frac{4pt(t+1)}{n^2}$, $A_n$ gives the correct answer on the $n$-bit string $x_i \circ s_{i,j} \circ r_{i,j}$, for all $i, j$, where $\circ$ denotes concatenation.

Fix an $i$. If all $s_{i,j}$ is valid, then with probability $1 - \left(1 - \frac{1}{p}\right)^{2pt} > 1 - e^{-2t}$ at least one $s_{i,j}$ represents a correct guess of the permanent of $M_i$. Therefore, the probability that for every $i$ there is a $j$, such that $s_{i,j}$ represents a correct guess of the permanent of $M_i$ is $> 1 - (t+1)e^{-2t}$.

So, $C'_n$ runs $A_n$ on the $2pt(t+1)$ samples $x_i \circ s_{i,j} \circ r_{i,j}$, reconstructs the polynomial $q(z) = \text{Permanent}(M + zR)$ from $A_n$'s answers and outputs $q(0)$.

The error probability of $C'_n$ is at most the sum $\frac{2p^2(t+1)t}{2^p} + \frac{4pt(t+1)}{n^2} + (t+1)e^{-2t} < \frac{1}{4}$. Thus, $C'_n$ is correct with probability $> \frac{3}{4}$. This can be amplified to a value more than $1 - 2^{-n^2}$ by repeating this process $\text{poly}(n)$ times and taking majority. This means that for a randomly picked string $r$, the probability that $C'_n$ gives the correct answer for *all* $x$'s when using $r$ as the source of randomness is $> 1 - \frac{2^n}{2^{n^2}}$. Note that the length of $r$ is polynomial in $n$.

*Construction of $C_n$*: To construct $C_n$, pick $r$ randomly and build it into $C_n$ which then simulates $C'_n$ with $r$ as the source of randomness.

*Time/size analysis for $C_n$*: Because the size of $A_n$ is $O(2^{3n^{\epsilon_0/10}})$, it is clear from the above that the time required to construct $C_n$ is $\text{poly}(n) \cdot O(2^{3n^{\epsilon_0/10}}) = O(2^{n^{\epsilon_0/9}})$. This is also an upper bound for the size of $C_n$.

*Probability of a correct $C_n$*: As seen earlier, a correct $C_n$ is obtained with probability $> 1 - \frac{2^n}{2^{n^2}}$. $\square$

This completes the inductive construction of $C_n$. The total time for the first four stages is as follows:

1. *Construction of $D_n$*: $O(2^{2(\log d)^c})$
2. *Construction of $E_n$*: $O(2^{2(\log d)^c} 2^{2n^{\frac{\epsilon_0}{9}}})$
3. *Construction of $F_n$*: $O(2^{5(\log d)^c})$
4. *Construction of $A_n$*: $O\left(2^{3n^{\epsilon_0/9}}\right)$

It can be checked that the total time taken by these four stages is less than $2^{n^{\epsilon_0/8}}$. We repeat these steps at most $2^{n^{\frac{\epsilon_0}{4}}}$ times and then test the circuits in time $\text{poly}(n) 2^{2n^{\frac{\epsilon_0}{9}}} 2^{n^{\frac{\epsilon_0}{4}}}$ as outlined earlier. The total time taken is at most $2^{n^{\frac{\epsilon_0}{2}}}$. The time required for the last stage (going from $A_n$ to $C_n$) is $O(2^{n^{\frac{\epsilon_0}{9}}})$. Thus, the overall time to construct $C_n$ is $< 2^{n^{3\epsilon_0/4}}$.

Therefore, in time $O(2^{n^{3\epsilon_0/4}})$ we have constructed a circuit $C_n$ of size $O(2^{n^{\epsilon_0/9}})$, that solves $Perm_n$. On inputs of length $n$, the BPTIME algorithm for $Perm$ will construct $C_1, \ldots, C_n$ in time at most $n \cdot O(2^{n^{3\epsilon_0/4}})$, and then simulate $C_n$ in time $O(2^{n^{\epsilon_0/9}})$, a total time $< 2^{n^{\epsilon_0}}$. This shows that $Perm \in \text{BPTIME}(2^{n^{\epsilon_0}})$.

# 5 Derandomizing BPQP if EXP is Hard

This section is devoted to the proof of Theorem 3. Suppose $\exists f \in$ EXP s.t. $f$ cannot be computed by circuits of size $2^{n^{\epsilon_1}}$ where $\epsilon_1$ is as in Theorem 3. Using the approach of [BFNW93] we will show that every language $L \in \mathrm{BPTIME}(2^{(\log d)^c})$ can be deterministically simulated in time $O(2^{(\log d)^{c'}})$ for infinitely many input lengths $d$.

Let $f_n$ be the restriction of $f$ to $n$-bit inputs. Let $g_n$ be the unique multi-linear extension of $f_n$. Then,

$$g_n(x_1, \dots, x_n) = \sum_{\bar{b} \in \{0,1\}^n} f_n(\bar{b}) \prod_i X_{i,\bar{b}},$$

where $X_{i,\bar{b}} = x_i$ if $b_i = 1$ and $1 - x_i$ otherwise. Let $F$ be a finite field of size $q$, where $q$ is the smallest power of 2 greater than or equal to $n+2$. Then $g_n$ is considered to be defined over $F$. Let $g$ be the family of functions $g_n$. The following sequence of lemmas will prove Theorem 3. $\square$

**Lemma 8** *There is a PSPACE machine with oracle access to $f$ that computes $g$.*

The proof is obvious from the definition of $g_n$. This lemma means that $g$ can be computed by an EXP machine.

**Lemma 9** *If no circuit of size $2^{n^{\epsilon_1}}$ computes $f_n$, then no circuit of size $2^{n^{\epsilon_1/2}}$ computes $g_n$ on a $1 - \frac{1}{3n}$ fraction of the inputs.*

**Proof.** Since $f_n$ cannot be computed by a circuit of size $2^{n^{\epsilon_1}}$, $g_n$ also cannot be computed by such a circuit. Assume that there is a circuit $D_n$ of size $2^{n^{\epsilon_1/2}}$ that computes $g_n$ on a $1 - \frac{1}{3n}$ fraction of the inputs. Then, there is a probabilisitic circuit $D'_n$ of size $2^{n^{3\epsilon_1/4}}$, such that $\forall y = (x_1, \dots, x_n)$,

$$\Pr_{\text{coins}}[D'_n(y) = g_n(y)] > \frac{3}{5}.$$

This can be seen as follows. To evaluate $g_n(\alpha_1, \dots, \alpha_n)$, $D'_n$ randomly chooses $r_1, \dots, r_n$ and computes using $D_n$, $p(t) = g(\alpha_1 + tr_1, \dots, \alpha_n + tr_n)$, for $t = 1, \dots, n+1$. Then, it reconstructs the $n$-degree polynomial $p$ and outputs $p(0)$. Since the $r_i$ have been chosen randomly, $g$ is evaluated on $n+1$ inputs, each of which is uniformly distributed. Therefore, with probability $\geq 1 - \frac{n+1}{3n} \geq \frac{3}{5}$, $D_n$ correctly evaluates $g_n$ on all the $n+1$ points. This probability can then be increased, to say $1 - 2^{-n}$, by repeating this process poly$(n)$ times and taking the most frequently output value. That is, there is a circuit $D''_n$ of size $2^{n^{\epsilon_1}}$ that achieves

$$\forall y \quad \Pr_{\text{coins}}[D''_n(y) = g_n(y)] > 1 - 2^{-n^2}.$$

This can be converted to a deterministic circuit of the same size $(2^{n^{\epsilon_1}})$ that produces the correct answer for every input. This is a contradiction. $\square$

Let $h$ be the Goldreich-Levin [GL89] predicate for $g$, i.e. $h(y, r) = \langle g(y), r \rangle$, where $\langle \cdot, \cdot \rangle$ is the inner product mod 2, $|y| = n \log q$ and $|r| = |g(y)| = \log q < n$. ($q$ is the size of the field on which $g_n$ is defined.)

**Lemma 10** *If there is no circuit of size $2^{n^{\epsilon_1/2}}$ that computes $g_n$ on a $1 - \frac{1}{3n}$ fraction of the inputs, then for all circuits $E_n$ of size $2^{n^{\epsilon_1/4}}$,*

$$\Pr_{y,r}[E_n(y, r) = h(y, r)] < 1 - \frac{1}{6n^2}.$$

**Proof.** Suppose there is a circuit $E_n$ of size $2^{n^{\epsilon_1/4}}$ such that,

$$\Pr_{y,r}[E_n(y,r) = h(y,r)] \geq 1 - \frac{1}{6n^2}.$$

Here is a probabilistic circuit $F_n$ to compute $g_n(y)$. To compute the $i^{\text{th}}$ bit of $g_n(y)$, $F_n$

1. Picks $r_i$ randomly. Let $\bar{r}_i$ be $r_i$ with the $i^{\text{th}}$ bit flipped.
2. Computes $E_n(y, r_i) \oplus E_n(y, \bar{r}_i)$.

With probability $\geq 1 - \frac{2}{6n^2}$, both the outputs of $E_n$ are correct and their parity is the correct value of the $i^{\text{th}}$ bit. Thus, with probability $\geq 1 - \frac{2n}{6n^2} = 1 - \frac{1}{3n}$ all bits of $g_n(y)$ are computed correctly, because $|g_n(y)| = \log q < n$. There is a setting of the random bits $r = R$ of $F_n$ which ensures that

$$\Pr_{y}[F_n(y) = g_n(y)] \geq 1 - \frac{1}{3n},$$

with $R$ as the random bits.

The size of $F_n$ is at most $2^{n^{\epsilon_1/2}}$ and this contradicts our assumption that no circuit of size $2^{n^{\epsilon_1/2}}$ computes $g_n$ on a $1 - \frac{1}{3n}$ fraction of the inputs. □

**Lemma 11** *If no circuit of size $2^{n^{\epsilon_1/4}}$ can compute $h$ correctly on more than a $1 - \frac{1}{6n^2}$ fraction of the inputs, then $\exists h' \in \mathrm{EXP}$ such that no circuit of size $2^{n^{\epsilon_1/8}}$ can compute $h'$ correctly on more than a $\frac{1}{2} + \frac{1}{2^{n^{\epsilon_1/8}}}$ fraction of the inputs of length $m = poly(n)$.*

**Proof.** In the notation of Nisan and Wigderson [NW94], $h$ is a $\left(1 - \frac{1}{3n^2}, 2^{n^{\epsilon_1/4}}\right)$-hard function. From Yao's XOR lemma [Yao82], we get that if $h$ is $(\epsilon, S)$ -hard then $h'(x_1, \ldots, x_k) = \bigoplus_{i=1}^{k} h(x_i)$ is $(\epsilon^k + \delta, \delta^2(1-\epsilon)^2 S)$-hard for any $\delta > 0$.

Our $\epsilon$ is $1 - \frac{1}{3n^2}$ and $S$ is $2^{n^{\epsilon_1/4}}$. Let $\epsilon_2 = \epsilon_1/8$. Let $k = 3n^2(n^{\epsilon_2} + 1)$ and $\delta = \frac{1}{2^{n^{\epsilon_2}+1}}$. Then $\epsilon^k + \delta \leq \frac{1}{2^{n^{\epsilon_2}}}$ and $\delta^2(1-\epsilon)^2 S > 2^{n^{\epsilon_2}}$. Let $m = k(n+1)\log q = 3n^2(n^{\epsilon_2}+1)(n+1)\log q$. So, now we have a function $h' \in \mathrm{EXP}$ whose hardness (refer [NW94] for definition) at input length $m$, (denoted $H_m(h')$) is greater than $2^{n^{\epsilon_1/8}}$. We can rewrite this as $H_m(h') \geq 2^{m^\eta}$, for some $\eta > 0$. □

Because there is an *infinite* sequence of $n$'s such that $h$ at length $n$ cannot be computed correctly on more than a $1 - \frac{1}{6n^2}$ fraction of the inputs by circuits of size $2^{n^{\epsilon_1/4}}$, we get an *infinite* sequence of $m$'s such that, for some $\eta > 0$, $h'$ has hardness $\geq 2^{m^\eta}$ at length $m$. The next lemma asserts that this is enough to give us an io-deterministic simulation of BPQP.

**Lemma 12** *If $\exists \phi \in \mathrm{EXP}$ and an infinite sequence of $m$'s such that $H_m(\phi) \geq 2^{m^\epsilon}$, for some $\epsilon > 0$, then $\forall c \geq 1\ \exists c'$ s.t. any $L \in \mathrm{BPTIME}(2^{(\log d)^c})$ can be simulated in $\mathrm{DTIME}(2^{(\log d)^{c'}})$, for infinitely many lengths $d$.*

**Proof.** Let $L \in \mathrm{BPTIME}(2^{(\log d)^c})$ for some $c \geq 1$. Let $\phi \in \mathrm{DTIME}(2^{n^\alpha})$. We will show $L \in \mathrm{DTIME}(2^{(\log d)^{c'}})$, for some $c' \geq c$ and for infinitely many $d$. Consider a design as in [NW94] with parameters $((\log d)^{2c_1}, (\log d)^{c_1}, (\log d)^c, 2^{(\log d)^c})$, where $c_1$ is any constant $> \frac{2c}{\epsilon}$. This gives us a PSRG $G$ mapping $(\log d)^{2c_1}$ bits to $2^{(\log d)^c}$ bits.

Now consider the following deterministic simulation for $L$. Cycle through all $2^{(\log d)^{2c_1}}$ seeds, and for each one of them run the BP machine for $L$ with the output of $G$ as the random bits and take majority. The total time required by this simulation is $O(2^{(\log d)^{c_1 \alpha}} 2^{(\log d)^c} 2^{(\log d)^{2c_1}}) = O(2^{(\log d)^{c'}})$, for an appropriate $c' \geq c$. The following claim completes the proof of Theorem 3. It shows that if the above deterministic simulation fails for inputs of all sufficiently large lengths, then there exists a circuit contradicting the hardness assumption for $\phi$. The proof of the claim can be found in the appendix.

**Claim** *If the above simulation fails for all sufficiently large $d$, then for all sufficiently large $m$, there exists a circuit $C_m$ of size $< 2^{m^\epsilon}$ that satisfies,*

$$\Pr_{x\in\{0,1\}^m}[C_m(x) = \phi(x)] > \frac{1}{2} + 2^{-m^\epsilon}.$$

*This contradicts the hardness assumption for $\phi$.*

## References


[ABHH93] Eric Allender, Richard Beigel, Ulrich Hertrampf, and Steven Homer. Almost-everywhere complexity hierarchies for nondeterministic time. *Theoretical Computer Science*, 115(2):225–241, 19 July 1993.

[BFNW93] László Babai, Lance Fortnow, Noam Nisan, and Avi Wigderson. BPP has subexponential time simulations unless EXPTIME has publishable proofs. *Computational Complexity*, 3(4):307–318, 1993.

[BH97] C. Berg and J. Håstad. On the BPP hierarchy problem. Technical Report TRITA NA 97–02, Royal Institute of Technology, 1997.

[BM84] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13(4):850–864, November 1984.

[Coo73] Stephen A. Cook. A hierarchy for nondeterministic time complexity. *Journal of Computer and System Sciences*, 7(4):343–353, August 1973.

[FS89] L. Fortnow and M. Sipser. Probabilistic computation and linear time. In *Proc. 21st Annual ACM Symposium on the Theory of Computing*, pages 148–156, 1989.

[FS97] L. Fortnow and M. Sipser. Retraction of "Probabilistic computation and linear time". In *Proc. 29th Annual ACM Symposium on the Theory of Computing*, page 750, 1997.

[GHS91] John G. Geske, Dung T. Huynh, and Joel I. Seiferas. A note on almost-everywhere-complex sets and separating deterministic-time-complexity classes. *Information and Computation*, 92(1):97–104, May 1991.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, pages 25–32, Seattle, Washington, 15–17 May 1989.

[HS65] J. Hartmanis and R. E. Stearns. On the computational complexity of algorithms. *Trans. Amer. Math. Soc.*, 117:285–306, 1965.

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

[IW97] Russell Impagliazzo and Avi Wigderson. $P = BPP$ if $E$ requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 220–229, El Paso, Texas, 4–6 May 1997.

[IW98] R. Impagliazzo and A. Wigderson. Randomness vs. time: De-randomization under a uniform assumption. In *Proc. 39th IEEE Symposium on Foundations of Computer Science (FOCS)*, 1998.

[KL80] Richard M. Karp and Richard J. Lipton. Some connections between nonuniform and uniform complexity classes. In *Conference Proceedings of the Twelfth Annual ACM Symposium on Theory of Computing*, pages 302–309, Los Angeles, California, 28–30 April 1980.

[KV87] M. Karpinski and R. Verbeek. Randomness, provability, and the separation of Monte Carlo time and space. In *Computation Theory and Logic*, volume 270 of *Lect. Notes in Comp. Sci.*, pages 189–207. Springer Verlag, 1987.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, October 1994.

[SFM78] Joel I. Seiferas, Michael J. Fischer, and Albert R. Meyer. Separating nondeterministic time complexity classes. *Journal of the ACM*, 25(1):146–167, January 1978.

[Tod91] Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, October 1991.

[Yao82] Andrew C. Yao. Theory and applications of trapdoor functions (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 80–91, Chicago, Illinois, 3–5 November 1982. IEEE.

# Appendix

**Construction of $E_n$ (Lemma 4)**

$G_n$ is a function from $l$ bits to $2^{(\log d)^c}$ bits where both $l$ and $d$ are functions of $n$. Let the output bits of $G_n$ be denoted by $b_1, \ldots, b_{2^{(\log d)^c}}$. Let $S_i$ be the set of $m$ input bits that output bit $b_i$ depends on. To probabilistically construct a circuit $E_n$, do the following,

1. Pick $i \in_U [1, \ldots, 2^{(\log d)^c}]$.

2. Pick randomly those input bits of $G_n$ that $b_i$ does not depend on, in other words, those input bits lying outside $S_i$.

3. Compute all possible values for $b_1, \ldots, b_{i-1}$. This involves computing $h_n$ on the bits in $S_1, \ldots, S_{i-1}$ respectively, assuming all possible values for the bits in $S_i \cap S_j$, $j < i$. Since for $i \neq j$, $|S_i \cap S_j|$ is at most $(\log d)^c$, at most $2^{(\log d)^c}$ bits need to be stored for each output bit. To compute $h_n$ we need to have the ability to compute $Perm_n$. But using the downward-self-reducibility of $Perm$ and the fact that circuits for $Perm_1, \ldots, Perm_{n-1}$ are available, we can do this computation.

4. Pick output bits $b_i, b_{i+1}, \ldots, b_{2^{(\log d)^c}}$ randomly.

*Working of $E_n$*: Given $w$, it looks up the correct value for the first $i-1$ output bits. and presents all the output bits to $D_n$. If $D_n$ returns 1 then $E_n$ outputs $b_i$, otherwise it outputs $\bar{b}_i$.

*Note.* We do not know if the $z$ chosen in the construction of $D_n$ is in $L$ or not. If $z \notin L$, $E_n$ should be constructed as above, if $z \in L$, $E_n$ should do the reverse. We can get around this problem by randomly choosing one of the two behaviors when constructing $E_n$.

*Time/size analysis for $E_n$*: It can be checked that the time to construct $E_n$ is $O(2^{2(\log d)^c} \cdot 2^{2n^{\frac{\epsilon_0}{9}}} +$ size of $D_n) = O(2^{2(\log d)^c} \cdot 2^{2n^{\frac{\epsilon_0}{9}}})$, because the size of $D_n$ is $O(2^{2(\log d)^c})$. We assume, by induction, that the size of lower circuits is at most $2^{n^{\frac{\epsilon_0}{9}}}$. The size of $E_n$ is also easily seen to be $O(2^{2(\log d)^c})$. Note that the size of $E_n$ is independent of the size of the circuits for $Perm_1, \ldots, Perm_{n-1}$.

*Probability of a good $E_n$*: Let $Q_E = \Pr_w[E_n(w) = h_n(w)]$. Then using standard techniques, it can be shown that

$$\mathrm{Exp}_E[Q_E] \geq \frac{1}{2} + \frac{1}{4 \cdot 2^{(\log d)^c}}.$$

Now, by Markov's inequality, we can prove that, with probability $\geq \frac{1}{4 \cdot 2^{(\log d)^c}}$, $Q_E \geq \frac{1}{2} + \frac{1}{8 \cdot 2^{(\log d)^c}}$. □

**Probability of a good $F_n$ (Lemma 5)**:

Let $Q_F = \Pr_y[F_n(y) = g_n(y)]$. By hypothesis,

$$\Pr_{y,r}[E_n(y,r) = h_n(y,r)] \geq \frac{1}{2} + \frac{1}{8 \cdot 2^{(\log d)^c}}.$$

By applying Markov's inequality, we can show that there is a set $T$ of $y$'s of size $\geq \frac{1}{16 \cdot 2^{(\log d)^c}} 2^{nk}$ such that, $\forall y \in T$,

$$\Pr_{r}[E_n(y,r) = h_n(y,r)] \geq \frac{1}{2} + \frac{1}{16 \cdot 2^{(\log d)^c}}.$$

Let us calculate the probability that $F_n$ computes the first bit wrongly. This probability is both over a random construction of $F_n$ and a random input to $F_n$. Assume $y \in T$. This happens with probability $> \frac{1}{2^{(\log d)^c+4}}$. Also assume the $\sigma_i$ are all correct guesses. This happens with probability $\frac{1}{2^s}$. Define random variables $X_i$, $1 \leq i \leq 2^s - 1$ , as follows : $X_i = 1$ iff $E_n(y, t_i^1)$ is correct, where we identify a non-zero $S \in \{0,1\}^s$ with an integer $i$ in the interval $[1, 2^s - 1]$ in the natural way. We know that $\mathrm{Exp}(X_i) \geq \frac{1}{2} + \frac{1}{2^{(\log d)^c+4}}$ and that the $X_i$ are pairwise independent. Because the $X_i$ are 0-1 random variables, $\mathrm{Var}(X_i) < 1$, for all $i$. Let $X = \sum_{i=1}^{2^s-1} X_i$. We have,

$$\begin{aligned}
\Pr(\text{majority gives wrong answer}) &< \Pr\left(|X - \mathrm{Exp}(X)| \geq \frac{2^s-1}{2^{(\log d)^c+4}}\right) \\
&< \frac{(2^s-1)(2^{2(\log d)^c+8})}{(2^s-1)^2} \\
&< \frac{2^{2(\log d)^c+8}}{2^{s-1}}
\end{aligned}$$

Therefore, the probability that all $k$ bits are computed correctly ($= \mathrm{Exp}_F(Q_F)$) can be estimated as follows:

$$\mathrm{Exp}_F(Q_F) > \frac{1}{2^s} \frac{1}{2^{(\log d)^c+4}} [1 - \frac{k}{2^{s-1}} 2^{2(\log d)^c+8}].$$

Since $k = \mathrm{poly}(n)$, a choice of $s = 3(\log d)^c$ ensures that $\mathrm{Exp}_F(Q_F) > \frac{1}{2^{(\log d)^{2c}}}$. Therefore, by applying the Markov inequality, we can conclude that $\Pr_F[Q_F > \frac{1}{2^{2(\log d)^{2c}}}] > \frac{1}{2^{2(\log d)^{2c}}}$. □

**Proof of Lemma 6**

With a choice of $k = O(-\frac{\log \epsilon}{\delta})$ (in our case $k = o(n^5)$), Impagliazzo and Wigderson construct a distribution on probabilistic circuits $A'$, taking $n$-bit inputs such that, for any subset $H$ of inputs of size $\geq \delta 2^n$,

$$\mathrm{Exp}_{A'}[\Pr_{x \in_U H, \text{coins of } A'}(A'(x) = Perm_n(x))] \geq \frac{1}{2} + \frac{\epsilon}{32}.$$

The distribution is obtained as follows : first $i$ is chosen uniformly in $\{1, \ldots, k\}$ and for each $j \neq i$, $x_j$ is chosen uniformly. Then, $Perm_n(x_j)$ is computed for each $j \neq i$. All this information is stored in $A'$. On input $x$, $A'$ sets $x_i = x$ and simulates $F_n$ on $(x_1, \ldots, x_k)$. It then computes $t =$ the number of $j$'s, $j \neq i$, $j^{\text{th}}$ output of $F_n \neq Perm_n(x_j)$. With probability $\frac{1}{2^t}$, it outputs the $i^{\text{th}}$ output bit of $F_n$ and outputs a randomly chosen bit otherwise.

Let $H = \{x \mid \Pr_{A',\text{coins of } A'}(A'(x) = Perm_n(x)) < 1/2 + \epsilon/32\}$. Then

$$\text{Exp}_{A'}[\Pr_{x \in_U H, \text{coins of } A'}(A'(x) = Perm_n(x))] < \frac{1}{2} + \frac{\epsilon}{32}.$$

This implies that $|H| < \delta 2^n$.

If $x \notin H$,

$$\Pr_{A', \text{ coins of } A'}(A'(x) = Perm_n(x)) \geq \frac{1}{2} + \frac{\epsilon}{32}.$$

Probabilistically construct $r$ samples that each behave like $A'$ except that the random bits that $A'$ needs are built into them. To construct one sample pick $i \in_U \{1, \ldots, k\}$. For each $j \neq i$, pick $x_j \in_U \{0,1\}^n$, compute $Perm_n(x_j)$ using lower circuits and store the results. Toss $k$ coins $c_1, \ldots, c_k$ and store the results. Do this $r$ times independently. Call the samples $C_1, \ldots, C_r$. Each $C_l$ works as follows on input $x$. It plugs $x$ into the $i^{\text{th}}$ spot and runs $F_n$ on $(x_1, \ldots, x_k)$. It then computes the number of $j$, $j \neq i$, such that the $j^{\text{th}}$ output bit produced by $F_n$ does not agree with the stored value for $Perm_n(x_j)$. Let this number be $t$. It then looks at $c_1, \ldots, c_t$. If they are all zero, it outputs the $i^{\text{th}}$ output bit of $F_n$, otherwise it outputs $c_k$. $A_n$ is the circuit that outputs the majority answer of the $C_l$.

We will show that by taking $r$ sufficiently large, we can ensure that $\forall x \notin H$ $\Pr_{A_n}(A_n(x) = Perm_n(x)) > 1 - 2^{-n^2}$. Fix an $x \notin H$. Let $A_n$ be produced as described above. Let $b_l = C_l(x)$. Define 0-1 random variables $X_l$ as follows:

$$X_l = 1 \text{ iff } b_l = Perm_n(x).$$

Thus, $X_l$ is a random variable that depends on the choices of $i$, $x_j$ for $j \neq i$, and $c_1, \ldots, c_k$ in the construction of $C_l$. Let $X = \sum_l X_l$. Because $\forall l$ $\text{Exp}(X_l) \geq \frac{1}{2} + \frac{\epsilon}{32}$, $\mu = \text{Exp}(X) \geq \frac{r}{2} + \frac{r\epsilon}{32}$. We now have to bound the probability that $X < \frac{r}{2}$. We have,

$$\begin{aligned} \Pr(X < \frac{r}{2}) &< \Pr(X < (1 - \frac{\epsilon}{32})\mu) \\ &= e^{-\Omega(\epsilon^2 r)}. \end{aligned}$$

So a choice of $r = O(n^2/\epsilon^2)$ makes this bound $< 2^{-n^2}$.

Thus for every $x \notin H$, we succeed in constructing $A_n$ computing $Perm_n(x)$ with probability $1 - 2^{-n^2}$. Therefore with probability $1 - \frac{2^n}{2^{n^2}}$, a randomly constructed $A_n$ will be simultaneously good for all $x \notin H$.

*Time/size analysis for $A_n$*: The time to construct $A_n$ can be seen to be $O((n^2/\epsilon^2)(2^{2n^{\epsilon_0/9}} + \text{size of } F_n))$, again assuming that the size of smaller circuits is at most $2^{n^{\epsilon_0/9}}$. The size of $A_n$, on the other hand, is only $O(\frac{n^2}{\epsilon^2}(\text{size of } F_n))$. Since the size of $F_n$ is $O(2^{5(\log d)^c})$, these quantities are $O\left((n^2/\epsilon^2)(2^{2n^{\epsilon_0/9}} + 2^{5(\log d)^c})\right) = O\left(2^{3n^{\epsilon_0/9}}\right)$ and $O\left(\frac{n^2}{\epsilon^2}\left(2^{5(\log d)^c}\right)\right) = O\left(2^{3n^{\epsilon_0/10}}\right)$ respectively.
□

**Proof of Claim**

Let $d = 2^{m^{1/c_1}}$. Assume $m$ is large enough so that the simulation fails for a $z \in \{0,1\}^d$. Wlog let $z \in L$, and let $M$ be the BP machine for $L$. Let $D$ be the circuit that takes the random bits of $M$ as input and simulates $M$ on $z$ with those random bits. Because the simulation fails for $z$,

$$\Pr_y[D(y) = 1] \geq \frac{3}{4}$$

but

$$\Pr_x[D(G(x)) = 1] < \frac{1}{2},$$

where $y \in_U \{0,1\}^{2^{(\log d)^c}}$ and $x \in_U \{0,1\}^{(\log d)^{2c_1}}$. Therefore, $\Pr_y[D(y) = 1] - \Pr_x[D(G(x)) = 1] \geq \frac{1}{4}$. It is clear that $D$ is of size $O(2^{2^{(\log d)^c}})$. Here is a probabilistic circuit $C_m$ to compute $\phi$ on inputs of length $m = (\log d)^{c_1}$. $C_m$ does the following on input $w \in \{0,1\}^m$ :

1. Guesses $i \in [1, \ldots, 2^{(\log d)^c}]$
2. Guesses output bits $b_i, \ldots, b_{2^{(\log d)^c}}$ of the generator
3. Guesses input bits of the generator that don't affect $b_i$
4. Computes $b_1, \ldots, b_{i-1}$ by means of circuits of size $2^{(\log d)^c}$
5. Runs $D$ on $b_1, \ldots, b_{2^{(\log d)^c}}$
6. If $D$ says 1, outputs $\bar{b}_i$ else outputs $b_i$

We have,

$$\Pr_{x,\text{coins}}[C_m(x) = \phi(x)] \geq \frac{1}{2} + \frac{1}{2^{(\log d)^c}} \frac{1}{4}.$$

Therefore, the private random bits of $C_m$ can be set in such a way that

$$\Pr_x[C_m(x) = \phi(x)] \geq \frac{1}{2} + \frac{1}{2^{(\log d)^c}} \frac{1}{4} > \frac{1}{2} + \frac{1}{2^{(\log d)^{2c}}}.$$

Also, $C_m$ is a circuit of size $< 2^{(\log d)^{2c}}$ and $(\log d)^{2c} < (\log d)^{c_1 \epsilon} = m^\epsilon$. $\square$