# Threshold Computation and Cryptographic Security*

Yenjo Han† Lane A. Hemaspaandra‡ Thomas Thierauf§

Revised Typescript Completed: September 6, 1995.

**Abstract**

Threshold machines are Turing machines whose acceptance is determined by what portion of the machine's computation paths are accepting paths. Probabilistic machines are Turing machines whose acceptance is determined by the probability weight of the machine's accepting computation paths. In 1975, Simon proved that for unbounded-error polynomial-time machines these two notions yield the same class, PP. Perhaps because Simon's result seemed to collapse the threshold and probabilistic modes of computation, the relationship between threshold and probabilistic computing for the case of bounded error has remained unexplored.

In this paper, we compare the bounded-error probabilistic class BPP with the analogous threshold class, $\mathrm{BPP}_{\mathrm{path}}$, and, more generally, we study the structural properties of $\mathrm{BPP}_{\mathrm{path}}$. We prove that $\mathrm{BPP}_{\mathrm{path}}$ contains both $\mathrm{NP}^{\mathrm{BPP}}$ and $\mathrm{P}^{\mathrm{NP[log]}}$, and that $\mathrm{BPP}_{\mathrm{path}}$ is contained in $\mathrm{P}^{\Sigma_2^p[\log]}$, $\mathrm{BPP}^{\mathrm{NP}}$, and PP. We conclude that, unless the polynomial hierarchy collapses, bounded-error threshold computation is strictly more powerful than bounded-error probabilistic computation.

We also consider the natural notion of secure access to a database: an adversary who watches the queries should gain no information about the input other than perhaps its length. We show, for both BPP and $\mathrm{BPP}_{\mathrm{path}}$, that if there is *any* database for which this formalization of security differs from the security given by oblivious database access, then $\mathrm{P} \neq \mathrm{PSPACE}$. It follows that if any set lacking small circuits can be securely accepted, then $\mathrm{P} \neq \mathrm{PSPACE}$.

*Research supported in part by the National Science Foundation under grants CCR-8957604, CCR-9057486, CCR-9322513, and INT-9116781/JSPS-ENG-207, and by DFG Postdoctoral Stipend Th 472/1-1.

†University of Rochester. Current address: Microtec Research Inc., 2350 Mission College Blvd., Santa Clara, CA 95054.

‡University of Rochester. Department of Computer Science, University of Rochester, Rochester, NY 14627 USA.

§Universität Ulm. Abteilung Theoretische Informatik, Universität Ulm, Oberer Eselsberg, 89069 Ulm, Germany. Work done in part while visiting the University of Rochester and Princeton University.

# 1 Introduction

In 1975, Simon [Sim75] defined threshold machines. A threshold machine is a nondeterministic Turing machine that accepts a given input if more than half of all computation paths on that input are accepting paths. Gill [Gil77] defined the class PP as the class of sets for which there exists a probabilistic polynomial-time Turing machine that accepts exactly the members of the set with probability greater than 1/2. Simon [Sim75] showed that the class of sets accepted by polynomial-time threshold machines characterizes the unbounded-error probabilistic class PP.

In this paper, we extend the notion of threshold computation to bounded-error probabilistic classes, and we study the degree to which threshold and probabilistic database ("oracle") computations hide information from observers.

In particular, we introduce $\mathrm{BPP}_{\mathrm{path}}$ and $\mathrm{R}_{\mathrm{path}}$ as the threshold analogs of BPP and R [Gil77]. We give evidence that, unlike the case for PP, these threshold classes are different from their probabilistic counterparts. Section 3 studies the properties of the class $\mathrm{BPP}_{\mathrm{path}}$ and its relationship to other complexity classes. For example, we show—in contrast to the BPP case—that $\mathrm{BPP}_{\mathrm{path}}$ is self-low (i.e., $\mathrm{BPP}_{\mathrm{path}}{}^{\mathrm{BPP}_{\mathrm{path}}} = \mathrm{BPP}_{\mathrm{path}}$) only if the polynomial hierarchy collapses. We also show that BPP is low for $\mathrm{BPP}_{\mathrm{path}}$, that there is a relativized world in which $\mathrm{BPP}_{\mathrm{path}}$ does not contain the smallest reasonable counting class, and that $\mathrm{BPP}_{\mathrm{path}}$ has many closure properties. Figure 1 gives an overview of the inclusion relations we establish between $\mathrm{BPP}_{\mathrm{path}}$ and other complexity classes; in particular, note that, though contained in the polynomial hierarchy, $\mathrm{BPP}_{\mathrm{path}}$ contains NP and coNP.

Section 4 studies, for threshold and probabilistic computations that have Turing (that is, adaptive) access to a database, the degree to which the input can be hidden from an observer. In particular, we consider the least restrictive possible notion ensuring that a powerful observer should gain no information about the input other than its length [BF90]. For the cases of unbounded-error probabilistic and threshold computation, we note that this optimal degree of security can be achieved in all cases. For the cases of bounded-error probabilistic and threshold computations, we prove the following result: If there exists any database $D$ to which secure access yields more power than oblivious access (a notion in which the querying machine—until finished querying—is wholly denied access to the input other than the length of the input [FFLS92]), then $\mathrm{P} \neq \mathrm{PSPACE}$.

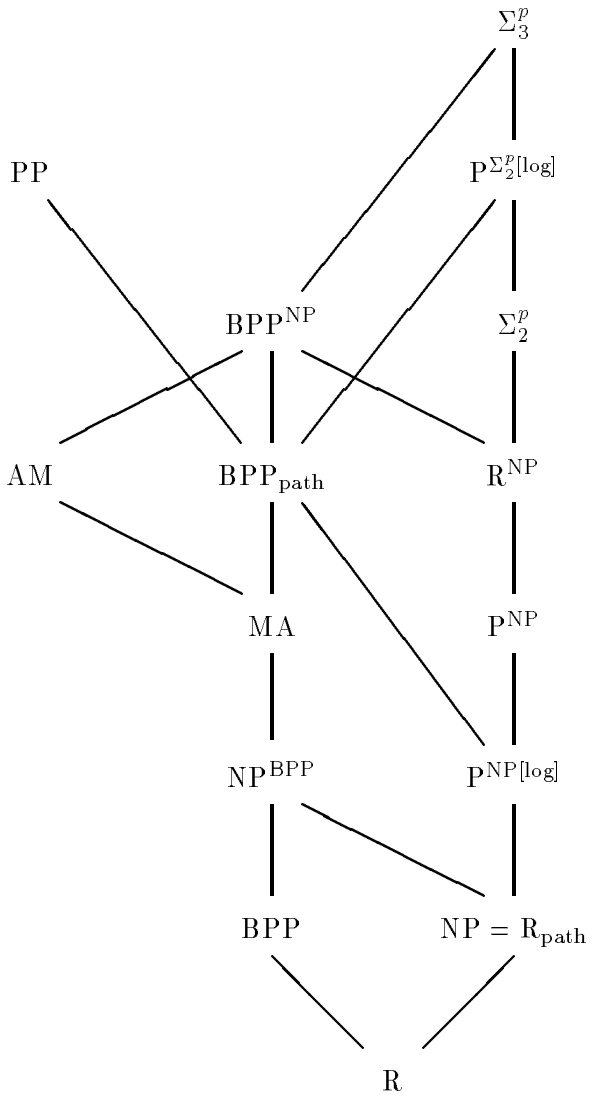


Figure 1: Inclusion relations between $\mathrm{BPP}_{\mathrm{path}}$ and other complexity classes. Not shown are the inclusions $\mathrm{BPP} \subseteq \mathrm{R}^{\mathrm{NP}}$ and $\mathrm{AM} \subseteq \mathrm{P}^{\Sigma_2^p[\log]}$ (in fact, $\mathrm{AM} \subseteq \Pi_2^p$). As an open problem, we ask whether $\mathrm{BPP}_{\mathrm{path}}$ is contained in $\Sigma_2^p$, or even $\mathrm{R}^{\mathrm{NP}}$. There is an oracle relative to which $\mathrm{BPP}_{\mathrm{path}}$ is not in $\mathrm{P}^{\mathrm{NP}}$.

## 2 Definitions and Discussion

Throughout this paper, we use the alphabet $\Sigma = \{0,1\}$. For a string $x \in \Sigma^*$, $|x|$ denotes the length of $x$. For a set $A \subseteq \Sigma^*$, $A(x)$ denotes the characteristic function of $A$, $A^{=n}$ denotes $\{\, y \mid y \in A \text{ and } |y| = n \,\}$, $A^{\leq n}$ denotes $\{\, y \mid y \in A \text{ and } |y| \leq n \,\}$, and $\|A\|$ denotes the cardinality of $A$. The complement of $A$ is $\overline{A} = \Sigma^* - A$, and for a class $\mathcal{C}$ of sets, $\mathrm{co}\mathcal{C} = \{\, \overline{A} \mid A \in \mathcal{C} \,\}$.

Let $(\cdot,\cdot)_b : \Sigma^* \times \Sigma^* \to \Sigma^*$ be a polynomial-time computable, polynomial-time invertible, one-to-one, onto function. For any string $z$, let $z+1$ denote the string that lexicographically follows $z$, and for any string $z \neq \epsilon$, let $z-1$ denote the string that lexicographically precedes $z$. Let $k_s$ be the lexicographically $k$th string in $\Sigma^*$. We define our (multi-arity, onto) pairing function by $(x_1, x_2, \cdots, x_k)$ equals (a) $(\epsilon,\epsilon)_b$ when $k = 0$, (b) $(\epsilon, x_1 + 1)_b$ when $k = 1$, and (c) $(k_s, (x_1, (x_2, (\cdots(x_{k-1}, x_k)_b \cdots)_b)_b)_b)_b$ when $k \geq 2$.

P (NP) denotes the class of languages that are accepted by polynomial-time deterministic (nondeterministic) Turing machines. For nondeterministic Turing machines we assume without loss of generality that the nondeterministic branching degree is at most two. $M$ is polynomial-normalized (henceforward denoted *normalized*) if there is a polynomial $p$ such that on every input $x$ the machine $M$ makes exactly $p(|x|)$ nondeterministic moves on each computation path. FP is the class of polynomial-time computable functions. One can define relativized classes such as $\mathrm{P}^{\mathrm{NP}}$ (respectively, $\mathrm{P}^{\mathrm{NP[log]}}$) by employing P machines having some NP oracle that can be asked polynomially (respectively, logarithmically) many queries, i.e., so-called oracle machines [BGS75]. This is called a *Turing reduction* (to NP). If the queries are made nonadaptively (i.e., in parallel) we call this a *truth-table reduction* (see Ladner, Lynch, and Selman [LLS75]). By $\mathrm{P}^{\mathrm{NP}}_{\mathrm{tt}}$ we denote the class of sets that are truth-table reducible to NP. But in fact, $\mathrm{P}^{\mathrm{NP}}_{\mathrm{tt}} = \mathrm{P}^{\mathrm{NP[log]}}$ [Hem89].

The *polynomial hierarchy* [MS72,Sto77] is defined as follows.

$$\begin{aligned} \Sigma^p_1 &= \mathrm{NP}, \\ \Sigma^p_{k+1} &= \mathrm{NP}^{\Sigma^p_k} \text{ (for } k \in \{1,2,3,\cdots\}\text{), and} \\ \mathrm{PH} &= \bigcup_{k\geq 1} \Sigma^p_k. \end{aligned}$$

P/poly [KL80] denotes the class of sets having small circuits.

For a nondeterministic polynomial-time Turing machine $M$, let $acc_M(x)$ ($rej_M(x)$) denote the number of accepting (rejecting) paths of $M$ on input $x$ and let $total_M(x)$ denote

the total number of paths of $M$ on input $x$. #P is the class of functions $f$ such that for some nondeterministic polynomial-time Turing machine $M$ it holds that $(\forall x)\,[f(x) = acc_M(x)]$.

## 2.1 Probabilistic and Threshold Computation

A *probabilistic polynomial-time Turing machine* [Gil77] is a nondeterministic polynomial-time Turing machine $M$ such that $M$ chooses with equal probability each of the nondeterministic choices at each choice point. $\Pr[M(x) = 1]$ denotes the probability weight of those paths on which $M$ accepts $x$ and $\Pr[M(x) = 0]$ denotes the probability weight of those paths on which $M$ rejects $x$.

We now define some complexity classes in terms of probabilistic polynomial-time Turing machines.

**Definition 2.1 [Probabilistic Classes]**

1. PP [Gil77] is the class of all sets $L$ such that there exists a probabilistic polynomial-time Turing machine $M$ such that for all $x \in \Sigma^*$ it holds that $\Pr[M(x) = L(x)] > 1/2$.
2. BPP [Gil77] is the class of all sets $L$ such that there exist a probabilistic polynomial-time Turing machine $M$ and an $\epsilon > 0$ such that for all $x \in \Sigma^*$ it holds that $\Pr[M(x) = L(x)] > 1/2 + \epsilon$.
3. R [Gil77] is the class of all sets $L$ such that there exists a probabilistic polynomial-time Turing machine $M$ such that for all $x \in \Sigma^*$ it holds that

$$\begin{aligned} x \in L &\implies \Pr[M(x) = 1] > 1/2, \text{ and} \\ x \notin L &\implies \Pr[M(x) = 0] = 1. \end{aligned}$$

By definition, we clearly have R $\subseteq$ BPP $\subseteq$ PP [Gil77].

The class PP can also be characterized as the class of sets $L$ such that there exist a nondeterministic polynomial-time Turing machine $M$ and a function $f \in$ FP such that for all $x \in \Sigma^*$ it holds that $x \in L \iff acc_M(x) \geq f(x)$.

By looking at the portion of accepting paths rather than the probability weight of the accepting paths, we now introduce the threshold analogs of the above probabilistic classes. Let $\#[M(x) = 1]$ denote $acc_M(x)$ and let $\#[M(x) = 0]$ denote $rej_M(x)$.

**Definition 2.2 [Threshold Classes]**

1. $\mathrm{PP}_{\mathrm{path}}$ [Sim75] is the class of all sets $L$ such that there exists a nondeterministic

polynomial-time Turing machine $M$ such that for all $x \in \Sigma^*$ it holds that $\#[M(x) = L(x)] > \frac{1}{2}\ total_M(x)$.

2. $\mathrm{BPP}_{\mathrm{path}}$ is the class of all sets $L$ such that there exist a nondeterministic polynomial-time Turing machine $M$ and an $\epsilon > 0$ such that for all $x \in \Sigma^*$ it holds that $\#[M(x) = L(x)] > (\frac{1}{2} + \epsilon)\ total_M(x)$.

3. $\mathrm{R}_{\mathrm{path}}$ is the class of all sets $L$ such that there exists a nondeterministic polynomial-time Turing machine $M$ such that for all $x \in \Sigma^*$ it holds that

$$\begin{aligned} x \in L &\implies acc_M(x) > \frac{1}{2}\ total_M(x), \text{ and} \\ x \notin L &\implies rej_M(x) = total_M(x). \end{aligned}$$

It is easy to see that $\mathrm{R}_{\mathrm{path}} \subseteq \mathrm{BPP}_{\mathrm{path}} \subseteq \mathrm{PP}_{\mathrm{path}}$. For all threshold classes in this paper, as a notational convenience we will place oracles above the word "path" (e.g., $\mathrm{BPP}_{\mathrm{path}}^{\mathrm{BPP}}$ denotes $(\mathrm{BPP}_{\mathrm{path}})^{\mathrm{BPP}}$).

It is known that R, BPP, and PP sets can be accepted via normalized probabilistic polynomial-time Turing machines: just extend each computation path of a given machine up to a fixed polynomial length and, on each new path, accept if the path that was extended accepted, and otherwise reject. The modified machine has the same acceptance probability as the original one. Observe that for normalized machines, the probabilistic interpretation of the machine accepts the same set as the threshold interpretation of the machine. Thus, each of the probabilistic classes is contained in the corresponding threshold class, i.e., $\mathrm{PP} \subseteq \mathrm{PP}_{\mathrm{path}}$, $\mathrm{BPP} \subseteq \mathrm{BPP}_{\mathrm{path}}$ and $\mathrm{R} \subseteq \mathrm{R}_{\mathrm{path}}$.

In fact, Simon [Sim75] has already shown that $\mathrm{PP}_{\mathrm{path}}$ is not a bigger class than PP. For completeness we give a proof here.

**Theorem 2.3 [Sim75]** $\mathrm{PP}_{\mathrm{path}} = \mathrm{PP}$.

**Proof:** It suffices to show $\mathrm{PP}_{\mathrm{path}} \subseteq \mathrm{PP}$. Let $L \in \mathrm{PP}_{\mathrm{path}}$ via $\mathrm{PP}_{\mathrm{path}}$ machine $M$ with polynomial $q$ bounding $M$'s runtime. Consider the machine $M'$ that on input $x$ extends each path $y$ of $M$ by appending a full binary subtree of depth $q(|x|) - |y|$. Furthermore, on the leftmost path of this appended subtree, $M'$ branches into two accepting (rejecting) paths, if $M$ accepted (rejected) on the path $y$. On each remaining path of the subtree, $M'$ branches into one accepting and one rejecting path.

$M'$ on input $x$ has $2^{q(|x|)+1}$ paths and, of these, $2^{q(|x|)} + acc_M(x) - rej_M(x)$ are accepting paths. This shows $L \in \mathrm{PP}$ via $M'$. ∎

Interestingly, this equivalence between probabilistic and threshold classes cannot hold for R and BPP unless the polynomial hierarchy collapses to its second level. This follows from the fact that NP is contained in $\mathrm{R}_{\mathrm{path}}$ and thus is also contained in $\mathrm{BPP}_{\mathrm{path}}$.

**Proposition 2.4** $\mathrm{R}_{\mathrm{path}} = \mathrm{NP}$.

**Proof:** $\mathrm{R}_{\mathrm{path}} \subseteq \mathrm{NP}$ is immediate from the definition. For the reverse inclusion, let $M$ be a nondeterministic Turing machine and let polynomial $p$ bound the runtime of $M$.

Consider the machine $M'$ that on input $x$ first simulates $M$ on input $x$, and if the simulation ends in an accepting path $y$, then $M'$ appends $2^{p(|x|)+1}$ accepting paths to $y$ and otherwise $M'$ rejects.

Now, more than half of all paths of $M'$ on input $x$ are accepting, if $x \in L(M)$, and $M'$ has no accepting paths, otherwise. This shows $L(M) \in \mathrm{R}_{\mathrm{path}}$. ∎

**Corollary 2.5** $\mathrm{NP} \subseteq \mathrm{BPP}_{\mathrm{path}}$.

It follows that if $\mathrm{BPP}_{\mathrm{path}}$ is equal to BPP, then $\mathrm{BPP}_{\mathrm{path}}$, and hence NP, has small circuits, which in turn, by the result of Karp, Lipton, and Sipser (see [KL80]), implies that the polynomial hierarchy collapses. So we cannot expect $\mathrm{BPP}_{\mathrm{path}}$ to have normalized machines. For different, contemporaneous work related to normalized versus non-normalized computation, see Hertrampf et al. [HLS+93] and Jenner, McKenzie, and Thérien [JMT94].

We have seen now that there are some crucial differences between BPP and its threshold analog, $\mathrm{BPP}_{\mathrm{path}}$. We will study $\mathrm{BPP}_{\mathrm{path}}$ in more detail in Section 3, and especially, we will strengthen Corollary 2.5.

## 2.2 Secure Computation

In this subsection and in Section 4, we study notions of secure adaptive access to databases in the presence of a powerful spying observer. We give below what we feel are the most natural definitions. In these definitions, we obtain security by requiring that an observer (seeing a path drawn uniformly from all the machine's paths) should learn nothing about the input string other than perhaps its length. For threshold computation, this notion is new. For probabilistic computation, the appendix section, Section 6, proves that this definition is equivalent to the notion of "one-oracle instance-hiding schemes that leak at most the length of their inputs" [BF90]. The original motivation for such classes, as explained for example by Beaver and Feigenbaum [BF90], is, very roughly, to study whether weak devices can solve hard problems by asking some powerful device questions in such a way that no observer can

tell which problem was actually solved by the weak device. Since NP $\subseteq$ BPP$_{\rm path}$, BPP$_{\rm path}$ clearly is not a computationally weak class. It nonetheless makes sense to consider the same interactive model in the case that applies here: studying whether a relatively powerful class (BPP$_{\rm path}$) can use a (potentially powerful) information source while shielding information on the problem being solved even from extremely powerful observers.

**Definition 2.6 [Secure Threshold Computation]** For any set $D$, a set $A$ is said to be in secureBPP$^D_{\rm path}$ (that is, is said to be "securely accepted by a bounded-error threshold polynomial-time machine via access to database $D$") if there is a nondeterministic polynomial-time Turing machine $N$ such that:

1. [$A \in$ BPP$^D_{\rm path}$ via machine $N$] There exists an $\epsilon > 0$ such that for all $x \in \Sigma^*$ it holds that $\#[N^D(x) = A(x)] > (1/2 + \epsilon)\, total_{N^D}(x)$ (see Part 2 of Definition 2.2).

2. [The queries of $N^D$ reveal no information to an observer other than perhaps the length of the input] For every $k \in \{0, 1, 2, \cdots\}$, and every vector $\boldsymbol{v} = (v_1, v_2, \cdots, v_k)$, $v_1, v_2, \cdots, v_k \in \Sigma^*$, and every pair of strings $x \in \Sigma^*$ and $y \in \Sigma^*$ such that $|x| = |y|$, it holds that

$$\frac{path\text{-}occurances_{N^D(x)}(\boldsymbol{v})}{total_{N^D}(x)} = \frac{path\text{-}occurances_{N^D(y)}(\boldsymbol{v})}{total_{N^D}(y)},$$

   where $path\text{-}occurances_{N^D(z)}(\boldsymbol{v}) = ||\{p \mid p$ is a path of $N^D(z)$ on which $\boldsymbol{v}$ is the sequence of queries asked to the oracle (in the order asked, possibly with duplications if the same query is asked more than once)[1]$\}||$.

Similarly, for the probabilistic class BPP, we have the following definition of secure access.

**Definition 2.7 [Secure Probabilistic Computation]** For any set $D$, a set $A$ is said to be in secureBPP$^D$ (that is, is said to be "securely accepted by a bounded-error probabilistic polynomial-time machine via access to database $D$") if there is a probabilistic polynomial-time Turing machine $N$ such that:

1. [$A \in$ BPP$^D$ via machine $N$] There exists an $\epsilon > 0$ such that for all $x \in \Sigma^*$ it holds that $\Pr[N^D(x) = A(x)] > 1/2 + \epsilon$ (see Part 2 of Definition 2.1).

2. [The queries of $N^D$ reveal no information to an observer other than perhaps the length of the input] For every $k \in \{0, 1, 2, \cdots\}$, and every vector $\boldsymbol{v} = (v_1, v_2, \cdots, v_k)$,

[1] Henceforward, we'll refer to this as a *query vector*.

$v_1, v_2, \cdots, v_k \in \Sigma^*$, and every pair of strings $x \in \Sigma^*$ and $y \in \Sigma^*$ such that $|x| = |y|$, it holds that

$$\Pr[\text{the query vector of } N^D(x) \text{ is } \boldsymbol{v}] = \Pr[\text{the query vector of } N^D(y) \text{ is } \boldsymbol{v}].$$

Oblivious self-reducibility was discussed in [FFLS92], and we now define complexity classes capturing the notion of oblivious access.

**Definition 2.8 [Oblivious Probabilistic and Threshold Classes]** For any set $D$, a set $A$ is said to be in obliviousBPP$^D_{\text{path}}$ (respectively, obliviousBPP$^D$) if there is a nondeterministic (respectively, probabilistic) polynomial-time Turing machine $N$ such that:

1. [$A \in \text{BPP}^D_{\text{path}}$ (respectively, $A \in \text{BPP}^D$) via machine $N$] There exists an $\epsilon > 0$ such that for all $x \in \Sigma^*$ it holds that $\#[N^D(x) = A(x)] > (1/2+\epsilon)\, total_{N^D}(x)$ (respectively, $\Pr[N^D(x) = A(x)] > 1/2 + \epsilon$).
2. $N$ is an oblivious machine in the sense that on an input $z$ it initially is given access to a "pre-input" tape on which $0^{|z|}$ is written. $N$ then performs its adaptive queries to $D$. Then, after making all queries to $D$, machine $N$ is given access to $z$.

We clearly have that, for every $D$,

$$\text{BPP}^D_{\text{path}} \supseteq \text{secureBPP}^D_{\text{path}} \supseteq \text{obliviousBPP}^D_{\text{path}}, \text{ and}$$

$$\text{BPP}^D \supseteq \text{secureBPP}^D \supseteq \text{obliviousBPP}^D.$$

Are these inclusions proper? In other words, does using security against observers as the *definition* of secure computation (secureBPP$^D_{\text{path}}$, secureBPP$^D$) yield a more flexible notion of security than does blinding the machine to its input (obliviousBPP$^D_{\text{path}}$, obliviousBPP$^D$)? Formally, is obliviousBPP$^D_{\text{path}} \neq$ secureBPP$^D_{\text{path}}$ or obliviousBPP$^D \neq$ secureBPP$^D$? Our intuition says that both inequalities hold. However, Section 4 shows that establishing that "yes" is the answer implies that P $\neq$ PSPACE (and even implies the stronger result that BPP $\neq$ PP). Since it is commonly believed that P $\neq$ PSPACE, this does not provide evidence that equality holds; rather, it merely suggests that witnessing a separation will be hard with current techniques. We note that results (such as Theorem 4.1 and Corollary 4.2) that connect the existence of an oracle separation to the existence of a real-world separation (see, e.g., the survey [Boo89]) usually occur in cases in which the oracle is tremendously restricted (e.g., to the class of tally sets or the class of sparse sets [BBS86,LS86]); in contrast, Section 4 provides such a relativization result that applies without restriction of the database $D$.

Note that we could also define classes, partially-secure-$\mathrm{BPP}^D_{\mathrm{path}}$ and partially-secure-$\mathrm{BPP}^D$, based on the notion (see, e.g., [FFLS92] and the papers cited therein) that an observer watching *one* query should get no information other than perhaps about the length (clearly, for all $D$, $\mathrm{BPP}^D_{\mathrm{path}} \supseteq$ partially-secure-$\mathrm{BPP}^D_{\mathrm{path}} \supseteq \mathrm{secureBPP}^D_{\mathrm{path}}$ and $\mathrm{BPP}^D \supseteq$ partially-secure-$\mathrm{BPP}^D \supseteq \mathrm{secureBPP}^D$), and, more generally, one could study a variety of classes between $\mathrm{BPP}^D$ and $\mathrm{secureBPP}^D$ (or between $\mathrm{BPP}^D_{\mathrm{path}}$ and $\mathrm{secureBPP}^D_{\mathrm{path}}$) based upon security against observers using various strengths of query access (for example, one could require security against observers who could see two queries, or against observers who could make $\mathcal{O}(\log n)$ adaptive queries into the query vector, or so on). However, we restrict our attention to what we feel are the most natural security classes: $\mathrm{secureBPP}^D$ and $\mathrm{secureBPP}^D_{\mathrm{path}}$.

There is no point in defining security classes for unbounded-error computation, as it is easy to see that, for every $D$, $\mathrm{PP}^D = \mathrm{securePP}^D = \mathrm{obliviousPP}^D = \mathrm{PP}^D_{\mathrm{path}} = \mathrm{securePP}^D_{\mathrm{path}} = \mathrm{obliviousPP}^D_{\mathrm{path}}$.

Finally, we note that all sets that are accepted by an oblivious machine relative to some database $D$ have small circuits. Let $\mathrm{obliviousBPP}^\star$ denote $\bigcup_{D \in 2^{\Sigma^*}} \mathrm{obliviousBPP}^D$. We have the following result.

**Proposition 2.9** $\mathrm{obliviousBPP}^\star = \mathrm{P/poly}$.

**Corollary 2.10** $(\exists L)\,[L \notin \mathrm{obliviousBPP}^L]$.

Though for most common classes $\mathcal{C}$ it holds that $(\forall L)\,[L \in \mathcal{C}^L]$, Corollary 2.10 should not be surprising; it is natural that weak machines, when accepting a hard set via a hard database, may leak some information to an observer. Interestingly, a similar result holds for secure computation. Namely, Abadi, Feigenbaum, and Kilian [AFK89] have shown that $\mathrm{secureBPP}^D \subseteq \mathrm{NP/poly} \cap \mathrm{coNP/poly}$ for any database $D$. Thus, for any set $D$, no NP-hard set is in $\mathrm{secureBPP}^D$ unless the polynomial hierarchy collapses.

# 3 $\mathbf{BPP_{path}}$

We have already argued that BPP and $\mathrm{BPP}_{\mathrm{path}}$ differ unless the polynomial hierarchy collapses. These classes nonetheless share certain properties. For example, as is also the case for BPP [ZH86], $\mathrm{BPP}_{\mathrm{path}}$ has a strong amplification property.

**Theorem 3.1** Let $L$ be in $\mathrm{BPP}_{\mathrm{path}}$. For each polynomial $q$, there is a nondeterministic polynomial-time Turing machine $M$ such that for all $x \in \Sigma^*$ it holds that

$$\#[M(x) = L(x)] > \left(1 - 2^{-q(|x|)}\right) \ total_M(x).$$

The proof is analogous to the corresponding proof for BPP.

BPP is closed under Turing reductions [Ko82,Zac82]. However, no relativizable proof can establish the closure of $\mathrm{BPP}_{\mathrm{path}}$ under Turing reductions. In particular, Beigel [Bei92] constructed an oracle relative to which $\mathrm{P}^{\mathrm{NP}}$ is not contained in PP. Since $\mathrm{BPP}_{\mathrm{path}}$ clearly is contained in PP (and the proof relativizes), it follows that, relative to the same oracle, $\mathrm{P}^{\mathrm{NP}}$ is not contained in $\mathrm{BPP}_{\mathrm{path}}$, and hence, since $\mathrm{NP} \subseteq \mathrm{BPP}_{\mathrm{path}}$, $\mathrm{BPP}_{\mathrm{path}}$ is not closed under Turing reductions relative to this oracle. That is, there exists an $A$ such that $\mathrm{BPP}^A_{\mathrm{path}} \neq \mathrm{P}^{\mathrm{BPP}^A_{\mathrm{path}}}$.

For $\mathrm{BPP}_{\mathrm{path}}$, we can prove closure under truth-table reductions,

**Theorem 3.2** $\mathrm{BPP}_{\mathrm{path}}$ is closed under polynomial-time truth-table reductions.

**Proof:** Let $A \leq^p_{tt} B$ for $B \in \mathrm{BPP}_{\mathrm{path}}$, i.e., there exists a polynomial-time Turing machine $M$ such that $L = L(M^B)$ and, for each input $x$ of length $n$, machine $M$ makes at most $q(n)$ queries (nonadaptively) to $B$. Without loss of generality, we may assume that all queries have the same length $l(n)$, $l(n) \geq n$, and that $q(n)$ is a nondecreasing function.

Let $N$ be a $\mathrm{BPP}_{\mathrm{path}}$ machine for $B$, such that on input $y$

$$\#[N(y) = B(y)] > \left(1 - \frac{1}{3q(|y|)}\right) \ total_N(y).$$

Consider the machine $M'$ that on input $x$, $|x| = n$, computes $y_1, \ldots, y_k$, the truth-table queries of $M$ on input $x$, where $k \leq q(n)$, and, for each query $y_i$, machine $M'$ guesses a path of $N$ on input $y_i$ and takes the output of this path as the answer to query $y_i$, for $i = 1, \ldots, k$. Using these answers instead of the oracle $B$, $M'$ simulates $M$ on input $x$ and outputs the result.

$M'$ has $total_{M'}(x) = \prod_{i=1}^{k} total_N(y_i)$ paths. At least on those paths on which all the answers to the oracle queries are correct, $M'$ decides correctly whether $x$ is in $A$, i.e., we have

$$\begin{aligned} \#[M'(x) = A(x)] &\geq \prod_{i=1}^{k} \#[N(y_i) = B(y_i)] \\ &\geq \prod_{i=1}^{k} \left(1 - \frac{1}{3q(l(n))}\right) total_N(y_i) \qquad \text{by assumption} \end{aligned}$$

$$\begin{aligned} &\geq \left(1 - \frac{1}{3q(n)}\right)^k \prod_{i=1}^{k} total_N(y_i) \qquad \text{since } l(n) \geq n \\ &\geq \left(1 - \frac{k}{3q(n)}\right) \prod_{i=1}^{k} total_N(y_i) \\ &\geq \frac{2}{3}\, total_{M'}(x). \end{aligned}$$

This shows that $A \in \mathrm{BPP}_{\mathrm{path}}$. ∎

**Corollary 3.3** $\mathrm{BPP}_{\mathrm{path}}$ is closed under complementation, intersection, and union.

Since NP is contained in $\mathrm{BPP}_{\mathrm{path}}$, it follows that the closure of NP under truth-table reductions is contained in $\mathrm{BPP}_{\mathrm{path}}$.

**Corollary 3.4** $\mathrm{P}^{\mathrm{NP[log]}} \subseteq \mathrm{BPP}_{\mathrm{path}}$.

It is known that BPP is low for PP [KSTT92] and for itself [Ko82,Zac82], i.e., $\mathrm{PP}^{\mathrm{BPP}} = \mathrm{PP}$ and $\mathrm{BPP}^{\mathrm{BPP}} = \mathrm{BPP}$. We show in the next theorem that BPP is also low for $\mathrm{BPP}_{\mathrm{path}}$. Observe that relative to Beigel's previously mentioned oracle making $\mathrm{P}^{\mathrm{NP}}$ not contained in PP, we must also have that NP, and hence $\mathrm{BPP}_{\mathrm{path}}$, cannot be low for PP. That is, there exists an $A$ such that $\mathrm{PP}^{\mathrm{BPP}^A_{\mathrm{path}}} \neq \mathrm{PP}^A$. Furthermore, by an easy induction, we have that if $\mathrm{BPP}_{\mathrm{path}}$ is low for itself then the polynomial hierarchy, PH, is contained in $\mathrm{BPP}_{\mathrm{path}}$. But, as we will see in Theorem 3.11 below, $\mathrm{BPP}_{\mathrm{path}}$ is contained in some level of the polynomial hierarchy. Thus, $\mathrm{BPP}_{\mathrm{path}}$ is not low for $\mathrm{BPP}_{\mathrm{path}}$ unless the polynomial hierarchy collapses.

**Theorem 3.5** $\mathrm{BPP}^{\mathrm{BPP}}_{\mathrm{path}} = \mathrm{BPP}_{\mathrm{path}}$.

**Proof:** Let $L \in \mathrm{BPP}^{\mathrm{BPP}}_{\mathrm{path}}$ via a machine $M$ and a set $A \in \mathrm{BPP}$ such that polynomial $p$ bounds the runtime of $M^A$ and for all $x \in \Sigma^*$ it holds that $\#[M^A(x) = L(x)] > \frac{7}{8}\ total_{M^A}(x)$.

Let $B = \{\, (0^n, w_1, a_1, \ldots, w_k, a_k) \mid k \leq p(n) \text{ and } (\forall i : 1 \leq i \leq k)[|w_i| \leq p(n) \text{ and } A(w_i) = a_i]\,\}$. Since BPP is closed under truth-table reductions [Ko82,Zac82], $B \in \mathrm{BPP}$. Hence, there exist a probabilistic polynomial-time Turing machine $M_B$ and a polynomial $q$ such that, for any input $z = (0^n, w_1, a_1, \ldots, w_k, a_k)$, $M_B$'s error probability is bounded by $2^{-(p(n)+4)}$ and $M_B$'s computation tree is a full binary tree with $total_{M_B}(z) = 2^{q(n)}$.

Consider the machine $M'$ that on input $x$, $|x| = n$, performs the following steps.

1. $M'$ simulates $M^A$ on input $x$. Whenever $M$ queries the oracle, $M'$ nondeterministically guesses the answer. Let $(w_1, a_1), \ldots, (w_k, a_k)$ be the sequence of queried strings and guessed answers along a computation path $y$.

2. To verify the guessed answers, $M'$ simulates $M_B$ on input $(0^n, w_1, a_1, \ldots, w_k, a_k)$.

3. $M'$ amplifies the output of $M$ on path $y$ from the first step if the guessed answers there are certified in the second step. More precisely, $M'$ now appends $2^{p(n)+4}$ accepting (rejecting) paths if path $y$ was accepting (rejecting) and the simulation in the second step ended in an accepting path of $M_B$. Otherwise, $M'$ rejects.

After the first two steps, $M'$ has at most $2^{p(n)}\,2^{q(n)}$ computation paths. In the last step, $M'$ amplifies all paths (a) in which the guessed oracle answers are correct and that are certified by $M_B$ in the second step, i.e., at most $total_{M^A}(x)\,2^{q(n)}$ paths, and (b) all paths in which the guessed oracle answers are false but are wrongly certified by $M_B$, i.e., at most $2^{p(n)}\,2^{-(p(n)+4)}\,2^{q(n)}$ paths. So we have

$$total_{M'}(x) \;\leq\; 2^{p(n)}\,2^{q(n)} \;+\; 2^{p(n)+4}\left(total_{M^A}(x)\,2^{q(n)} \;+\; 2^{p(n)}\,2^{-(p(n)+4)}\,2^{q(n)}\right).$$

The paths on which $M'$ decides correctly include at least those paths that correspond to correct paths of $M$ in the first step and are subsequently certified in the second step. Since these paths are amplified in the last step, we have

$$\#[M'(x) = L(x)] \;\geq\; \left(\frac{7}{8}\,total_{M^A}(x)\right)\;\left((1-2^{-(p(n)+4)})\,2^{q(n)}\right)\;2^{p(n)+4}.$$

Now, it is not hard to see that $\#[M'(x) = L(x)] > \frac{2}{3}\;total_{M'}(x)$. Thus, $L \in \mathrm{BPP}_{\mathrm{path}}$. ∎

If we define a function class $\mathrm{FBPP}_{\mathrm{path}}$ in the natural manner (see the analogous class FBPP of Ko [Ko82]), then it is not hard to see that the same proof technique also establishes that $\mathrm{FBPP}^{\mathrm{BPP}}_{\mathrm{path}} = \mathrm{FBPP}_{\mathrm{path}}$.

**Corollary 3.6** $\mathrm{NP}^{\mathrm{BPP}} \subseteq \mathrm{BPP}_{\mathrm{path}}$.

Indeed, we even have $\mathrm{P}^{\mathrm{NP}^{\mathrm{BPP}}[\log]\oplus\mathrm{BPP}} \subseteq \mathrm{BPP}_{\mathrm{path}}$.

Babai [Bab85] introduced the Arthur-Merlin classes MA and AM. It is known that $\mathrm{NP}^{\mathrm{BPP}} \subseteq \mathrm{MA} \subseteq \mathrm{AM} \subseteq \mathrm{BPP}^{\mathrm{NP}}$ [Bab85,Zac88]. It is not known whether any of the inclusions is strict or not, though various relevant oracle separations are known (e.g., Fenner et al. [FFKL93] have constructed an oracle world in which $\mathrm{NP}^{\mathrm{BPP}}$ and MA differ). Below, we strengthen Corollary 3.6 to show that even MA is contained in $\mathrm{BPP}_{\mathrm{path}}$. This improves the result of Vereshchagin [Ver92] that $\mathrm{MA} \subseteq \mathrm{PP}$.

**Theorem 3.7** $\mathrm{MA} \subseteq \mathrm{BPP}_{\mathrm{path}}$.

**Proof:** Let $L \in \mathrm{MA}$. By standard amplification technique, there exist a polynomial-time predicate $Q$ and polynomials $p$ and $q$ such that for all $x \in \Sigma^*$

$$\begin{aligned} x \in L &\implies (\exists y \in \Sigma^{p(|x|)})\ [\Pr[Q(x,y,z)] > 1 - 2^{-(p(|x|)+4)}], \\ x \notin L &\implies (\forall y \in \Sigma^{p(|x|)})\ [\Pr[Q(x,y,z)] < 2^{-(p(|x|)+4)}], \end{aligned}$$

where the probability is taken uniformly over all $z \in \Sigma^{q(|x|)}$.

Consider the machine $M$ that on input $x$ guesses $y \in \Sigma^{p(|x|)}$ and $z \in \Sigma^{q(|x|)}$, and if $Q(x,y,z)$ is false $M$ rejects, otherwise, $M$ produces $2^{p(|x|)+2}$ accepting paths.

It is not difficult to see that $\#[M(x) = L(x)] > \frac{2}{3}\ total_M(x)$. Thus, $L \in \mathrm{BPP}_{\mathrm{path}}$. ∎

It is an open question whether AM is contained in $\mathrm{BPP}_{\mathrm{path}}$. Vereshchagin [Ver92] constructed an oracle $A$ such that relative to $A$ the class AM is not a subset of PP, i.e., $\mathrm{AM}^A \not\subseteq \mathrm{PP}^A$. Thus, AM is not a subset of $\mathrm{BPP}_{\mathrm{path}}$ relative to $A$. On the other hand, $\mathrm{BPP}_{\mathrm{path}}$ is not a subset of AM unless the polynomial hierarchy collapses. This follows from the result of Boppana, Håstad, and Zachos [BHZ87] that if coNP $\subseteq$ AM then the polynomial hierarchy collapses to its second level. Since coNP $\subseteq \mathrm{BPP}_{\mathrm{path}}$, we get the same consequence from the assumption that $\mathrm{BPP}_{\mathrm{path}}$ is contained in AM.

Sipser and Gács ([Sip83], see also [Lau83]) showed that BPP $\subseteq \mathrm{R}^{\mathrm{NP}}$. It is an open question whether the same inclusion holds for $\mathrm{BPP}_{\mathrm{path}}$. However, we show that $\mathrm{BPP}_{\mathrm{path}} \subseteq \mathrm{BPP}^{\mathrm{NP}}$. As a first step, we show that a $\mathrm{BPP}_{\mathrm{path}}$ set can be decided by a deterministic polynomial-time Turing machine making logarithmically many queries to a $\Sigma_2^p$ oracle, and hence $\mathrm{BPP}_{\mathrm{path}}$ is in the polynomial hierarchy. A randomized version of this algorithm can decide a $\mathrm{BPP}_{\mathrm{path}}$ set with an NP oracle. The proof applies Sipser's Coding Lemma for universal hashing [Sip83].

We mention that we could get a shorter proof by applying the results of Stockmeyer [Sto85] to approximate #P functions and of Jerrum, Valiant, and Vazirani [JVV86], who showed a probabilistic version of Stockmeyer's theorem. However, we prefer to give a self-contained proof here, thereby encouraging the reader to see whether he or she can improve our result, for example, by getting a one-sided error probabilistic algorithm (in Part 2 of Theorem 3.11). Since there is an oracle relative to which BPP is not contained in $\mathrm{P}^{\mathrm{NP}}$ [Sto85], one cannot obtain a deterministic algorithm with relativizable techniques.

**Definition 3.8 [Sip83]** Let $X \subseteq \Sigma^m$ and let $H_1, \ldots, H_k : \Sigma^m \to \Sigma^k$ be a collection of linear functions given as $k \times m$ 0-1 matrices. The predicates *Separate* and *Hash* are defined

as follows.

1. $Separate_X(H_1, \ldots, H_k) \iff (\forall\, y \in X)(\exists\, i : 1 \leq i \leq k)(\forall\, z \in X : y \neq z)\,[H_i(y) \neq H_i(z)]$, where $H_i(y)$ means multiplication of the $k \times m$ matrix $H_i$ with the $m$ vector $y$, yielding a $k$ vector, with the arithmetic done in GF[2].

2. $Hash_X(k) \iff (\exists H_1, \ldots, H_k \in \Sigma^{km})\,[Separate_X(H_1, \ldots, H_k)]$.

The intuition about predicate *Hash* is that the size of the range of the hash functions (which is determined by $k$) has to be sufficiently large, with respect to the size of $X$, for a collection $H_1, \ldots, H_k$ that separates $X$ to exist.

**Lemma 3.9 [Sip83]** Let $X \subseteq \Sigma^m$ and let $k = \lfloor \log \|X\| \rfloor + 2$. For a random collection of functions $H_1, \ldots, H_k : \Sigma^m \to \Sigma^k$,

$$\Pr[Separate_X(H_1, \ldots, H_k)] \;\geq\; \frac{7}{8}.$$

As a consequence of this lemma, we get a lower bound for the size of a set $X$. The upper bound follows by the pigeon hole principle (see [Sto85]).

**Corollary 3.10 [Sip83]** If $X \subseteq \Sigma^m$ and $k_X$ is the smallest $k$ such that $Hash_X(k)$ is true, then $2^{k_X - 3} \leq \|X\| \leq k_X\, 2^{k_X}$.

**Theorem 3.11**

1. $\mathrm{BPP}_{\mathrm{path}} \subseteq \mathrm{P}^{\Sigma_2^p[\log]}$.

2. $\mathrm{BPP}_{\mathrm{path}} \subseteq \mathrm{BPP}^{\mathrm{NP}}$.

**Proof:** Let $L \in \mathrm{BPP}_{\mathrm{path}}$. There exist a nondeterministic Turing machine $M$ and a polynomial $p$ that bounds the runtime of $M$ such that for all $x \in \Sigma^*$ it holds that $\#[M(x) = L(x)] > (1 - 2^{-|x|})\ total_M(x)$.

Sipser's proof that $\mathrm{BPP} \subseteq \Sigma_2^p$ uses the fact that $total_M(x)$ is known *a priori*. However, here we have only an upper bound.

Fix $x \in \Sigma^*$; let $n$ denote $|x|$. Define

$$\begin{aligned} A &= \{\, y\, 0^{p(n)-|y|} \mid y \text{ is an accepting computation of } M \text{ on input } x \,\} \text{ and} \\ R &= \{\, y\, 0^{p(n)-|y|} \mid y \text{ is a rejecting computation of } M \text{ on input } x \,\}. \end{aligned}$$

Clearly, $\|A\| = acc_M(x)$ and $\|R\| = rej_M(x)$.

Observe that *Separate* is a coNP predicate in $x$ and the hash functions $H_1, \ldots, H_k$ when applied to $A$ or $R$, and *Hash* is a $\Sigma_2^p$ predicate in $x$ and $k$.

Let $k_A$ ($k_R$) denote the minimal $k$ such that $Hash_A(k)$ ($Hash_R(k)$) is true. $k_A$ and $k_R$ can be computed by a binary search making at most $\log p(n)$ many queries to $Hash_A$ and $Hash_R(k)$, respectively. From Corollary 3.10, it follows that $2^{k_A-3} \leq acc_M(x) \leq k_A\, 2^{k_A}$ and that $2^{k_R-3} \leq rej_M(x) \leq k_R 2^{k_R}$. Now it is not difficult to see that for all but finitely many $x$ we have $x \in L \iff k_R < k_A$. This proves $L \in \mathrm{P}^{\Sigma_2^p[\log]}$.

Next, we show that $L \in \mathrm{BPP}^{\mathrm{NP}}$. Consider the following probabilistic procedure, which tries to approximate $k_A$ and $k_R$ by randomly generating a collection of functions $H_1, \ldots, H_k$, and directly asking the oracle $Separate_X$ about $(H_1, \ldots, H_k)$, for a given set $X$ and increasing $k$.

```
APPROXIMATE(x, X)
k ← 0
repeat
   k ← k + 1
   randomly choose H_1, ..., H_k
until Separate_X(H_1, ..., H_k) or k = p(n)
return k
```

The following main algorithm decides whether $x$ is in $L$, and is correct with high probability.

```
MAIN(x)
k_a ← APPROXIMATE(x, A)
k_r ← APPROXIMATE(x, R)
if k_a > k_r then accept
else reject.
```

By the definition of $k_A$, we always have $k_A \leq k_a$. Note that, by the upper bound of Corollary 3.10 and since $k_A \leq p(n)$, it follows that $\log(\|A\|/p(n)) \leq k_A$. From Lemma 3.9, it follows that $k_a \leq \lfloor \log \|A\| \rfloor + 2$ holds with probability at least 7/8. Since the same bounds hold for $k_r$, we have that with probability at least 3/4 it holds that both (a) $\log \frac{acc_M(x)}{p(n)} \leq k_a \leq \log acc_M(x) + 2$, and (b) $\log \frac{rej_M(x)}{p(n)} \leq k_r \leq \log rej_M(x) + 2$. This implies that for all but finitely many $x$ it holds that $x \in L \iff k_a > k_r$, with probability at least 3/4. Thus, $L \in \mathrm{BPP}^{\mathrm{NP}}$. ∎

As already mentioned before Theorem 3.5, $\mathrm{BPP}_{\mathrm{path}}$ cannot be low for itself unless the polynomial hierarchy collapses to $\mathrm{BPP}_{\mathrm{path}}$. From Theorem 3.11 we thus have the following claim.

**Corollary 3.12** If $\mathrm{BPP}_{\mathrm{path}}^{\mathrm{BPP}_{\mathrm{path}}} = \mathrm{BPP}_{\mathrm{path}}$ then $\mathrm{PH} = \mathrm{P}^{\Sigma_2^p[\log]}$.

Zachos [Zac88] has shown that $\mathrm{NP} \subseteq \mathrm{BPP}$ implies $\mathrm{PH} = \mathrm{BPP}$. Since this result relativizes (i.e., for all $A$, $\mathrm{NP}^A \subseteq \mathrm{BPP}^A$ implies $\mathrm{PH}^A = \mathrm{BPP}^A$), we obtain the following corollary from Theorem 3.11.

**Corollary 3.13** $\Sigma_2^p \subseteq \mathrm{BPP}_{\mathrm{path}} \Longrightarrow \mathrm{PH} = \mathrm{BPP}^{\mathrm{NP}}$.

Toda [Tod91] and Toda and Ogiwara [TO92] showed that $\mathrm{PH} \subseteq \mathrm{BPP}^{\mathcal{C}}$ for any class $\mathcal{C}$ among $\{\mathrm{PP}, \mathrm{C}_{=}\mathrm{P}, \oplus\mathrm{P}\}$. As a consequence, none of these classes can be contained in the polynomial hierarchy unless the polynomial hierarchy collapses. Thus, none of these classes can be contained in $\mathrm{BPP}_{\mathrm{path}}$ unless the polynomial hierarchy collapses.

Ogiwara and Hemachandra [OH93] and Fenner, Fortnow, and Kurtz [FFK94] independently defined the counting class SPP as follows.

**Definition 3.14 [OH93,FFK94]** SPP is the class of all sets $L$ such that there exist a nondeterministic polynomial-time Turing machine $M$ and an FP function $f$ such that for all $x \in \Sigma^*$ it holds that

$$\begin{aligned} x \in L &\Longrightarrow acc_M(x) = f(x) + 1, \text{ and} \\ x \notin L &\Longrightarrow acc_M(x) = f(x). \end{aligned}$$

Fenner, Fortnow, and Kurtz [FFK94] argue that SPP is, in some sense, the smallest class that is definable in terms of the number of accepting and rejecting computations. In particular, SPP is low for $\mathrm{PP}, \mathrm{C}_{=}\mathrm{P}$, and $\oplus\mathrm{P}$ [FFK94]. Though it is an open question whether SPP is contained in $\mathrm{BPP}_{\mathrm{path}}$, there is an oracle relative to which this is not the case.[2]

**Theorem 3.15** There is an oracle $A$ such that $\mathrm{SPP}^A \not\subseteq \mathrm{BPP}_{\mathrm{path}}^A$.

**Proof:** Let $M_1, M_2, \ldots$ be an enumeration of nondeterministic polynomial-time Turing machines and let $p_1, p_2, \ldots$ be an enumeration of polynomials such that polynomial $p_i$

[2] Very recently, Fortnow [For94] has improved our result by constructing an oracle relative to which SPP is not contained in the polynomial hierarchy.

bounds the runtime of machine $M_i$. Without loss of generality, we assume $p_i(n) = n^i + i$. Let $s(i)$, $i = 1, 2, \ldots$, be a sequence of integers defined by $s(1) = 5$ and, for $i > 1$, $s(i+1) = 2^{s(i)}$.

We define the test language

$$L(A) = \{\, 1^n \mid (\exists j)\,[n = s(j) \text{ and } \|A^{=n}\| = 2^{n-1}]\,\}.$$

Below, we will construct a set $A$ such that for every $i \geq 1$, $\|A^{=s(i)}\|$ is either $2^{s(i)-1}$ or $2^{s(i)-1} - 1$. For such an $A$, we have $L(A) \in \mathrm{SPP}^A$. Furthermore, we will construct $A$ such that, for each $i \geq 1$, at least one of the following requirements holds.

(R1) $M_i^A$ is not a $\mathrm{BPP}^A_{\mathrm{path}}$ machine. That is, there exists an $x \in \Sigma^*$ such that

$$\frac{1}{4}\, total_{M_i^A}(x) \leq acc_{M_i^A}(x) \leq \frac{3}{4}\, total_{M_i^A}(x).$$

(R2) There exists an $n \geq 1$ such that $M_i^A(1^n)$ accepts if and only if $1^n \notin L(A)$.

It follows from Theorem 3.1 that the existence of such an oracle establishes the theorem.

We construct the set $A$ in stages. In stage $i$, we diagonalize against machine $M_i$. Initially, $i = 1$ and $A_1 = \emptyset$.

*Stage* $i$. Let $n = s(i)$. We will add only strings of length $n$ to $A_i$. Since $p_j(s(j)) < n$ for all $j < i$, this will not effect the construction done in earlier stages.

Define

$$\begin{aligned} \mathcal{A} &= \{\, A_i \cup Z \mid Z \subseteq \Sigma^n \text{ and } \|Z\| = 2^{n-1} \,\} \text{ and} \\ \mathcal{B} &= \{\, A_i \cup Z \mid Z \subseteq \Sigma^n \text{ and } \|Z\| = 2^{n-1} - 1 \,\}. \end{aligned}$$

If there is a set $X \in \mathcal{A} \cup \mathcal{B}$ such that $X$ fulfills requirement (R1), i.e., $M_i^X$ is not a $\mathrm{BPP}^X_{\mathrm{path}}$ machine, then define $A_{i+1} = X$ and go to the next stage. Otherwise, we show that there is a set in $\mathcal{A} \cup \mathcal{B}$ such that requirement (R2) is fulfilled.

Let $X$ be a set such that the number of paths of $M_i^X$ on input $1^n$ is maximal for all $X \in \mathcal{A} \cup \mathcal{B}$. That is, we have

$$(\forall\, Y \in \mathcal{A} \cup \mathcal{B})\ [total_{M_i^Y}(1^n) \leq total_{M_i^X}(1^n)]. \qquad (\star)$$

Suppose $X \in \mathcal{A}$. If $1^n \notin L(M_i^X)$ then we are done since $1^n \in L(X)$. So suppose that $1^n \in L(M_i^X)$. For $w \in X \cap \Sigma^n$, define $X_w = X - \{w\}$. By definition, $1^n \notin L(X_w)$. We claim that there exists a $w \in X \cap \Sigma^n$ such that $1^n \in L(M_i^{X_w})$. For such a $w$, define $A_{i+1} = X_w$. Then requirement (R2) is fulfilled.

To prove our claim, assume that, for all $w \in X \cap \Sigma^n$, it holds that $1^n \notin L(M_i^{X_w})$. By taking $w$ out of $X$, at least $acc_{M_i^X}(1^n) - acc_{M_i^{X_w}}(1^n)$ accepting paths of $M$ either change to rejecting paths or disappear, and hence $w$ must have been queried on those paths. Since

$$\begin{aligned} acc_{M_i^X}(1^n) - acc_{M_i^{X_w}}(1^n) &\geq \frac{3}{4}\, total_{M_i^X}(1^n) - \frac{1}{4}\, total_{M_i^{X_w}}(1^n) \\ &\geq \frac{1}{2}\, total_{M_i^X}(1^n) \qquad \text{by } (\star), \end{aligned}$$

each $w \in X \cap \Sigma^n$ is queried by $M_i^X$ on input $1^n$ on at least half of all paths. Thus, $M_i^X$ asks at least $2^{n-1}\,\frac{1}{2}\, total_{M_i^X}(1^n) = 2^{n-2}\, total_{M_i^X}(1^n)$ queries to its oracle. On the other hand, $M_i^X$ cannot ask more than $p_i(n)\, total_{M_i^X}(1^n)$ queries to its oracle. Since $p_i(n) < 2^{n-2}$, this yields a contradiction.

The case $X \in \mathcal{B}$ is symmetric. Here, one has to define $X_w$ by *adding* a string $w \in \Sigma^n - X$ to $X$, and then, in case $1^n \in L(M_i^{X_w})$ for all $w \in \Sigma^n - X$, argue regarding the number of rejecting instead of accepting paths of $M_i$. ∎

# 4 If Secure and Oblivious Computation Differ, then P ≠ PSPACE

We show, for both threshold and probabilistic computation, that secure computation is more powerful than oblivious computation only if BPP ≠ PP (which would resolve in the affirmative the important question of whether polynomial time differs from polynomial space).

**Theorem 4.1** If there is a database $D$ such that $\text{secureBPP}_{\text{path}}^D \neq \text{obliviousBPP}_{\text{path}}^D$, then BPP ≠ PP.

**Proof:** Assume BPP = PP. Note that this implies that $\text{BPP} = \text{P}^{\#\text{P}}$ (since $\text{P}^{\text{PP}} = \text{P}^{\#\text{P}}$ [BBS86] and $\text{BPP} = \text{P}^{\text{BPP}}$). Let $D$ be a database and let $L$ be a language such that $L \in \text{secureBPP}_{\text{path}}^D$. We will show that $L \in \text{obliviousBPP}_{\text{path}}^D$, thereby proving the theorem.

Let $N$ be the machine of Definition 2.6 certifying that $L \in \text{secureBPP}_{\text{path}}^D$. We may assume, without loss of generality (since it is easy to see that $\text{secureBPP}_{\text{path}}^D$ machines can be amplified in the standard way and still remain secure) that the $\epsilon$ of Definition 2.6 satisfies $\epsilon > 1/4$. Also, let $p(n)$ be a polynomial, of the form $n^i + i$ for some integer $i \geq 1$, such that for all sets $L$ the runtime of $N^L$ is at most $p(n)$.

Very informally summarized, in the following a secure computation of $N$ is decomposed (query vector by query vector), to allow an oblivious $\mathrm{BPP}_{\mathrm{path}}$ machine to mimic $N$'s computation. This will be possible because our assumption gives #P-like computational power to our oblivious $\mathrm{BPP}_{\mathrm{path}}$ machine.

We will now define an oblivious machine $Q$ such that $Q^D$ certifies that $L \in \mathrm{obliviousBPP}^D_{\mathrm{path}}$. Let $x$, $|x| = n$, be the input for $N^D$. The computation of $Q^D$ has essentially two stages. In the first stage, as long as the oblivious machine $Q^D$ asks oracle queries, it only has $0^n$ available as input. What it does is: $Q^D$ simulates $N^D$ on input $0^n$. At the end of each path, $Q^D$ has defined a query vector, say, $\boldsymbol{v}$. By the definition of secure computation, the proportion of occurrences of $\boldsymbol{v}$ is the same in $N^D(0^n)$ and $N^D(x)$, that is,

$$\frac{\mathit{path\text{-}occurances}_{N^D(0^n)}(\boldsymbol{v})}{\mathit{total}_{N^D}(0^n)} \quad = \quad \frac{\mathit{path\text{-}occurances}_{N^D(x)}(\boldsymbol{v})}{\mathit{total}_{N^D}(x)}. \tag{1}$$

In the second stage, $Q$ gets access to its input $x$ (and thus cannot ask anymore oracle queries). Let $\alpha_{N^D(x)}(\boldsymbol{v})$ denote the number of accepting paths of $N^D(x)$ that have query vector $\boldsymbol{v}$. Roughly speaking, at each path with query vector $\boldsymbol{v}$ found in the first stage, $Q$ will append a full binary tree having approximately a portion of $\alpha_{N^D(x)}(\boldsymbol{v})/\mathit{path\text{-}occurances}_{N^D(x)}(\boldsymbol{v})$ accepting paths. So, $Q^D$ will have approximately the same overall acceptance behavior as $N^D$.

More formally, we partition the unit interval into $2^q$ intervals of equal length, for some appropriately chosen $q$, and take the largest $k/2^q$, $k \in \{\, 0, \ldots, 2^q - 1 \,\}$, that is still less than $\alpha_{N^D(x)}(\boldsymbol{v})/\mathit{path\text{-}occurances}_{N^D(x)}(\boldsymbol{v})$ as an approximation for it. This is done as follows. For a query vector $\boldsymbol{v}$ let $V = \{\, v \mid v \in D \text{ and } v \text{ is a component of } \boldsymbol{v} \,\}$. Now, $Q$ guesses $k$ of length $q$ and tests whether $(x, \boldsymbol{v}, V, k) \in A$, where $A$ is defined as follows. For $y \in \Sigma^*$, a vector $\boldsymbol{w}$ of at most $p(|y|)$ strings each of length at most $p(|y|)$, a set of strings $W$ each occurring as a component of vector $\boldsymbol{w}$, and a string $j$ of length $q$, interpreted as a binary number between 0 and $2^q - 1$,

$$(y, \boldsymbol{w}, W, j) \in A \iff j \leq 2^q \frac{\alpha_{N^W(y)}(\boldsymbol{w})}{\mathit{path\text{-}occurances}_{N^W(y)}(\boldsymbol{w})} - 1.$$

Clearly, $A \in \mathrm{P}^{\#\mathrm{P}}$, and thus $A$ is in BPP, by assumption. Hence, there exist a probabilistic machine $M_A$ and a polynomial $h$ such that $M_A$ accepts $A$ with error probability bounded by $2^{-q}$, and furthermore, for any input $(y, \boldsymbol{w}, W, j)$, the computation tree of $M_A$ is a full binary tree with $2^{h(|y|)}$ paths.

In order to test whether $(x, \boldsymbol{v}, V, k)$ is in $A$, $Q$ simulates $M_A$ on input $(x, \boldsymbol{v}, V, k)$. $Q$ accepts $x$ if and only if the simulation ends in an accepting state of $M_A$. This completes the definition of $Q$.

We will argue that the machine $Q$ has the desired properties. By the definition of $Q$, it is clearly an oblivious machine. Furthermore, for any given input $x$, let $\boldsymbol{v}$ be a query vector that actually occurs in the run of $N^D(x)$. From equation (1), we get that the portion of paths in the tree of $Q^D$ that have query vector $\boldsymbol{v}$ is identical to the portion in the tree of $N^D(x)$ that have query vector $\boldsymbol{v}$ . We now argue that those paths in $Q^D(x)$ having query vector $\boldsymbol{v}$ have almost the same portion accepting as do those paths in $N^D(x)$. Since $\boldsymbol{v}$ was an arbitrary occurring query vector, it will follow that $Q^D(x)$ has appropriate behavior.

By our construction, we can bound $\alpha_{Q^D(x)}(\boldsymbol{v})$, the number of accepting paths of $Q^D$ that have query vector $\boldsymbol{v}$ as follows. Let $V$ be the associated answer set for $\boldsymbol{v}$. Note that $\alpha_{N^V(x)}(\boldsymbol{v}) = \alpha_{N^D(x)}(\boldsymbol{v})$ and $\mathit{path\text{-}occurances}_{N^V(x)}(\boldsymbol{v}) = \mathit{path\text{-}occurances}_{N^D(x)}(\boldsymbol{v})$. Hence, we have $(x, \boldsymbol{v}, V, k) \in A$ if and only if $0 \leq k \leq \lfloor 2^q \frac{\alpha_{N^D(x)}(\boldsymbol{v})}{\mathit{path\text{-}occurances}_{N^D(x)}(\boldsymbol{v})} \rfloor - 1$. Since $M_A$ has error probability at most $2^{-q}$, we get the following lower bound for $\alpha_{Q^D(x)}(\boldsymbol{v})$:

$$\mathit{path\text{-}occurances}_{Q^D(x)}(\boldsymbol{v}) \frac{\lfloor 2^q \frac{\alpha_{N^D(x)}(\boldsymbol{v})}{\mathit{path\text{-}occurances}_{N^D(x)}(\boldsymbol{v})} \rfloor}{2^q} (1 - 2^{-q}) \quad \leq \quad \alpha_{Q^D(x)}(\boldsymbol{v}).$$

For an upper bound, we have to count the small number of extra accepting paths caused by the error probability of $M_A$:

$$\alpha_{Q^D(x)}(\boldsymbol{v}) \quad \leq \quad \mathit{path\text{-}occurances}_{Q^D(x)}(\boldsymbol{v}) \frac{2^q \frac{\alpha_{N^D(x)}(\boldsymbol{v})}{\mathit{path\text{-}occurances}_{N^D(x)}(\boldsymbol{v})} + 1}{2^q}.$$

With these bounds on $\alpha_{Q^D(x)}(\boldsymbol{v})$, it is now easy to bound the error of $Q^D$ for query vector $\boldsymbol{v}$. Namely, let

$$\mathit{error}(\boldsymbol{v}) \quad = \quad \left| \frac{\alpha_{Q^D(x)}(\boldsymbol{v})}{\mathit{path\text{-}occurances}_{Q^D(x)}(\boldsymbol{v})} - \frac{\alpha_{N^D(x)}(\boldsymbol{v})}{\mathit{path\text{-}occurances}_{N^D(x)}(\boldsymbol{v})} \right|,$$

then we get from the above bounds on $\alpha_{Q^D(x)}(\boldsymbol{v})$ that $\mathit{error}(\boldsymbol{v}) \leq 2^{-q+1}$. Since this holds for each occurring query vector $\boldsymbol{v}$, it certainly holds that $2^{-q+1}$ bounds the overall error portion: the difference between the portion of accepting paths of $N^D(x)$ and the portion of accepting paths of $Q^D(x)$ is at most $2^{-q+1}$. Now, define $q = 4$. Since $N^D$ had an $\epsilon$ (of Definition 2.6) of at least $1/4$, and since we have $\frac{1}{4} - \frac{1}{8} = \frac{1}{8}$, we may conclude that $Q^D$ is an oblivious machine accepting the same language as $N^D$ and having $\epsilon$ (of Definition 2.8) equal to $1/8$. ∎

The proof of Theorem 4.1 can easily be modified to show the corresponding result for probabilistic classes.

**Corollary 4.2** If there is a database $D$ such that secureBPP$^D \neq$ obliviousBPP$^D$, then BPP $\neq$ PP.

Recall that sets in obliviousBPP$^D$ have small circuits. Thus, the existence of a set in secureBPP$^D$ not having a small circuit would separate obliviousBPP$^D$ from secureBPP$^D$.

**Corollary 4.3** If there is a database $D$ such that secureBPP$^D \not\subseteq$ P/poly, then BPP $\neq$ PP.

Since P $\subseteq$ BPP $\subseteq$ PP $\subseteq$ PSPACE, we immediately have the result promised in the section title.

**Corollary 4.4** If there is a database $D$ such that secureBPP$^D_{\text{path}} \neq$ obliviousBPP$^D_{\text{path}}$, then P $\neq$ PSPACE.

# 5 Open Problems

There are several open problems regarding BPP$_{\text{path}}$. Is BPP$_{\text{path}}$ contained in $\Sigma^p_2$ or even in R$^{\text{NP}}$? It seems that the proof technique of Theorem 3.11 doesn't suffice to establish either of these relationships. Does BPP$_{\text{path}}$ have complete sets? There is a relativized world in which BPP lacks complete sets [HH88]; we conjecture that the same holds for BPP$_{\text{path}}$.

Regarding secure computation, does there exist a structural condition that completely characterizes the conditions under which $(\forall D)$ [secureBPP$^D$ = obliviousBPP$^D$] or that completely characterizes the conditions under which $(\forall D)$ [secureBPP$^D_{\text{path}}$ = obliviousBPP$^D_{\text{path}}$]? The study, mentioned in Section 2.2, of classes between BPP$^D$ and secureBPP$^D$, and of classes between BPP$^D_{\text{path}}$ and secureBPP$^D_{\text{path}}$, also remains an interesting open area.

**Acknowledgments**

For helpful discussions, we are grateful to F. Ablayev, G. Brassard, J. Cai, L. Fortnow, F. Green, J. Seiferas, and S. Toda. We thank an anonymous conference referee for pointing out Theorem 6.2, and for helpful pointers to the literature.

# 6 Appendix: Randomized Databases Do Not Strengthen Secure Probabilistic Computation

The secure probabilistic computation of Definition 2.7 can be considered a special case of 2-player interactive computation. In particular, the database can be considered a powerful player that truthfully answers difficult questions asked by a polynomial-time player. When the powerful player in a secure probabilistic computation answers a query, it is unable to take the past history of transactions into consideration. In contrast, players in the usual interactive computation models can remember the history of past transactions. Nonetheless, the secure probabilistic computation model is quite powerful. Even if the database is replaced with a deterministic player that has unlimited computation power and memory, it is clear that the resulting interactive computation can be simulated by a polynomial-time player with a new database that is merely a set.

In this section, we consider the effect of allowing the powerful player to be probabilistic. The resulting model is called a one-oracle instance-hiding scheme that leaks at most the length of its input [BF90]. We present a slightly modified but equivalent definition.

**Definition 6.1 [One-oracle instance-hiding schemes that leaks at most the length of its input]** For a set $L$, a *one-oracle instance-hiding scheme that leaks at most the length of its input* is a synchronous protocol executed by two players, $M_A$ and $M_B$. The number of rounds is bounded by a polynomial in the length of the input. In each round, $M_A$ does a randomized polynomial-time local computation and sends a message (i.e., query) to $M_B$. Upon receiving the query from $M_A$, $M_B$ does an unbounded amount of local computation (possibly using an oracle and a random tape) and sends a message (i.e., answer) to $M_A$. The round is completed when $M_A$ receives the answer sent by $M_B$. Let $\tau$ denote the sequence of messages sent and received by $M_A$ along a computation path, and let $T_A$ denote the random tape of $M_A$. After the last round, $M_A$ uses $\tau$, $T_A$, and the input $x$ to compute a value $M_A(x)$. The interactive computation scheme should satisfy the following two conditions:

1. [Probability of acceptance is bounded away from 1/2] There exists an $\epsilon > 0$ such that for all $x \in \Sigma^*$ it holds that $\Pr[M_A(x) = L(x)] > 1/2 + \epsilon$. (Note that the probability depends on the combined effect of the randomness of both $M_A$ and $M_B$.)

2. [The messages reveal no information to an observer other than perhaps the length of the input] For every $k \in \{0, 1, 2, \cdots\}$, and every vector

$\boldsymbol{v} = (q_1, a_1, q_2, a_2, \cdots, q_k, a_k)$, $q_1, a_1, q_2, a_2, \cdots, q_k, a_k \in \Sigma^*$, and every pair of strings $x \in \Sigma^*$ and $y \in \Sigma^*$ such that $|x| = |y|$, it holds that

$$\Pr[\tau = \boldsymbol{v} \text{ on input } x] = \Pr[\tau = \boldsymbol{v} \text{ on input } y].$$

For any polynomial $p(\cdot)$, the above probability $1/2 + \epsilon$ can be amplified to $1 - 2^{-p(|x|)}$ via the standard technique of repeating computations and using the most frequent result. Clearly, if $L \in \text{secureBPP}^D$ for some database $D$, then $L$ has a one-oracle instance-hiding scheme that leaks at most the length of its input. The following theorem, pointed out to us by an anonymous conference referee, shows that the converse is also true.

**Theorem 6.2** If $L$ is a language that has a one-oracle instance-hiding scheme that leaks at most the length of its input, then there exists a database $D$ such that $L \in \text{secureBPP}^D$.

**Proof:** Let $L$ be a language that has a one-oracle instance-hiding scheme that leaks at most the length of its input. In this proof, we use the notation of Definition 6.1. Following [AFK89], we use the term *transcript* to denote $\tau$, the sequence of queries and answers along a computation path. Without loss of generality, we assume that no transcript is a proper prefix of another transcript and that the length of an input is passed to $M_B$ as the first query. In this proof, we first show that $M_B$ can be modified so that it needs only a polynomial number of random bits. Then we show that these random bits can be supplied by $M_A$, thereby eliminating the need for $M_B$ to be random. It follows that the resulting powerful but deterministic player can be replaced with a set as claimed in the theorem. In the rest of the proof, we call the machines $M_A$ and $M_B$ the *client* and the *server*, respectively.

Given an input of length $n$, the set of transcripts that have non-zero probabilities define a tree whose depth is bounded by a polynomial in $n$. Let's call this a *strategy tree*. (As will become clear later in this proof, the strategy tree effectively defines the strategy of the server. Also, it serves as a convenient template for modifying the strategy of the server.) There are two types of nodes in a strategy tree: server nodes and client nodes. These two types of nodes alternate in each path from the root to a leaf. The root is a client node. The leaves are also client nodes. Each edge from a client node is labeled with a query string; each edge from a server node is labeled with an answer string. Each leaf represents a transcript that has a non-zero probability; the transcript consists of labels read from the edges along the path from the root to the leaf. Edges from the same node have distinct labels so that a transcript defines a unique path in a strategy tree. Corresponding to each

internal node in a strategy tree, there exists a *partial transcript* that consists of the labels that are read from the edges along the path from the root to the node.

Associated with each leaf is the probability with which the transcript corresponding to the leaf occurs. Clearly, based on this probability distribution, we can associate with each internal node the probability with which the partial transcript corresponding to the node occurs. To each edge from a node, we associate the conditional probability with which its label occurs as the next query or answer in a computation, given that the current partial transcript of the computation is the one represented by the node. Note that the sum of the probabilities associated with all the edges from a node is one and that the probability associated with each node is the product of the probabilities associated with the edges along the path from the root to the node.

It is easy to see that an interactive computation reveals at most the length of the input (in the sense of Part 2 in Definition 6.1) if and only if its strategy tree is the same for all inputs of the same length. In particular, the strategy of the server (that is, the probability distribution among edges from each server node) is the same for all inputs of the same length. Further, if we modify the server but (i) we do not add new transcripts to the strategy tree and (ii) the client is not changed, then the resulting strategy tree is the same for all inputs of the same length. Hence, we may arbitrarily adjust the probability distribution among the existing edges from each server node without affecting the instance-hiding nature of the computation. However, such change could affect the acceptance probabilities of input strings. Therefore, in the following, we carefully modify the behavior of the server so that the acceptance of each input string remains intact. In particular, assuming without loss of generality that the probability of correctness (in the sense of Part 1 in Definition 6.1) of the original instance-hiding computation is greater than $\frac{3}{4}$, we will ensure that the probability of correctness of the modified instance-hiding computation is greater than $\frac{5}{8}$.

Let $q(n)$ be a polynomial that bounds both the length of the label of each edge and the depth of the strategy tree. The main obstacle in transforming the randomized server to a deterministic one is the fact that the probability of an edge from a server node can be an arbitrary value. In order to get around the obstacle, we adjust the probability of each edge from server nodes so that it is an integral multiple of $2^{-q^2(n)-q(n)-3}$ and that it differs from the original probability by less than $2^{-q^2(n)-q(n)-3}$. Thus, the probability change at each leaf of the strategy tree is less than $q(n)2^{-q^2(n)-q(n)-3}$. Since there are at most $2^{q^2(n)}$ leaves, it is easy to see that the change in the probability of correctness of the whole computation is less than $\frac{1}{8}$. Therefore, the probability of correctness of the modified secure computation

is greater than $\frac{5}{8}$. Note that the resulting strategy tree can be constructed by the server upon receiving the first query (i.e., the length of the input). The server uses this strategy tree to answer all the queries.

The server modified in this way needs at most a polynomial number $(q(n)(q^2(n) + q(n) + 3))$ of random bits. Hence, the necessary random bits can be supplied to the server by the client at the beginning of a computation. Note that this modification affects neither the instance-hiding nature of the computation nor the probability of correctness of the computation. The resulting server is deterministic, but it may not yet be considered a deterministic function oracle since it may give different answers to different instances of the same queried string. By prefixing each query with an appropriate public information with which the server can uniquely locate the current stage of computation in the strategy tree (for example, $\langle q_1, \ldots, q_{i-1} \rangle$ can be used as a prefix to the $i$-th query along a computation path on which $q_j$ $(0 < j < i)$ is the $j$-th query), the server can be transformed into a deterministic function oracle. It is easy to see that we can further modify the client so that it securely accepts the same language with a set oracle $(D)$ instead of a function oracle. Clearly, the resulting computation is a secure probabilistic computation. ∎

## References


[AFK89] M. Abadi, J. Feigenbaum, and J. Kilian. On hiding information from an oracle. *Journal of Computer and System Sciences*, 39:21–50, 1989.

[Bab85] L. Babai. Trading group theory for randomness. In *Proceedings of the 17th ACM Symposium on Theory of Computing*, pages 421–429, April 1985.

[BBS86] J. Balcázar, R. Book, and U. Schöning. The polynomial-time hierarchy and sparse oracles. *Journal of the ACM*, 33(3):603–617, 1986.

[Bei92] R. Beigel. Perceptrons, PP, and the polynomial hierarchy. In *Proceedings of the 7th Structure in Complexity Theory Conference*, pages 14–19. IEEE Computer Society Press, June 1992.

[BF90] D. Beaver and J. Feigenbaum. Hiding instances in multioracle queries. In *Proceedings of the 7th Annual Symposium on Theoretical Aspects of Computer Science*, pages 37–48. Springer-Verlag *Lecture Notes in Computer Science #415*, 1990.

[BGS75] T. Baker, J. Gill, and R. Solovay. Relativizations of the P=?NP question. *SIAM Journal on Computing*, 4(4):431–442, 1975.

[BHZ87] R. Boppana, J. Håstad, and S. Zachos. Does co-NP have short interactive proofs? *Information Processing Letters*, 25:127–132, 1987.

[Boo89] R. Book. Restricted relativizations of complexity classes. In J. Hartmanis, editor, *Computational Complexity Theory*, pages 47–74. American Mathematical Society, 1989. Proceedings of Symposia in Applied Mathematics #38.

[FFK94] S. Fenner, L. Fortnow, and S. Kurtz. Gap-definable counting classes. *Journal of Computer and System Sciences*, 48(1):116–148, 1994.

[FFKL93] S. Fenner, L. Fortnow, S. Kurtz, and L. Li. An oracle builder's toolkit. In *Proceedings of the 8th Structure in Complexity Theory Conference*, pages 120–131. IEEE Computer Society Press, May 1993.

[FFLS92] J. Feigenbaum, L. Fortnow, C. Lund, and D. Spielman. The power of adaptiveness and additional queries in random-self-reductions. In *Proceedings of the 7th Structure in Complexity Theory Conference*, pages 338–346. IEEE Computer Society Press, June 1992. Final version appears in *Computational Complexity*, v. 4, 1994.

[For94] L. Fortnow, December 1994. Personal Communication.

[Gil77] J. Gill. Computational complexity of probabilistic Turing machines. *SIAM Journal on Computing*, 6(4):675–695, 1977.

[Hem89] L. Hemachandra. The strong exponential hierarchy collapses. *Journal of Computer and System Sciences*, 39(3):299–322, 1989.

[HH88] J. Hartmanis and L. Hemachandra. Complexity classes without machines: On complete languages for UP. *Theoretical Computer Science*, 58:129–142, 1988.

[HLS+93] U. Hertrampf, C. Lautemann, T. Schwentick, H. Vollmer, and K. Wagner. On the power of polynomial time bit-reductions (extended abstract). In *Proceedings of the 8th Structure in Complexity Theory Conference*, pages 200–207. IEEE Computer Society Press, May 1993.

[JMT94] B. Jenner, P. McKenzie, and D. Thérien. Logspace and logtime leaf languages. In *Proceedings of the 9th Structure in Complexity Theory Conference*, pages 242–253. IEEE Computer Society Press, June/July 1994.

[JVV86] M. Jerrum, L. Valiant, and V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43(2,3):169–188, 1986.

[KL80] R. Karp and R. Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the 12th ACM Symposium on Theory of Computing*, pages 302–309, April 1980. An extended version has also appeared as: Turing machines that take advice, *L'Enseignement Mathématique*, 2nd series 28, 1982, pages 191–209.

[Ko82] K. Ko. Some observations on the probabilistic algorithms and NP-hard problems. *Information Processing Letters*, 14(1):39–43, 1982.

[KSTT92] J. Köbler, U. Schöning, S. Toda, and J. Torán. Turing machines with few accepting computations and low sets for PP. *Journal of Computer and System Sciences*, 44(2):272–286, 1992.

[Lau83] C. Lautemann. BPP and the polynomial hierarchy. *Information Processing Letters*, 14:215–217, 1983.

[LLS75] R. Ladner, N. Lynch, and A. Selman. A comparison of polynomial time reducibilities. *Theoretical Computer Science*, 1(2):103–124, 1975.

[LS86] T. Long and A. Selman. Relativizing complexity classes with sparse oracles. *Journal of the ACM*, 33(3):618–627, 1986.

[MS72] A. Meyer and L. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *Proceedings of the 13th IEEE Symposium on Switching and Automata Theory*, pages 125–129, 1972.

[OH93] M. Ogiwara and L. Hemachandra. A complexity theory for closure properties. *Journal of Computer and System Sciences*, 46:295–325, 1993.

[Sim75] J. Simon. *On Some Central Problems in Computational Complexity.* PhD thesis, Cornell University, Ithaca, N.Y., January 1975. Available as Cornell Department of Computer Science Technical Report TR75-224.

[Sip83] M. Sipser. A complexity theoretic approach to randomness. In *Proceedings of the 15th ACM Symposium on Theory of Computing*, pages 330–335, 1983.

[Sto77] L. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1977.

[Sto85] L. Stockmeyer. On approximation algorithms for #P. *SIAM Journal on Computing*, 14(4):849–861, 1985.

[TO92] S. Toda and M. Ogiwara. Counting classes are at least as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 21(2):316–328, 1992.

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

[Ver92] N. Vereshchagin. On the power of PP. In *Proceedings of the 7th Structure in Complexity Theory Conference*, pages 138–143. IEEE Computer Society Press, June 1992.

[Zac82] S. Zachos. Robustness of probabilistic complexity classes under definitional perturbations. *Information and Computation*, 54:143–154, 1982.

[Zac88] S. Zachos. Probabilistic quantifiers and games. *Journal of Computer and System Sciences*, 36:433–451, 1988.

[ZH86] S. Zachos and H. Heller. A decisive characterization of BPP. *Information and Control*, 69:125–135, 1986.