A Quantum "Lifting Theorem" for Constructions of Pseudorandom Generators from Random Oracles

Jonathan Katz* Benjamin Sela*

Abstract

We study the (quantum) security of pseudorandom generators (PRGs) constructed from random oracles. We prove a "lifting theorem" showing, roughly, that if such a PRG is unconditionally secure against classical adversaries making polynomially many queries to the random oracle, then it is also (unconditionally) secure against quantum adversaries in the same sense. As a result of independent interest, we also show that any pseudo-deterministic quantum-oracle algorithm (i.e., a quantum algorithm that with high probability returns the same value on repeated executions) can be simulated by a computationally unbounded—but query bounded—classical-oracle algorithm with only a polynomial blowup in the number of queries. This implies as a corollary that our lifting theorem holds even for PRGs that themselves make quantum queries to the random oracle.

1 Introduction

The random-oracle model (ROM) has long been a useful heuristic for proving the security of cryptosystems that make black-box use of a cryptographic hash function [5]. Boneh et al. [6] observed the need to consider the quantum random-oracle model (QROM) when dealing with quantum adversaries, to take into account the fact that such attackers can evaluate the random oracle on a superposition of inputs. Although many security results in the ROM can be translated to the QROM [6,7,10,14], this is not always the case. Indeed, Zhang et al. [14] show that, in general, an adversary making polynomially many quantum queries to a random oracle cannot be simulated by an adversary making polynomially many classical queries to the random oracle. Yamakawa and Zhandry extended this result to show cryptosystems that are secure in the ROM but not in the QROM, first for the case of digital signatures and public-key encryption schemes [10], and then for one-way functions and collision-resistant hash functions [11].

The above line of work left open the question of showing separations or translation results for other cryptographic primitives constructed from random oracles. For the particular case of pseudorandom generators (PRGs), Yamakawa and Zhandry suggest (without proof) that if the Aaronson-Ambainis conjecture [1] holds[1] then any unconditionally secure construction of a PRG in the ROM remains secure in the QROM. Here, we prove this claim by Yamakawa and Zhandry; in fact, we show the claim is true without assuming the Aaronson-Ambianis conjecture. Specifically,

*Dept. of Computer Science, University of Maryland. **Email:** jkatz2@gmail.com, benjsela@cs.umd.edu. Work supported by NSF award CNS-2154705.

[1]The Aaronson-Ambainis conjecture roughly states that any quantum-oracle algorithm outputting a single bit can be simulated by a classical-oracle algorithm making a similar number of queries.

we prove a "lifting theorem" showing that for any deterministic algorithm G having access to a random oracle, if there is an attacker distinguishing the output of G from random using Q *quantum* queries to the random oracle then there is an attacker distinguishing the output of G from random using $\mathsf{poly}(Q)$ *classical* queries to the random oracle. As a result of additional interest, we analyze the behavior of so-called *pseudo-deterministic* algorithms, i.e., quantum-oracle algorithms that, with high probability over the measurement randomness of their final state, output the same value on repeated executions (for any oracle). We prove that any pseudo-deterministic algorithm making Q quantum queries to an oracle can be simulated by an algorithm making $\mathsf{poly}(Q)$ classical queries to the same oracle. An easy corollary is that our aforementioned result holds even if G makes quantum queries to its random oracle.

1.1 Overview of our Techniques

We give a high-level overview of our results.

PRG lifting theorem. Consider a deterministic (classical) algorithm G based on a random oracle H. We want to show that any algorithm $\mathcal{A}_{\mathsf{quantum}}$ distinguishing the output of G (on a uniform seed) from a random string, and making Q quantum queries to H can be emulated by an algorithm $\mathcal{A}_{\mathsf{classical}}$ making $\mathsf{poly}(Q)$ classical queries to H. Intuitively, our proof relies on the fact that the distinguishing advantage of $\mathcal{A}_{\mathsf{quantum}}$ can only come from correlations between the output of G and the random oracle. Since G only makes polynomially many queries to H, there are only polynomially many oracle inputs that are "usefully" correlated with the output, and those queries can be made by $\mathcal{A}_{\mathsf{classical}}$.

More concretely, say an oracle input is useful if it has some nonnegligible probability of being queried by G, where the probability is over the choice of the seed and the random oracle. There can only be polynomially many such inputs. Our classical distinguisher $\mathcal{A}_{\mathsf{classical}}$ proceeds by first querying the oracle H on those "useful" points, and then extending its partial view of the random oracle to a function H' defined on the entire domain by uniformly generating the remainder of the function values. We prove that the distinguishing advantage of $\mathcal{A}_{\mathsf{quantum}}$ given oracle H' is almost as large as its advantage when given access to the original oracle H. Since $\mathcal{A}_{\mathsf{classical}}$ knows the entire function H', it can simulate $\mathcal{A}_{\mathsf{quantum}}$ without making any more queries to H and the result follows.

Pseudo-deterministic algorithms. The above considers a classical PRG, but we would like to extend the results to the case where G itself makes quantum queries to H. In that case, it seems natural to require that G satisfies the following property: for any H and any seed s, there is a value g such that $G^{|H\rangle}(s)$ outputs g with overwhelming probability over its measurement randomness. A quantum algorithm with that property is called pseudo-deterministic [4]. We show that any pseudo-deterministic oracle algorithm can be simulated by a computationally unbounded classical algorithm making a polynomialy related number of queries to the same oracle. This allows us to replace the quantum PRG with a classical one, at which point we can apply our previous result.

As intuition for our proof, consider a pseudo-deterministic algorithm $\mathcal{A}_{\mathsf{quantum}}$ interacting with an oracle H. We show that even though the answer to a quantum query to H might depend on the value of the oracle at every point in its domain, the output of $\mathcal{A}_{\mathsf{quantum}}$ can only depend on the value of H at a polynomial number of "critical points." (Yamakawa and Zhandry [11] mention this fact in passing without proof.) We not only prove this statement (cf. Lemma 9), but also provide a way to identify those critical points. We then present an algorithm which, given any

pseudo-deterministic oracle algorithm, makes a polynomially related number of classical queries to the same oracle and outputs the same value as the quantum algorithm with high probability.

1.2 Related Work

There has been much work on translating proofs of security in the ROM to the QROM, generally by showing that if a security proof in the ROM satisfies certain conditions on how it interacts with the random oracle, then the security lifts to the setting where an adversary has quantum access to the random oracle for free. For instance, in their work introducing the QROM [6], Boneh et al. defined *history-free* reductions and showed that security proofs satisfying this definition can be lifted from the ROM to the QROM. Song [9] introduced a more detailed framework for translating certain classes of reductions from the ROM to the QROM, and as a consequence proved a lifting theorem for the Full Domain Hash (FDH) signature scheme. Zhang et al. [14] formalized *committed programming reductions* in which the strategy by which the simulator reprograms the random oracle is required to be fixed prior to interacting with the adversary. They not only show that this restricted class of reductions can be used to prove security of a handful of well-known cryptographic schemes, but that such reductions can be lifted to the QROM. Yamakawa and Zhandry [10] give a lifting theorem for something they call a search-type game as well as for digital signature schemes that are restricted in how they make use of the ROM; that model captures the FDH signature scheme as well as signatures constructed using the Fiat-Shamir transform. Kramer and Struck [7] studied the class of so-called "oracle-simple" PKE schemes and show that security of such schemes in the ROM lifts to the QROM.

We note that the above works largely either focus on translating classes of security proofs from the ROM to the QROM, or showing that security can be lifted if the cryptographic scheme itself satisfies certain restrictions on how it interacts with the random oracle. In contrast, we wish to give a lifting theorem for the security for an arbitrary PRG construction where we cannot argue that a certain security proof can be employed in the ROM, and where none of the aforementioned random oracle restrictions can be assumed.

1.3 Open Questions

We conjecture that our result can be extended to constructions of pseudorandom functions (PRFs) in the ROM. Note that there are PRFs that are secure for distinguishers making classical queries to the PRF but not for distinguishers making quantum queries to the PRF [13]. Thus, any lifting theorem in this setting should consider distinguishers making only classical queries to the PRF but making classical/quantum queries to the random oracle. A starting point might be to try to extend our results to the case of *weak* PRFs, where the distinguisher is not given the ability to make any queries to the PRF at all.

2 Preliminaries

For a finite set X, we write $x \leftarrow X$ to denote that x is sampled uniformly from X. For a distribution $\mathcal{X}$, we write $x \leftarrow \mathcal{X}$ to denote that x is sampled according to $\mathcal{X}$. We let $\mathbb{1}$ be the identity function, where the domain and range will be clear from context. For a function H, we let D_H denote the domain of H. Let $\mathsf{Func}_{n,m} = \{H : \{0,1\}^n \mapsto \{0,1\}^m\}$. If $H : D_H \mapsto \{0,1\}^m$, with $D_H \subseteq \{0,1\}^n$, is a (partial) function, and h is a partial function whose domain is a subset of D_H,

we say that H and h are *consistent* if $H(x) = h(x)$ for all $x \in D_h$; we write $\mathsf{Func}_{n,m}(h)$ to denote the subset of $\mathsf{Func}_{n,m}$ that is consistent with h. For a partial function f (not necessarily consistent with H) such that $D_f \subseteq D_H$, we write $H^{(f)}$ to denote the following function:

$$H^{(f)}(x) = \begin{cases} f(x) & x \in D_f \\ H(x) & \text{otherwise.} \end{cases}$$

For a partial function f and a function $H \in \mathsf{Func}_{n,m}(f)$, we let $H \setminus f$ denote the partial function consistent with H and defined exactly where f is not defined. Similarly, for two partial functions f, g which do not disagree on any value for which they are both defined, we let $f \cup g$ denote the (partial) function with domain $D_f \cup D_g$ that is consistent with both functions. We let $\mathsf{negl}(\cdot)$ denote an unspecified negligible function, and let $\mathsf{poly}(\cdot)$ denote an unspecified polynomial.

2.1 Quantum Computation

We associate an n-qubit quantum system with the complex Hilbert space $\mathcal{H} = \mathbb{C}^{2^n}$. A quantum state $|\psi\rangle \in \mathcal{H}$ is a column vector with norm 1. The computational basis of $\mathcal{H}$ is $\{|x\rangle \mid x \in \{0,1\}^n\}$, and any quantum state in $\mathcal{H}$ can be written as $\sum_{x\in\{0,1\}^n} \alpha_x |x\rangle$, where $\alpha_x \in \mathbb{C}$ for all x and $\sum_{x\in\{0,1\}^n} |\alpha_x|^2 = 1$. A measurement of a quantum state $|\psi\rangle = \sum_x \alpha_x |x\rangle$ in the computational basis returns the value $x \in \{0,1\}^n$ with probability $|\alpha_x|^2$; we call the observed x the measurement outcome. If the outcome of measuring $|\psi\rangle$ is x, then the measurement collapses $|\psi\rangle$ to $|x\rangle$. If $|\psi\rangle$ is a column vector, then $\langle\psi|$ is the row vector obtained by taking the conjugate transpose of $|\psi\rangle$.

We write $\||\psi\rangle\|$ to denote the Euclidean norm of the vector $|\psi\rangle$; i.e., if $|\psi\rangle = \sum_x \alpha_x |x\rangle$ then $\||\psi\rangle\| = \sqrt{\sum_x |\alpha_x|^2}$. The Euclidean distance $\||\phi\rangle - |\psi\rangle\|$ gives a measure of how much two quantum states $|\phi\rangle, |\psi\rangle$ differ. Another measure of distance between two quantum states is given by the trace distance between their respective density matrices. For a quantum state $|\psi\rangle$, its corresponding density matrix is $|\psi\rangle\langle\psi|$. For quantum states $|\phi\rangle$ and $|\psi\rangle$, let $\rho = |\phi\rangle\langle\phi|$ and $\sigma = |\psi\rangle\langle\psi|$ be their density matrices. The trace distance between ρ and σ is given by

$$\mathrm{TD}(\rho, \sigma) = \frac{1}{2}\mathrm{Tr}\left[\sqrt{(\rho-\sigma)^\dagger(\rho-\sigma)}\right].$$

The Euclidean distance between two quantum states is an upper bound on the trace distances between their respective density matrices [12].

Lemma 1. *Let $|\phi\rangle$ and $|\psi\rangle$ be quantum states with Euclidean distance ϵ. Then the trace distance between $|\phi\rangle\langle\phi|$ and $|\psi\rangle\langle\psi|$ is at most $\epsilon \cdot \sqrt{1-\epsilon^2/4} \leq \epsilon$.*

Since the trace distance between two density matrices is an upper bound on an algorithm's ability to distinguish the corresponding quantum states, the distinguishing advantage of an algorithm given one of two quantum states is upper bounded by their Euclidean distance.

Quantum algorithms. We consider quantum algorithms having classical input and output. Any such algorithm $\mathcal{A}$ consists of a unitary transformation mapping some initial state $|\psi_{\text{in}}\rangle$ to a final state $|\psi_{\text{out}}\rangle$, followed by a measurement of the final state in the computational basis to produce the output. We write $\mathcal{A} : \{0,1\}^n \mapsto \{0,1\}^m$ to indicate that an n-bit classical input is encoded as the initial quantum state, and the measurement outcome is an m-bit string. We write $y \leftarrow \mathcal{A}(x)$ to denote that y is the result of running $\mathcal{A}$ on input $|x\rangle$ and measuring the resulting quantum state.

Oracle algorithms. Algorithm $\mathcal{A}$ with classical oracle access to a function $H : \{0,1\}^n \mapsto \{0,1\}^m$ (written $\mathcal{A}^H$) can query $x \in \{0,1\}^n$ and receive the result $H(x) \in \{0,1\}^m$ in a single step. Giving $\mathcal{A}$ quantum oracle access to H (written $\mathcal{A}^{|H\rangle}$) means that $\mathcal{A}$ can evaluate the unitary transformation that maps $|x\rangle|y\rangle \mapsto |x\rangle|y \oplus H(x)\rangle$ in a single step. Concretely, this means that $\mathcal{A}$ can query H on a superposition of inputs $|\phi\rangle = \sum_{x\in\{0,1\}^n} \alpha_x|x\rangle$ and receive in return the state $|\psi\rangle = \sum_{x\in\{0,1\}^n} \alpha_x|x\rangle|H(x)\rangle$.[2] For a quantum query $|\phi\rangle = \sum_{x\in\{0,1\}^n} \alpha_x|x\rangle$, we define the *query magnitude at x* to be $q_x(|\phi\rangle) := |\alpha_x|^2$. If $\mathcal{A}^{|H\rangle}$ makes a total of Q queries $|\phi_1\rangle, \ldots, |\phi_Q\rangle$ to its oracle H, then the *total query magnitude at x* is $q_x^H := \sum_{i=1}^{Q} q_x(|\phi_i\rangle)$. The following lemma is one way to formalize the intuition that if the total query magnitude of some algorithm at a point x is small, then the output of the algorithm cannot depend too much of the value of the oracle at x.

Lemma 2 (Swapping lemma [8])**.** *Let $|\phi_f\rangle$ and $|\phi_g\rangle$ denote the final states of a Q-query algorithm $\mathcal{A}$ when given quantum oracle access to f and g, respectively. Then*

$$\||\phi_f\rangle - |\phi_g\rangle\| \leq \sqrt{Q \cdot \sum_{x\,:\,f(x)\neq g(x)} q_x^f}.$$

A related result is that if an oracle is reprogrammed via a random process that is not too likely to reprogram any particular point, then a quantum-oracle algorithm interacting with either the original or reprogrammed oracle cannot detect the reprogramming. The following is a restatement of a result by Alagic et al. [2, Lemma 3].

Lemma 3 (Reprogramming lemma)**.** *Let $\mathcal{D}$ be a distinguisher in the following experiment:*

1. *$\mathcal{D}$ outputs a function $F_0 = F : \{0,1\}^n \mapsto \{0,1\}^m$ and a randomized algorithm $\mathcal{B}$, whose output is a partial function $R : D_R \mapsto \{0,1\}^m$, where $D_R \subseteq \{0,1\}^n$. Let*

$$\epsilon = \max_{x\in\{0,1\}^n} \left\{ \Pr_{R\leftarrow\mathcal{B}}[x \in D_R] \right\}.$$

2. *$\mathcal{B}$ is run to obtain R. Let $F_1 = F_0^{(R)}$. A uniform bit b is chosen, and $\mathcal{D}$ is given quantum access to F_b.*

3. *$\mathcal{D}$ loses access to F_b, and receives the randomness r used to invoke $\mathcal{B}$ in phase 2. Then $\mathcal{D}$ outputs a guess b'.*

For any $\mathcal{D}$ making Q queries, it holds that

$$|\Pr[\mathcal{D} \text{ outputs } 1 \mid b = 1] - \Pr[\mathcal{D} \text{ outputs } 1 \mid b = 0]| \leq 2 \cdot Q \cdot \sqrt{\epsilon}.$$

Pseudo-deterministic algorithms. Quantum algorithms whose outputs are (close to) deterministic will be of particular interest to us. We reformulate the definition to incorporate a parameter δ denoting the probability with which the algorithm returns a "wrong" value.

Definition 1. *Let $\mathcal{A} : \{0,1\}^n \mapsto \{0,1\}^m$ be a quantum algorithm. We say $\mathcal{A}$ is* **δ-deterministic** *if for all $x \in \{0,1\}^n$ there exists $y_x \in \{0,1\}^m$ such that*

$$\Pr[y \leftarrow \mathcal{A}(x) : y \neq y_x] \leq \delta.$$

Quantum-oracle algorithm $\mathcal{A}$ is δ-deterministic if $\mathcal{A}^{|f\rangle}$ is δ-deterministic for every oracle f. (We stress that in this case the value y_x may depend on f.)

[2] Technically there are two registers here, and $\mathcal{A}$ can query $|\phi\rangle = \sum_{x\in\{0,1\}^n, y\in\{0,1\}^m} \alpha_x|x\rangle|y\rangle$ to receive in return $\sum_{x\in\{0,1\}^n, y\in\{0,1\}^m} \alpha_x|x\rangle|y \oplus H(x)\rangle$, but we often omit the second register for notational clarity.

2.2 Pseudorandom Generators

Roughly, a pseudorandom generator G takes as input a uniform seed $s \in \{0,1\}^k$ and outputs a longer string $g \in \{0,1\}^\ell$ that is indistinguishable from a uniform ℓ-bit string. As we are interested in the case where G has oracle access to a random oracle, we specialize our definition to that setting. Formally, the distinguishing advantage of $\mathcal{A}$ relative to G is

$$\mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{A},G} = \left| \Pr_{\substack{s\leftarrow\{0,1\}^k \\ H\leftarrow\mathsf{Func}_{n,m}}}[\mathcal{A}(G(s))=1] - \Pr_{\substack{g\leftarrow\{0,1\}^\ell \\ H\leftarrow\mathsf{Func}_{n,m}}}[\mathcal{A}(g)=1] \right|,$$

where both $\mathcal{A}$ and G are given oracle access to H. If $\mathcal{A}$ is given classical (resp., quantum) access to H, we may emphasize this by referring to the *classical (resp., quantum) distinguishing advantage of $\mathcal{A}$ relative to G*. Note that G might have either classical or quantum access to H.

3 Lifting Theorem for Classical PRGs

Fix an algorithm $G : \{0,1\}^k \mapsto \{0,1\}^\ell$ that always makes exactly Q_G classical queries to an oracle $H : \{0,1\}^n \mapsto \{0,1\}^m$. We show (cf. Theorem 8) that for any algorithm $\mathcal{A}$ making $Q_\mathcal{A}$ *quantum* queries to H there is an algorithm $\mathcal{B}$ making *classical* queries to H such that (1) the distinguishing advantage of $\mathcal{B}$ relative to G is close to the distinguishing advantage of $\mathcal{A}$ relative to G, and (2) the number of queries $\mathcal{B}$ makes to its oracle is polynomial in k, n, Q_G, $Q_\mathcal{A}$, and the inverse of the distinguishing advantage of $\mathcal{A}$.

We begin by introducing some notation and definitions. For notational convenience, we overload the definition of G so that in addition to its actual output $g \in \{0,1\}^\ell$ it also outputs the list τ of queries it made to H and the associated answers. We let $\mathcal{G}$ denote the distribution on (H,s,g,τ) generated by sampling a uniform function $H \in \mathsf{Func}_{n,m}$, choosing a uniform seed $s \in \{0,1\}^k$, and then running $G^H(s)$ to generate g and τ. We also define the following conditional distributions:

- For $g \in \{0,1\}^\ell$ in the range of the PRG, let $\mathcal{T}_g$ be the distribution on τ after the above process conditioned on g being the output of the PRG. Similarly, for a partial function h such that there exists (H,s) with $H \in \mathsf{Func}_{n,m}(h)$ satisfying $g = G^H(s)$, let $\mathcal{T}_{g,h}$ be the distribution on τ conditioned on (i) g being the output of the PRG and (ii) $H \in \mathsf{Func}_{n,m}(h)$.
- We write $\mathcal{O}$ for the uniform distribution over $\mathsf{Func}_{n,m}$. For g in the range of the PRG, we let $\mathcal{O}_g$ denote the distribution over $\mathsf{Func}_{n,m}$ conditioned on g being the output of the PRG. Similarly, for a partial function h such that there exists (H,s) with $H \in \mathsf{Func}_{n,m}(h)$ satisfying $g = G^H(s)$, let $\mathcal{O}_{g,h}$ be the distribution over $\mathsf{Func}_{n,m}$ conditioned on (i) g being the output of the PRG and (ii) $H \in \mathsf{Func}_{n,m}(h)$.

Writing everything out explicitly, we have

$$\mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{A},G} = \left| \Pr[\mathsf{PRG}_{\mathcal{A},G}=1] - \Pr[\mathsf{Rand}_{\mathcal{A},G}=1] \right|$$

where $\mathsf{PRG}_{\mathcal{A},G}$ and $\mathsf{Rand}_{\mathcal{A},G}$ are defined as follows:

$\mathsf{PRG}_{\mathcal{A},G}$
1: $H \leftarrow \mathcal{O}$
2: $s \leftarrow \{0,1\}^k$
3: $(g,\tau) := G^H(s)$
4: $b \leftarrow \mathcal{A}^{\lvert H\rangle}(g)$
5: return b

$\mathsf{Rand}_{\mathcal{A},G}$
1: $H \leftarrow \mathcal{O}$
2: $g \leftarrow \{0,1\}^\ell$
3: $b \leftarrow \mathcal{A}^{\lvert H\rangle}(g)$
4: return b

We also define versions of these experiments where we condition on a particular value g:

$\mathsf{PRG}_{\mathcal{A},G}(g)$
1: $H \leftarrow \mathcal{O}_g$
2: $b \leftarrow \mathcal{A}^{\lvert H\rangle}(g)$
3: return b

$\mathsf{Rand}_{\mathcal{A},G}(g)$
1: $H \leftarrow \mathcal{O}$
2: $b \leftarrow \mathcal{A}^{\lvert H\rangle}(g)$
3: return b

It is immediate that

$$\Pr[\mathsf{PRG}_{\mathcal{A},G} = 1] = \Pr[(H,s,g,\tau) \leftarrow \mathcal{G} : \mathsf{PRG}_{\mathcal{A},G}(g) = 1].$$

In Figure 1 we describe an algorithm $\mathcal{B}$ that makes only *classical* queries to H. On input g, algorithm $\mathcal{B}$ first runs a subroutine $\mathsf{findTranscript}^H$ that, intuitively, results in $\mathcal{B}$ querying H on all points "likely" to appear in the transcript of G's interaction with H (conditioned on g being the output of G). The output h of $\mathsf{findTranscript}^H$ contains all query/answer pairs that $\mathcal{B}$ thus learns. $\mathcal{B}$ then chooses a random function H' consistent with the partial function h, and runs $\mathcal{A}$ with input g and access to H'. (Note that this can be done with no further queries to H.) Finally, $\mathcal{B}$ outputs whatever $\mathcal{A}$ does.

Our aim is to show that the distribution of the output of $\mathcal{B}^H(g)$ is close to the distribution of the output of $\mathcal{A}^H(g)$ regardless of whether g is a uniform string or whether g was the output of $G^H(s)$ on a uniform seed s. This is easy to show when g is uniform, as in that case g is uncorrelated with the random oracle H and so the distinguishing advantage of $\mathcal{A}$ is unchanged regardless of whether it interacts with oracle H or an independent random oracle H'.

The more challenging case is when g is the output of the PRG. Intuitively we can still resample the values of the oracle which were not queried by the PRG to produce g, as the computation of the PRG is independent of these values. This leaves us with the problem of simulating the values of the oracle which are now correlated with g. For each point on which the oracle H was queried by G, we can categorize the point based on if it had a high (non negligible) probability of being queried by G, where the probability is over the randomness of the PRG input s as well as the randomness over the previous oracle values which were queried. To handle points which are likely to be queried, our distinguisher simply queries all points which have a sufficiently large probability of showing up in a transcript between the oracle and the PRG. This process is formalized in Algorithm 6, and we prove its efficiency in Lemma 5. To handle the remaining points in the transcript which are not likely to have been queried, we rely on an oracle reprogramming result due to Alagic et al. [2] (Lemma 3), which states that if an oracle is reprogrammed by some randomized process then a quantum distinguisher interacting with either the original oracle or the reprogrammed oracle will not notice the reprogramming provided that no particular point is very likely to be reprogrammed. Viewing the randomized reprogramming process as sampling a transcript between the oracle and

PRG, and then only outputting the points which were not likely to have been queried, we can argue that the distinguisher will not notice if we resample these points randomly.

Algorithm 5 $\mathcal{B}^H(g)$

if $\nexists s, H$ such that $G^H(s) = g$ **then**
    return 0
end if
$\delta = \left(\frac{\mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{A},G}}{6 \cdot Q_{\mathcal{A}}}\right)^2$
$h = \mathsf{findTranscript}^H(g, \delta)$
if $h = \perp$ **then** return 0 **end if**
$H' \leftarrow \mathsf{Func}_{n,m}(h)$
$b \leftarrow \mathcal{A}^{|H'\rangle}(g)$
return b

Algorithm 6 $\mathsf{findTranscript}^H(g, \delta)$

$\mathsf{limit} = \left\lceil \frac{-\ln(\delta) \cdot 4Q_G^2}{\delta^2} \right\rceil + 1$
$i = 0$, $h = \emptyset$, $\epsilon = 1$
while $\epsilon > \delta$ and $i < \mathsf{limit}$ **do**
    $i = i + 1$
    if $\mathcal{T}_{g,h}$ is undefined **then** return $\perp$ **end if**
    $x' = \arg\max_x \{\Pr_{\tau \leftarrow \mathcal{T}_{g,h}}[(x, \star) \in \tau]\}$
    Query H on x' and add $(x', H(x'))$ to h.
    $\epsilon = \max_x \{\Pr_{\tau \leftarrow \mathcal{T}_{g,h}}[(x, \star) \in \tau]\}$
end while
if $i \geq \mathsf{limit}$ **then** return $\perp$ **end if**
return h

Figure 1

We now give a more technical sketch of the proof. As just discussed, the bulk of the technical work is in showing that $\mathcal{B}^{(\cdot)}(g)$ (Algorithm 5) correctly simulates the behavior of the quantum algorithm in the experiment $\mathsf{PRG}_{\mathcal{A},G}$ (Lemma 7). Since the input g given to the distinguisher is fixed in $\mathsf{PRG}_{\mathcal{A},G}(g)$, the distribution over outputs from the distinguisher only depends on the distribution from which the oracle is drawn. With this in mind, our goal will be to find a distribution $\mathcal{O}'_g$ over oracles such that (1) the output of $\mathcal{A}$ is distributed almost identically when given an oracle from $\mathcal{O}'_g$ or from $\mathcal{O}_g$, and (2) there is some way of sampling an oracle from $\mathcal{O}'_g$ without making a large number of queries to the original oracle.

We now introduce three hybrid experiments, each of which with a slightly different method of generating the oracle that $\mathcal{A}$ interacts with.

- Hyb_1: In this experiment we sample an oracle $H \leftarrow \mathcal{O}_g$, run $\mathsf{findTranscript}^H(g, \delta)$ on that oracle to obtain a partial function h, and then resample the remaining values of the oracle that have not been queried by $\mathsf{findTranscript}$ from the distribution $\mathcal{O}_{g,h}$.

- Hyb_2: This experiment replaces the sampling procedure $H \leftarrow \mathcal{O}_{g,h}$ with a procedure in which we sample a transcript $\tau \leftarrow \mathcal{T}_{g,h}$, extend the domain of the partial function h by uniformly generating the remaining values to produce $H_0 \leftarrow \mathsf{Func}_{n,m}(h)$. Finally we reprogram H_0 according to the partial function $\tau \setminus h$ to produce $H_1 = H_0^{(\tau \setminus h)}$. In Lemma 6 we prove that this procedure is equivalent to the one in Hyb_1.

- Hyb_3: In this experiment, we omit the final reprogramming step of Hyb_2 and give $\mathcal{A}$ access to H_0 as defined above. To compare Hyb_2 and Hyb_3, note that the task of distinguishing H_0 from H_1 matches the reprogramming game of Lemma 3, where the randomized reprogramming process from the lemma outputs the partial function $\tau \setminus h$. Lemma 3 gives a bound on distinguishing H_0 and H_1 based on the largest probability of any particular point showing up in the domain of $\tau \setminus h$. In Lemma 5 we obtain such a bound by showing that with high probability over the partial function h returned by $\mathsf{findTranscript}$, no point has a high probability of showing up in the domain of $\tau \setminus h$.

We now turn to the proof. We rely on the following result of Austrin et al. [3, Lemma 3.6].

Lemma 4. *Let $L, z_1, x_1, ..., z_q, x_q$ be a finite sequence of correlated random variables, where L ranges over subsets of a universe $\mathcal{U}$ and $x_i \in \mathcal{U}$ for all i, and with the property that if $i \neq j$ then $x_i \neq x_j$. Conditioned on a sequence $z_1, x_1, ..., z_q$, say that x_i is δ-heavy if $\Pr[x_i \in L \mid z_1, x_1, ..., z_i] \geq \delta$ and, for the same sequence, define $\mathcal{S} = \{x_i \mid x_i \text{ is } \delta\text{-heavy}\}$. Then, $\mathbb{E}[|S|] \leq \mathbb{E}[|L|]/\delta$.*

We start by proving an important property about the $\mathsf{findTranscript}$ subroutine.

Lemma 5. *Let $\delta \in (0,1)$ and $g \in \{0,1\}^\ell$. Then with probability at least $1-\delta$ (over the randomness of the oracle H drawn from $\mathcal{O}_g$), $\mathsf{findTranscript}^H(g,\delta)$ returns a partial function h with domain of size at most $\frac{Q_G}{\delta^2}$ such that*

$$\max_{x \notin D_h} \left\{ \Pr_{\tau \leftarrow \mathcal{T}_{g,h}} [(x, \star) \in \tau] \right\} \leq \delta.$$

Proof. The final inequality in the lemma holds by the termination condition for $\mathsf{findTranscript}$, so it only remains to prove that the size of the partial function h is at most Q_G^2/δ. Let L be the random variable describing a transcript $\tau \leftarrow \mathcal{T}_g$ of the queries G makes to the random oracle, and let $z_1 = g$ be the output of the PRG. For $i \geq 1$, let the random variable x_i be the ith query made by $\mathsf{findTranscript}$ to the random oracle, and let z_{i+1} be the value returned by the oracle. Letting q be the number of queries made by $\mathsf{findTranscript}$, set $x_i = \perp$ and $z_{i+1} = 0$ for $q < i \leq 2^n$. Every query made by $\mathsf{findTranscript}$ is δ-heavy, and so Lemma 4 implies $\mathbb{E}[|\mathcal{S}|] \leq Q_G/\delta$. It follows from Markov's inequality that $\Pr\left[|\mathcal{S}| > Q_g/\delta^2\right] \leq \delta$. Since the partial function $h = \{(x_i, z_{i+1})\}_{i=1}^q$ returned by $\mathsf{findTranscript}$ has $\mathcal{S}$ as its domain, the lemma statement follows. □

Now we turn to the proof of Lemma 6 which justifies the transition from $\mathsf{Hyb}_1(g)$ to $\mathsf{Hyb}_2(g)$.

Lemma 6. *Let $g_0 \in \{0,1\}^\ell$, and let h_0 be a partial function such that there exists (s, H) where $H \in \mathsf{Func}_{n,m}(h_0)$ and $(g_0, \tau) = G^H(s)$ for some τ. Define the following random variables:*

$$\mathsf{O}'_{g_0,h_0} := \left\{ \begin{array}{c} \tau \leftarrow \mathcal{T}_{g_0,h_0} \\ H \leftarrow \mathsf{Func}_{n,m}(h_0) \end{array} : H^{(\tau \setminus h_0)} \right\} \quad \text{and} \quad \mathsf{O}_{g_0,h_0} := \left\{ H \leftarrow \mathcal{O}_{g_0,h_0} : H \right\}.$$

Then O'_{g_0,h_0} and O_{g_0,h_0} are distributed identically.

Proof. Let $\mathcal{U}$ be the uniform distribution over oracle-seed combinations in $\mathsf{Func}_{n,m} \times \{0,1\}^k$, and let $R_{g,\tau,h}$ be a relation over oracle-seed pairs such that

$$(H, s) \in R_{g,\tau,h} \iff H \in \mathsf{Func}(h) \wedge (g, \tau) = G^H(s).$$

First, we can write the probability of O'_{g_0,h_0} generating a particular function H as

$$\Pr[\mathsf{O}'_{g_0,h_0} = H] = \sum_\tau \Pr[\tau \leftarrow \mathcal{T}_{g_0,h_0}] \cdot \Pr_{\mathsf{O} \leftarrow \mathsf{Func}_{n,m}(h_0 \cup \tau)} [\mathsf{O} = H]. \tag{1}$$

On the other hand, we can write the probability of the random variable O_{g_0,h_0} generating some function H as

$$\begin{aligned} \Pr[\mathsf{O}_{g_0,h_0} = H] &= \Pr_{\mathsf{O},\mathsf{S} \leftarrow \mathcal{U}} [\mathsf{O} \in \mathsf{Func}(H \setminus h_0) \mid \mathsf{O} \in \mathsf{Func}(h_0), (g_0, \star) = G^{\mathsf{O}}(\mathsf{S})] \\ &= \sum_\tau \Pr[\tau \leftarrow \mathcal{T}_{g_0,h_0}] \cdot \Pr_{\mathsf{O},\mathsf{S} \leftarrow \mathcal{U}} [\mathsf{O} \in \mathsf{Func}(H \setminus h_0) \mid \mathsf{O} \in \mathsf{Func}(h), (g_0, \tau) = G^{\mathsf{O}}(\mathsf{S})] \\ &= \sum_\tau \Pr[\tau \leftarrow \mathcal{T}_{g_0,h_0}] \cdot \Pr_{\mathsf{O},\mathsf{S} \leftarrow \mathcal{U}} [\mathsf{O} \in \mathsf{Func}(H \setminus h_0) \mid (\mathsf{O}, \mathsf{S}) \in R_{g_0,\tau,h_0}] \end{aligned} \tag{2}$$

Comparing Equation (1) above with Equation (2), we will prove our result by showing that for any τ that is consistent with h_0,

$$\Pr_{\mathsf{O},\mathsf{S}\leftarrow\mathcal{U}}[\mathsf{O} \in \mathsf{Func}\,(H \setminus h_0) \mid (\mathsf{O},\mathsf{S}) \in R_{g_0,\tau,h_0}] = \Pr_{\mathsf{O}\leftarrow\mathsf{Func}_{n,m}(h_0\cup\tau)}[\mathsf{O} = H].$$

To prove the above, we will show that for any two oracles $H_0, H_1 \in \mathsf{Func}(h_0 \cup \tau)$,

$$\Pr_{\mathsf{O},\mathsf{S}\leftarrow\mathcal{U}}[\mathsf{O} = H_0 \mid (\mathsf{O},\mathsf{S}) \in R_{g_0,\tau,h_0}] = \Pr_{\mathsf{O},\mathsf{S}\leftarrow\mathcal{U}}[\mathsf{O} = H_1 \mid (\mathsf{O},\mathsf{S}) \in R_{g_0,\tau,h_0}].$$

Let τ be any partial function consistent with h_0, and consider two oracle seed pairs (H_0, s_0) and (H_1, s_1) such that both pairs are in the relation R_{g_0,τ,h_0}. Then we have

$$\Pr_{\mathsf{O},\mathsf{S}\leftarrow\mathcal{U}}[(\mathsf{O},\mathsf{S}) = (H_0, s_0) \mid (\mathsf{O},\mathsf{S}) \in R_{g_0,\tau,h_0}] = \Pr_{\mathsf{O},\mathsf{S}\leftarrow\mathcal{U}}[(\mathsf{O},\mathsf{S}) = (H_1, s_1) \mid (\mathsf{O},\mathsf{S}) \in R_{g_0,\tau,h_0}].$$

Define the following set:

$$\mathsf{Good}_{g_0,\tau}(H) = \{s \mid G^H(s) = (g_0, \tau)\}.$$

Since $H_0, H_1 \in \mathsf{Func}(\tau)$, we have that $\mathsf{Good}_{g_0,\tau}(H_0) = \mathsf{Good}_{g_0,\tau}(H_1)$. To see this, note that since the execution of G on input $s \in \mathsf{Good}_{g_0,\tau}(H_0)$ with oracle H_0 is independent of oracle values outside of τ, it follows that the execution on G with oracle H_1 will be identical. It follows that

$$\begin{aligned}
\Pr_{\mathsf{O},\mathsf{S}\leftarrow\mathcal{U}}[\mathsf{O} = H_0 \mid (\mathsf{O},\mathsf{S}) \in R_{g_0,\tau,h_0}] &= \sum_{s\in\mathsf{Good}_{g_0,\tau}(H_0)} \Pr_{\mathsf{O},\mathsf{S}\leftarrow\mathcal{U}}[(\mathsf{O},\mathsf{S}) = (H_0, s) \mid (\mathsf{O},\mathsf{S}) \in R_{g_0,\tau,h_0}] \\
&= \sum_{s\in\mathsf{Good}_{g_0,\tau}(H_1)} \Pr_{\mathsf{O},\mathsf{S}\leftarrow\mathcal{U}}[(\mathsf{O},\mathsf{S}) = (H_1, s) \mid (\mathsf{O},\mathsf{S}) \in R_{g_0,\tau,h_0}] \\
&= \Pr_{\mathsf{O},\mathsf{S}\leftarrow\mathcal{U}}[\mathsf{O} = H_1 \mid (\mathsf{O},\mathsf{S}) \in R_{g_0,\tau,h_0}].
\end{aligned}$$

This concludes the proof. □

With the above two results, we are now ready to proceed to the main proof. We start by showing that our classical distinguisher $\mathcal{B}^H$ matches the performance of the quantum distinguisher $\mathcal{A}^{|H\rangle}$ when interacting with a PRG.

Lemma 7. *Let $\mathcal{A}$ be a quantum algorithm making $Q_\mathcal{A}$ quantum oracle queries, and let $\mathcal{B}^H$ be as described in Algorithm 5. Then,*

$$\left|\Pr[\mathsf{PRG}_{\mathcal{A},G} = 1] - \Pr[\mathsf{PRG}_{\mathcal{B},G} = 1]\right| \leq \frac{1}{2}\mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{A},G}.$$

Proof. Let $\epsilon = \max_g \left\{\left|\Pr[\mathsf{PRG}_{\mathcal{A},G}(g) = 1] - \Pr[\mathsf{PRG}_{\mathcal{B},G}(g) = 1]\right|\right\}$. Then we have

$$\begin{aligned}
&\left|\Pr[\mathsf{PRG}_{\mathcal{A},G} = 1] - \Pr[\mathsf{PRG}_{\mathcal{B},G} = 1]\right| \\
&\qquad = \left|\sum_g \Pr_{\substack{s\leftarrow\{0,1\}^k \\ H\leftarrow\mathsf{Func}_{n,m}}}[g = G^H(s)]\left(\Pr[\mathsf{PRG}_{\mathcal{A},G}(g) = 1] - \Pr[\mathsf{PRG}_{\mathcal{B},G}(g) = 1]\right)\right| \\
&\qquad \leq \epsilon \cdot \sum_g \Pr_{\substack{s\leftarrow\{0,1\}^k \\ H\leftarrow\mathsf{Func}_{n,m}}}[g = G^H(s)] \;=\; \epsilon.
\end{aligned}$$

We now derive a bound for ϵ via the sequence of games Hyb_1, Hyb_2, Hyb_3, discussed earlier. Let g be in the range of the PRG (for at least one seed-oracle combination s, H). The pseudocode for the games is presented here. Note that the games only differ in how the oracle is generated. Thus, our task will be to show that in each transition between games, the quantum distinguisher $\mathcal{A}$ will not notice the difference between oracles.

$\mathsf{Hyb}_1(g)$

1: $H \leftarrow \mathcal{O}_g$
2: $h = \mathsf{findTranscript}^H(g, \delta)$
3: $H_1 \leftarrow \mathcal{O}_{g,h}$
4: $b \leftarrow \mathcal{A}^{|H_1\rangle}(g)$
5: return b

$\mathsf{Hyb}_2(g)$ $\boxed{\mathsf{Hyb}_3(g)}$

1: $H \leftarrow \mathcal{O}_g$
2: $h = \mathsf{findTranscript}^H(g, \delta)$
3: $H_0 \leftarrow \mathsf{Func}_{n,m}(h)$
4: $\tau \leftarrow \mathcal{T}_{g,h}$
5: $H_1 = H_0^{(\tau \setminus h)}$
6: $b \leftarrow \mathcal{A}^{|H_1\rangle}(g)$ $\boxed{b \leftarrow \mathcal{A}^{|H_0\rangle}(g)}$
7: return b

($\mathsf{PRG}_{\mathcal{A},G}(g) \mapsto \mathsf{Hyb}_1(g)$): We show that H in $\mathsf{PRG}_{\mathcal{A},G}(g)$ is distributed identically to H_1 in Hyb_1. The only property of $\mathsf{findTranscript}$ that we require is that the algorithm outputs every query that it makes. Fix some oracle H, and let $h = \mathsf{findTranscript}^H(g, \delta)$. We will show that H has the same probability of being returned in each of the games. First, we have

$$\begin{aligned}
\Pr_{\mathsf{O} \leftarrow \mathcal{O}_g}[\mathsf{O} = H] &= \Pr_{\mathsf{O} \leftarrow \mathcal{O}_g}[\mathsf{O} \in \mathsf{Func}_{n,m}(h) \wedge \mathsf{O} \in \mathsf{Func}_{n,m}(H \setminus h)] \\
&= \Pr_{\mathsf{O} \leftarrow \mathcal{O}_g}[\mathsf{O} \in \mathsf{Func}_{n,m}(h)] \cdot \Pr_{\mathsf{O} \leftarrow \mathcal{O}_g}[\mathsf{O} \in \mathsf{Func}_{n,m}(H \setminus h) \mid \mathsf{O} \in \mathsf{Func}_{n,m}(h)] \\
&= \Pr_{\mathsf{O} \leftarrow \mathcal{O}_g}[\mathsf{O} \in \mathsf{Func}_{n,m}(h)] \cdot \Pr_{\mathsf{O} \leftarrow \mathcal{O}_{g,h}}[\mathsf{O} \in \mathsf{Func}_{n,m}(H \setminus h)]
\end{aligned}$$

The left hand side of the above equation is the probability of $\mathsf{PRG}_{\mathcal{A},G}(g)$ generating H on line 1. We will now argue that the right hand side gives the probability of $\mathsf{Hyb}_1(g)$ generating H on line 3. Since the partial function h output by $\mathsf{findTranscript}$ contains all oracle values that were queried, h is independent of any oracle values outside of the domain of h. Therefore, for any $H' \neq H$,

$$H' \in \mathsf{Func}_{n,m}(h) \iff h = \mathsf{findTranscript}^{H'}(g, \delta).$$

It follows that $\Pr_{\mathsf{O} \leftarrow \mathcal{O}_g}[O \in \mathsf{Func}_{n,m}(h)] = \Pr_{\mathsf{O} \leftarrow \mathcal{O}_g}[h = \mathsf{findTranscript}^{\mathsf{O}}(g, \delta)]$. Therefore the right hand side of the previous calculation gives the probability of H being generated on line 3 of $\mathsf{Hyb}_1(g)$.

($\mathsf{Hyb}_1(g) \mapsto \mathsf{Hyb}_2(g)$): It follows from Lemma 6 that H_1 on line 5 of $\mathsf{Hyb}_2(g)$ is distributed identically to H_1 on line 3 of $\mathsf{Hyb}_1(g)$. It follows that $\Pr[\mathsf{Hyb}_1(g) = 1] = \Pr[\mathsf{Hyb}_2(g) = 1]$.

($\mathsf{Hyb}_2(g) \mapsto \mathsf{Hyb}_3(g)$): We will map the task of distinguishing Hyb_2 from Hyb_3 onto the distinguishing game of Lemma 3. Referring to the notation in Lemma 3, let F_0 represent H_0 on line 3 of Hyb_2 (and Hyb_3). Define the randomized reprogramming algorithm $\mathcal{B}$ so that it samples $\tau \leftarrow \mathcal{T}_{g,h}$ and produces $F_1 = F_0^{(\tau \setminus h)}$ (lines 4–5 of Hyb_2 and Hyb_3). Since distinguishing F_0 and F_1 (or equivalently H_0 and H_1) is equivalent to distinguishing the two games, it follows that

$$\left| \Pr[\mathsf{Hyb}_2(g) = 1] - \Pr[\mathsf{Hyb}_3(g) = 1] \right| \leq 2 \cdot Q_A \cdot \sqrt{\max_{x \notin D_h} \left\{ \Pr_{\tau \leftarrow \mathcal{T}_{g,h}}[(x, \star) \in \tau] \right\}},$$

where we recall that D_h denotes the domain of the partial function h. Lemma 5 tells us that with probability at least $1-\delta$ over the function h, the distribution $\mathcal{T}_{g,h}$ on line 4 of $\mathsf{Hyb}_2(g)$ has the property that

$$\max_{x \notin D_h} \left\{ \Pr_{\tau \leftarrow \mathcal{T}_{g,h}} [(x,\star) \in \tau] \right\} \leq \delta.$$

It follows that $\mathsf{Hyb}_2(g)$ and $\mathsf{Hyb}_3(g)$ can be distinguished with advantage at most $(1-\delta)\cdot 2Q_A\sqrt{\delta}+\delta$, where $Q_{\mathcal{A}}$ is the number of oracle queries made by $\mathcal{A}$. Therefore we have

$$\left| \Pr[\mathsf{Hyb}_2(g)=1] - \Pr[\mathsf{Hyb}_3(g)=1] \right| \leq 2 \cdot Q_{\mathcal{A}} \cdot \sqrt{\delta} + \delta.$$

($\mathsf{Hyb}_3(g) \mapsto \mathsf{PRG}_{\mathcal{B},G}(g)$): These two games are identical as $\mathcal{B}$ can generate H_0 on line 3 itself and is therefore able to simulate the entire computation of $A^{|H_0\rangle}(g)$ including the superposition queries. Therefore, $\Pr[\mathsf{Hyb}_3(g)=1] = \Pr[\mathsf{PRG}_{\mathcal{B},G}(g)=1]$. Putting everything together, we have

$$\left| \Pr[\mathsf{PRG}_{\mathcal{A},G}(g)=1] - \Pr[\mathsf{PRG}_{\mathcal{B},G}(g)=1] \right| \leq 2Q_{\mathcal{A}}\sqrt{\delta} + \delta.$$

Letting $\delta = \left(\frac{\mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{A},G}}{6\cdot Q_{\mathcal{A}}}\right)^2$ as in Algorithm 5, we obtain the bound in the lemma statement. □

We can now prove the main result of this section.

Theorem 8. *Let $G : \{0,1\}^k \mapsto \{0,1\}^\ell$ be a deterministic algorithm making Q_G classical queries to an oracle $H : \{0,1\}^n \mapsto \{0,1\}^m$, and let $\mathcal{A}$ be an algorithm making $Q_{\mathcal{A}}$ quantum queries to H. Let $\epsilon = \mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{A},G}$. Then there is an algorithm $\mathcal{B}$ making $\mathsf{poly}(k,n,Q_G,Q_{\mathcal{A}},\epsilon^{-1})$ classical queries to H with $\mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{B},G} \geq \epsilon/2$.*

Proof. Let $\mathcal{A}$ be a quantum distinguisher and let $\mathcal{B}$ be Algorithm 2. Let

$$\delta_{\mathsf{PRG}} = \Pr[\mathsf{PRG}_{\mathcal{A},G}=1] - \Pr[\mathsf{PRG}_{\mathcal{B},G}=1], \qquad \text{and} \qquad \delta_{\mathsf{rand}} = \Pr[\mathsf{Rand}_{\mathcal{B},G}=1] - \Pr[\mathsf{Rand}_{\mathcal{A},G}=1].$$

Then

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{B},G} &= |\Pr[\mathsf{PRG}_{\mathcal{B},G}=1] - \Pr[\mathsf{Rand}_{\mathcal{B},G}=1]| \\
&= |\Pr[\mathsf{PRG}_{\mathcal{A},G}=1] - \delta_{\mathsf{PRG}} - \Pr[\mathsf{Rand}_{\mathcal{A},G}=1] - \delta_{\mathsf{rand}}| \\
&= |\Pr[\mathsf{PRG}_{\mathcal{A},G}=1] - \Pr[\mathsf{Rand}_{\mathcal{A},G}=1] - (\delta_{\mathsf{PRG}} + \delta_{\mathsf{rand}})| \\
&\geq |\Pr[\mathsf{PRG}_{\mathcal{A},G}=1] - \Pr[\mathsf{Rand}_{\mathcal{A},G}=1]| - |\delta_{\mathsf{PRG}} + \delta_{\mathsf{rand}}| \\
&= \mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{A},G} - |\delta_{\mathsf{PRG}} + \delta_{\mathsf{rand}}|\,.
\end{aligned}$$

Lemma 6 implies $\delta_{\mathsf{PRG}} \leq \frac{1}{2}\mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{A},G}$. On the other hand, in $\mathsf{Rand}_{\mathcal{B},G}$ and $\mathsf{Rand}_{\mathcal{A},G}$ the oracle is independent of g and therefore $\Pr[\mathsf{Rand}_{\mathcal{B},G}(g)=1] = \Pr[\mathsf{Rand}_{\mathcal{A},G}(g)=0]$. The theorem follows. □

4 Simulating Pseudo-Deterministic Algorithms Classically

We now turn to the case where the PRG is a quantum algorithm which may make quantum queries to the oracle. In this section we show (1) that any pseudo-deterministic oracle algorithm can only depend on a polynomial number of oracle values, and (2) that for any pseudo-deterministic quantum

oracle algorithm $\mathcal{A}$ making queries to an oracle H, there exists a computationally unbounded but query bounded classical algorithm $\mathcal{B}$ that makes only polynomially more (classical) queries to the same oracle H, and outputs the same value as $\mathcal{A}$ with high probability. We stress that in this section our results concern fixed (rather than random) oracles. In section 4.3 we show as a corollary that our lifting theorem for PRGs applies to PRGs making quantum queries to the random oracle.

4.1 From δ-Deterministic to Classical Algorithms

First we prove that any δ-deterministic oracle algorithm can only depend on a polynomial number of oracle values.

Lemma 9. *Let $\mathcal{A}^{(\cdot)}$ be a quantum algorithm making Q quantum queries to an oracle $F : \{0,1\}^n \mapsto \{0,1\}^*$. Assume that $\mathcal{A}^{(\cdot)}$ is δ-deterministic given oracle access to any function on n-bit strings. Suppose that $\mathcal{A}$ makes Q queries to F. Then there exists a subset $S \subset \{0,1\}^n$ satisfying the following properties:*

1. $|S| \leq \frac{Q^4}{(1-2\delta)^4}$
2. $\mathcal{A}^{|F\rangle} \overset{\delta}{\simeq} \mathcal{A}^{|H\rangle}$ *for any H satisfying $H|_S = F|_S$*
3. $q_z^F \geq \frac{(1-2\delta)^4}{Q^3} \quad \forall z \in S$

Proof. First we define some notation relative to a partial function $R = \{(x_i, y_i)\}_{i=1}^k$. Recall that $D_R = \{x_i\}_{i=1}^k$ denotes the domain of R. We define the following sequence of functions. For $i = 1, ..., k$, let $F_i = F^{(R\setminus\{(x_i,y_i)\})}$. Let $|\phi_i\rangle$ be the quantum state that results from the execution of $A^{|F_i\rangle}$ (excluding the final measurement). Let $|\phi\rangle$ be the quantum state output by $A^{|F\rangle}$ and let $|\phi_R\rangle$ be the quantum state output by $A^{|F^{(R)}\rangle}$.

Now we describe a procedure to construct the set S from the lemma statement. Let $K = \{0,1\}^n$ and let $S = \emptyset$. Let R be the partial function with the smallest domain D_R, such that $D_R \subseteq K$ and $A^{|F\rangle} \overset{\delta}{\not\simeq} A^{|F^{(R)}\rangle}$. Let x_m be the element of R with the largest total query magnitude $q_{x_m}^F$. Add x_m to S and remove x_m from K. Repeat the above procedure until no such partial function R exists. At this point return S.

We now prove that this set S satisfies all properties from the lemma statement. First note that property (2) is just the termination condition for the above algorithm and so S must satisfy this property. We also note that property (3) implies property (1) since the total query magnitude of $A^{|F\rangle}$ across all points is at most Q. Thus it remains to prove that property (3) is satisfied.

To show that each R contains some x_m such that $q_{x_m}^F \geq \frac{(1-2\delta)^4}{Q^3}$, we prove that (1) each set R is of size at most $\frac{Q^2}{(1-2\delta)^2}$, and (2) that $\sum_{z\in D_R} q_z^F \geq \frac{(1-2\delta)^2}{Q}$. These two claims give a lower bound on the average query magnitude across points in D_R when executing $\mathcal{A}^{|F\rangle}(x)$, and therefore a lower bound on $q_{x_m}^F$.

Proceeding with (1), we will upper bound the size of D_R by showing that $\mathcal{A}^{|F^{(R)}\rangle}$ must have queried each point in D_R with a "large" weight. Since the total weight of all queries across all points for a Q query algorithm is at most Q, this gives an upper bound on the size of D_R. Since R is the partial function with the smallest domain such that $\mathcal{A}^{|F\rangle} \overset{\delta}{\not\simeq} A^{|F^{(R)}\rangle}$, it follows that for any $i \in \{1, ..., |D_R|\}$, we have $\mathcal{A}^{|F\rangle} \overset{\delta}{\simeq} \mathcal{A}^{|F_i\rangle}$ (since F_i differs from F on a subset of the domain of

size $|D_R| - 1 < |D_R|$), and therefore $\mathcal{A}^{|F_i\rangle} \overset{\delta}{\not\approx} \mathcal{A}^{|F^{(R)}\rangle}$. Therefore we have the following inequalities, where the first inequality comes from the fact that $\mathcal{A}^{|F_i\rangle} \overset{\delta}{\not\approx} \mathcal{A}^{|F^{(R)}\rangle}$ together with Lemma 1, and the second follows from the swapping Lemma:

$$(1-2\delta) \leq \||\phi_R\rangle - |\phi_i\rangle\| \leq \sqrt{Q \cdot q_{x_i}^{F^{(R)}}} \implies \frac{(1-2\delta)^2}{Q} \leq q_{x_i}^{F^{(R)}}. \tag{3}$$

In more detail, the first inequality above is due to the following. Since $\mathcal{A}^{|F_i\rangle} \overset{\delta}{\not\approx} \mathcal{A}^{|F^{(R)}\rangle}$, it follows that $|\phi_i\rangle$ and $|\phi_R\rangle$ can be distinguished with probability at least $1-2\delta$, and therefore the trace distance between $|\phi_i\rangle$ and $|\phi_R\rangle$ is at least $(1-2\delta)$. Lemma 1 tells us that the Euclidean distance is lower bounded by the trace distance, and therefore we have $\||\phi_i\rangle - |\phi_R\rangle\| \geq (1-2\delta)$. Since the total query magnitude of a Q query algorithm (and in particular of $\mathcal{A}^{|F^{(R)}\rangle}$) across all points is at most Q, it follows that $|D_R| \leq \frac{Q^2}{(1-2\delta)^2}$.

Proceeding with (2), note that $\mathcal{A}^{|F\rangle} \overset{\delta}{\not\approx} \mathcal{A}^{|F^{(R)}\rangle}$, and therefore

$$(1-2\delta) \leq \||\phi_R\rangle - |\phi\rangle\| \leq \sqrt{Q \sum_{z \in D_R} q_z^F} \implies \frac{(1-2\delta)^2}{Q} \leq \sum_{z \in D_R} q_z^F, \tag{4}$$

where the above inequalities are proved in the same way as they were for Equation 3 above. Combining the above lower bound with the upper bound on $|D_R|$ from before gives

$$\frac{1}{|D_R|} \sum_{z \in D_R} q_z^F \geq \frac{(1-2\delta)^4}{Q^3}.$$

It follows that there is some $x_m \in D_R$ such that $q_{x_m}^F \geq \frac{(1-2\delta)^4}{Q^3}$. □

4.2 The Simulation Algorithm

We now introduce and motivate our classical-query algorithm in Figure 2. Let $\mathcal{A}$ be a δ-deterministic oracle algorithm making $Q_{\mathcal{A}}$ queries to some oracle F. The result in the previous section gives us a set S_F such that the output of $\mathcal{A}$ only depends on the oracle values on points in S_F. Thus, our approach will be to learn S_F, query the oracle only on values in S_F, and then classically simulate the computation of $\mathcal{A}$ using an oracle that is consistent with F on S_F. Since $\mathcal{A}$ only depends on values inside this set, we may define our oracle arbitrarily on all other values.

The algorithm itself is quite simple. We start with an initially empty partial function f_0 which we shall modify as we make classical queries to the oracle. For i starting at 0 we do the following. We classically simulate the execution of $\mathcal{A}^{|\mathbb{1}^{(f_i)}\rangle}$ and record all points in the domain of the oracle with query magnitude exceeding $q_{\min}$. We then query F these points and update our partial function to be consistent with the oracle F on these additional points. Let f_{i+1} be the updated partial function. As we prove below, repeating this process a sufficient number of times results in a partial function whose domain contains S_F, and therefore has all of the required information to compute the most likely output of $\mathcal{A}^{|F\rangle}$.

Algorithm 9 $\mathsf{update}^F(\mathcal{A}, f)$

$S = \left\{x \mid q_x^{\mathbb{1}^{(f)}} \geq \left\lfloor \frac{(1-2\delta)^4}{Q^3} \right\rfloor\right\}$
for $x \in S$ **do**
  if $x \notin D_f$ **then**
    $f = f \cup \{(x, F(x))\}$
  end if
end for
Return f

Algorithm 10 $\mathsf{getPoint}^F(\mathcal{A}, f_0)$

$c = 0$
$y_{\text{old}} = \arg\max_y \left\{\Pr[y \leftarrow \mathcal{A}^{|\mathbb{1}^{(f_0)}\rangle}]\right\}$
$k = \left\lceil \frac{Q^4}{(1-2\delta)^4} \right\rceil$
while $c \leq k$ and $y_{\text{old}} = \arg\max_y \left\{\Pr[y \leftarrow \mathcal{A}^{|\mathbb{1}^{(f_c)}\rangle}]\right\}$ **do**
  $c \leftarrow c + 1$
  $f_c \leftarrow \mathsf{update}^F(\mathcal{A}, f_{c-1})$
end while
return (f_c, c)

Algorithm 11 $\mathsf{simOracle}^F(\mathcal{A})$

$f_0 = \emptyset$
$k = \left\lceil \frac{Q^4}{(1-2\delta)^4} \right\rceil$
for $i = 1, ..., k$ **do**
  $(f'_{i-1}, c_1) \leftarrow \mathsf{getPoint}^F(\mathcal{A}, f_{i-1})$
  if $c_1 = k$ **then** return $\mathbb{1}^{(f'_{i-1})}$ **end if**
  $(f_i, c_2) \leftarrow \mathsf{getPoint}^F(\mathcal{A}, f'_{i-1})$
  if $c_2 = k$ **then** return $\mathbb{1}^{(f_i)}$ **end if**
end for
return $\mathbb{1}^{(f_k)}$.

Figure 2: Classical query algorithm for simulating quantum oracle algorithms.

We now give a brief overview of the algorithms in Figure 4, along with a sketch of the analysis. The proof of correctness for the main simulation algorithm simOracle (Algorithm 11) is given in Lemma 11, and an analysis of the subroutine getPoint (Algorithm 10) is given in Lemma 10.

- $\mathsf{update}^F(\mathcal{A}, f)$: Given classical oracle access to a function $F : \{0,1\}^n \mapsto \{0,1\}^*$, and on input a description of a quantum algorithm $\mathcal{A}$, and a partial function f such that $F \in \mathsf{Func}_{n,m}(f)$, update first classically simulates the computation $\mathcal{A}^{|\mathbb{1}^{(f)}\rangle}$, and finds all $x \in \{0,1\}^n$ such that $q_x^{\mathbb{1}^{(f)}} \geq \left\lfloor \frac{(1-2\delta)^4}{Q^3} \right\rfloor$. The oracle F is then queried on each of these points, and the partial function f is extended so that $f(x) = F(x)$ for each point x on which F was queried.

- $\mathsf{getPoint}^F(\mathcal{A}, f_0)$: Given classical oracle access to a function $F : \{0,1\}^n \mapsto \{0,1\}^*$, and on input a description of a quantum algorithm $\mathcal{A}$, and a partial function f_0 such that $F \in \mathsf{Func}_{n,m}(f_0)$, getPoint first computes the value $y_0 = \arg\max_x \left\{\Pr[y \leftarrow \mathcal{A}^{|\mathbb{1}^{(f_0)}\rangle}]\right\}$. Then, starting at $i = 0$, getPoint updates the partial function by computing $f_{i+1} \leftarrow \mathsf{update}^F(\mathcal{A}, f_i)$. This process terminates either when $\left\lceil \frac{Q^4}{(1-2\delta)^4} \right\rceil$ iterations of this update process have passed, or when for some iteration i, the quantity $\arg\max_y \left\{\Pr[y \leftarrow \mathcal{A}^{|\mathbb{1}^{(f_i)}\rangle}]\right\}$ no longer equals the initial most likely value y_0. In Lemma 10, we will prove several facts about getPoint, the implications of which we discuss here. Starting with some partial function f_i, consider the

following sequence of two invocations of $\mathsf{getPoint}$:

$$(f_i', c_1) \leftarrow \mathsf{getPoint}^F(\mathcal{A}, f_i), \qquad (f_{i+1}, c_2) \leftarrow \mathsf{getPoint}^F(\mathcal{A}, f_i').$$

It turns out that if $c_1 = \left\lceil \frac{Q^4}{(1-2\delta)^4} \right\rceil$, then $\mathcal{A}^{|\mathbb{1}^{(f_i')}\rangle} \overset{\delta}{\simeq} \mathcal{A}^{|F\rangle}$, and similarly if $c_2 = \left\lceil \frac{Q^4}{(1-2\delta)^4} \right\rceil$, then $\mathcal{A}^{|\mathbb{1}^{(f_{i+1})}\rangle} \overset{\delta}{\simeq} \mathcal{A}^{|F\rangle}$. On the other hand, if $c_1, c_2 < \left\lceil \frac{Q^4}{(1-2\delta)^4} \right\rceil$, then there is some $x \in S_F$ which is in the domain of f_{i+1} but not in the domain of f_i. In other words, invoking $\mathsf{getPoint}$ twice either returns a partial function that can be used to simulate $\mathcal{A}^{|F\rangle}$, or it results in learning a new element of S_F. Therefore repeating the above sequence of two invocations $k = \left\lceil \frac{Q^4}{(1-2\delta)^4} \right\rceil$ times should result in the domain of f_k containing all of S_F as $|S_F| \leq k$. With this in mind, the main algorithm is as follows.

- $\mathsf{simOracle}^F(\mathcal{A})$: Given classical oracle access to a function $F : \{0,1\}^n \mapsto \{0,1\}^*$, and on input a description of a quantum algorithm $\mathcal{A}$, $\mathsf{simOracle}$ carries out the following process up to $k = \left\lceil \frac{Q^4}{(1-2\delta)^4} \right\rceil$ times. Compute two invocations of $\mathsf{getPoint}$ as described above. If either of c_1 or c_2 are equal to $\left\lceil \frac{Q^4}{(1-2\delta)^4} \right\rceil$, then return $\mathbb{1}^{(f)}$, where f is the corresponding partial function that was returned by $\mathsf{getPoint}$. Otherwise, return $\mathbb{1}^{(f_k)}$ after k iterations of this process. The proof of correctness is based on the discussion in the previous paragraph. We now proceed to the formal proofs.

Lemma 10. *Let $\mathcal{A}$ be a δ-deterministic quantum algorithm given quantum oracle access to a function F and making Q queries. Let $S_F \subseteq D_F$ be such that $A^{|F\rangle} \overset{\delta}{\simeq} A^{|H\rangle}$ for any H such that $H|_{S_F} = F|_{S_F}$. Let f_0 be some partial function such that $F \in \mathsf{Func}_{n,m}(f_0)$. Suppose that we run the function (described in Algorithm 10)*

$$(f_c, c) \leftarrow \mathsf{getPoint}^F(\mathcal{A}, f_0)$$

and let (f_c, c) be the output. Then:

1. *If $\mathcal{A}^{|F\rangle} \overset{\delta}{\not\simeq} \mathcal{A}^{|\mathbb{1}^{(f_0)}\rangle}$, then there exists some $x \in S_F$ such that x is in the domain of f_c but not in the domain of f_0.*
2. *If $\mathcal{A}^{|F\rangle} \overset{\delta}{\not\simeq} \mathcal{A}^{|\mathbb{1}^{(f_0)}\rangle}$, then there exists some $j < \left\lceil \frac{Q^4}{(1-2\delta)^4} \right\rceil$ such that $\mathcal{A}^{|\mathbb{1}^{(f_j)}\rangle} \overset{\delta}{\not\simeq} \mathcal{A}^{|\mathbb{1}^{(f_0)}\rangle}$, where f_j refers to the state of the partial function after the j'th iteration of the for loop in Algorithm 10.*
3. *If $c = \left\lceil \frac{Q^4}{(1-2\delta)^4} \right\rceil$, then $A^{|\mathbb{1}^{(f_0)}\rangle} \overset{\delta}{\simeq} A^{|F\rangle}$.*

Proof. For notational compactness in the following calculations we let $k = \left\lceil \frac{Q^4}{(1-2\delta)^4} \right\rceil$. As in the lemma statement let i refer to some iteration of the while loop of Algorithm 10. We first note that item (3) is implied by (2). We now prove item (1). To start, we show that for any $i \in \{0, ..., k-1\}$, if $\mathcal{A}^{|\mathbb{1}^{(f_i)}\rangle} \overset{\delta}{\not\simeq} \mathcal{A}^{|F\rangle}$, then there exists some $x \in S_F$ in the domain of f_{i+1} which is not in the domain of f_i. In other words, we show that after another iteration of the while loop, a new element of S_F has

been learned. Assuming that $\mathcal{A}^{|F\rangle} \overset{\delta}{\not\simeq} \mathcal{A}^{|\mathbb{1}^{(f_i)}\rangle}$, we first show there must exist some $x \in S_F \cap S_{\mathbb{1}^{(f_i)}}$ such that $F(x) \neq \mathbb{1}^{(f_i)}(x)$. If there were no such x, then we could define the function

$$F'(x) = \begin{cases} F(x) & x \in S_F \\ \mathbb{1}^{(f_i)}(x) & \text{otherwise.} \end{cases}$$

Applying Lemma 9 we have $A^{|F'\rangle} \overset{\delta}{\simeq} \mathcal{A}^{|F\rangle}$ since $F'|_{S_F} = F|_{S_F}$, and $\mathcal{A}^{|F'\rangle} \overset{\delta}{\simeq} \mathcal{A}^{|\mathbb{1}^{(f_i)}\rangle}$ since $\mathbb{1}^{(f_i)}|_{S_{\mathbb{1}^{(f_i)}}} = F'|_{S_{\mathbb{1}^{(f_i)}}}$, which contradicts the assumption that $\mathcal{A}^{|F\rangle} \overset{\delta}{\not\simeq} \mathcal{A}^{|\mathbb{1}^{(f_i)}\rangle}$. However since $x \in S_{\mathbb{1}^{(f_i)}}$, Lemma 9 tells us that the query magnitude of $\mathcal{A}^{|\mathbb{1}^{(f_i)}\rangle}$ on x will exceed $\left\lfloor \frac{(1-2\delta)^4}{Q^3} \right\rfloor$, and therefore the algorithm $\mathsf{update}(\mathcal{A}, f_i)$ will query F on x. It follows that the domain of f_{i+1} will contain x. This proves item (1) since if this element x is in the domain of f_{i+1} it is also in the domain of f_c since $c \geq i+1$.

To prove item (2), suppose that $\mathcal{A}^{|\mathbb{1}^{(f_0)}\rangle} \overset{\delta}{\simeq} ... \overset{\delta}{\simeq} \mathcal{A}^{|\mathbb{1}^{(f_{k-1})}\rangle}$. It follows from the above that the domain of f_{k-1} contains $k-1$ values in S_F. We also have that $\mathcal{A}^{|\mathbb{1}^{(f_{k-1})}\rangle} \overset{\delta}{\not\simeq} \mathcal{A}^{|F\rangle}$ since one of the assumptions in the lemma statement is that $\mathcal{A}^{|\mathbb{1}^{(f_0)}\rangle} \overset{\delta}{\not\simeq} \mathcal{A}^{|F\rangle}$. It follows from the above that the domain of f_k contains k elements in S_F, but since $|S_F| \leq k$, the domain of f_k contains all of S_F. This, together with Lemma 9 implies that $\mathcal{A}^{|\mathbb{1}^{(f_k)}\rangle} \overset{\delta}{\simeq} \mathcal{A}^{|F\rangle}$ which proves item (2). □

We are now ready to prove the main result below.

Lemma 11 (Oracle simulation)**.** *Let $\mathcal{A}$ be a δ-deterministic quantum algorithm making Q queries to an oracle $F : \{0,1\}^n \mapsto \{0,1\}^*$. Then Algorithm 11 makes at most $\frac{2Q^{12}}{(1-2\delta)^{12}}$ classical queries to F and returns a function $H : \{0,1\}^n \mapsto \{0,1\}^*$ such that $\mathcal{A}^{|F\rangle} \overset{\delta}{\simeq} \mathcal{A}^{|H\rangle}$.*

Proof. Each execution of $\mathsf{getPoint}$ calls update at most $\left\lceil \frac{Q^4}{(1-2\delta)^4} \right\rceil$ times and $\mathsf{getPoint}$ is called at most $2\left\lceil \frac{Q^4}{(1-2\delta)^4} \right\rceil$ times. Each execution of update queries F at most $\left\lceil \frac{Q^4}{(1-2\delta)^4} \right\rceil$ times which gives the stated query complexity.

Now we show correctness of $\mathsf{simOracle}$ (Algorithm 11). We consider two ways that Algorithm 11 can terminate. The first case, is that the algorithm terminates on lines 5 or 7 (we consider these two as a single case), and the second case is that the algorithm terminates on line 9. We show that in either case, the function $\mathbb{1}^{(f)}$ that is returned satisfies $\mathcal{A}^{|\mathbb{1}^{(f)}\rangle} \overset{\delta}{\simeq} \mathcal{A}^{|F\rangle}$. Proceeding with the first case, suppose that $c_b = \lceil \frac{Q^4}{(1-2\delta)^4} \rceil$ for some $b \in \{0,1\}$ as is required to terminate on lines 5 or 7. It follows from Lemma 10 (item 3) that in this case the function f_c output by $\mathsf{getPoint}$ was such that $\mathcal{A}^{|\mathbb{1}^{(f_c)}\rangle} \overset{\delta}{\simeq} \mathcal{A}^{|F\rangle}$.

Now we consider the case that the algorithm terminates on line 9. In keeping with the notation used in Algorithm 11, for the i'th iteration of the for loop, let f_i be the partial function prior to the first invocation of $\mathsf{getPoint}$, let f_i' be the partial function returned by the first invocation of $\mathsf{getPoint}$, and let f_{i+1} be the partial function returned by the second invocation of $\mathsf{getPoint}$. We will prove that there is some $x \in S_F$ such that the domain of f_{i+1} contains x but the domain of f_i does not contain x. It will then follow that after $k = \left\lceil \frac{Q^4}{(1-2\delta)^4} \right\rceil$ iterations of the for loop, the domain of f_k will contain all of S_F since $|S_F| \leq k$. Therefore by Lemma 9, it follows that $\mathcal{A}^{|\mathbb{1}^{(f_k)}\rangle} \overset{\delta}{\simeq} \mathcal{A}^{|F\rangle}$.

First suppose that $\mathcal{A}^{|\mathbb{1}^{(f_i)}\rangle} \overset{\delta}{\not\simeq} \mathcal{A}^{|F\rangle}$. Then it follows from Lemma 10 (item 3) that the domain of f_i' contains an element of S_F which is not in the domain of f_i. It follows that f_{i+1} also contains this point. Now suppose on the other hand that $\mathcal{A}^{|\mathbb{1}^{(f_i)}\rangle} \overset{\delta}{\simeq} \mathcal{A}^{|F\rangle}$. Since by assumption $c < \left\lceil \frac{Q^4}{(1-2\delta)^4} \right\rceil$, the subroutine $\mathsf{getPoint}$ must have terminated because $\mathcal{A}^{|\mathbb{1}^{(f_i)}\rangle} \overset{\delta}{\not\simeq} \mathcal{A}^{|\mathbb{1}^{(f_i')}\rangle}$. It follows that $\mathcal{A}^{|\mathbb{1}^{(f_i')}\rangle} \overset{\delta}{\not\simeq} \mathcal{A}^{|F\rangle}$. It follows once again from Lemma 10 (item 3) that the second invocation of $\mathsf{getPoint}$ will result in the domain of f_{i+1} containing some element of S_F which is not contained in f_i', and therefore was not contained in f_i. □

4.3 Lifting Theorem for PRGs Making Quantum Queries

We can apply the above results to show that our PRG lifting theorem applies even if the PRG itself makes quantum queries to the random oracle. For a δ-deterministic PRG $G_1^{|H\rangle}$ making quantum queries to its oracle, we can apply Lemma 11 to construct a PRG G_2^H making a polynomially related number of classical queries to the same oracle and having (almost) the same output as G_1. It is a straightforward exercise (Lemma 12) to show that for any quantum distinguisher $\mathcal{A}_1$ against G_1, there is a quantum distinguisher $\mathcal{A}_2$ against G_2 with almost the same advantage. As the latter distinguisher is interacting with a PRG making classical queries, our lifting theorem applies and we can construct a classical distinguisher $\mathcal{B}$ against G_2 with advantage almost as large as that of $\mathcal{A}_2$.

Lemma 12. *Let $G_1^{|H\rangle}$ be a δ-deterministic PRG making quantum queries to a random oracle. Let G_2^H be a PRG making classical queries to a random oracle such that $\forall s$,*

$$G_1^{|H\rangle}(s) \overset{\delta}{\simeq} G_2^H(s).$$

Let $A_1^{|H\rangle}$ be a distinguisher with quantum oracle access to H with distinguishing advantage against G_1 given by $\mathsf{Adv}^{\mathsf{PRG}}_{|\mathcal{A}_1\rangle,G_1}$. Then there exists $\mathcal{A}_2^{|H\rangle}$ such that

$$\mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{A}_2,G_2} \geq \mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{A}_1,G_1} - \delta.$$

Proof. On input g, $\mathcal{A}_2(g)$ simply runs $\mathcal{A}_1(g)$ and returns the resulting output. First note that

$$\Pr[\mathsf{Rand}_{\mathcal{A}_1,G_1} = 1] = \Pr[\mathsf{Rand}_{\mathcal{A}_2,G_2} = 1]$$

since H and g are distributed identically in each case. In the case of $\mathsf{PRG}_{\mathcal{A}_1,G_1}$ and $\mathsf{PRG}_{\mathcal{A}_2,G_2}$, H is distributed identically in each case, but g may not be due to the fact that G_1 is only δ-deterministic. It follows from the definition that the two distributions on g are at most δ apart, and therefore the distributions on the outputs of $\mathcal{A}_1$ in each case are also only a distance of δ apart. Therefore

$$|\Pr[\mathsf{PRG}_{\mathcal{A}_1,G_1} = 1] - \Pr[\mathsf{PRG}_{\mathcal{A}_2,G_2} = 1]| \leq \delta.$$

The lemma follows. □

Lemma 13. *Let $G^{|H\rangle} : \{0,1\}^k \mapsto \{0,1\}^\ell$ be a δ-deterministic PRG making Q_G quantum queries to a random oracle $H : \{0,1\}^n \mapsto \{0,1\}^m$, and let $\mathcal{A}$ be a quantum distinguisher making $Q_\mathcal{A}$ quantum queries to the same oracle. Let $\epsilon = \mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{A},G}$. Then there exists a classical query algorithm $\mathcal{B}^H$ making $\mathsf{poly}(k, n, Q_\mathcal{A}, Q_G, \epsilon^{-1})$ classical queries to the random oracle such that*

$$\mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{B},G} \geq \frac{\epsilon}{2} - \delta.$$

Proof. Let G_2^H be a classical-query algorithm that on input s outputs $\arg\max_y\{\Pr[y \leftarrow G^{|H\rangle}(s)]\}$ and makes $Q_{G_2} = \mathsf{poly}(Q_G)$ queries to H. Such an algorithm exists by Lemma 11. It follows from Lemma 12 that there exists a quantum distinguisher $\mathcal{A}_2^{|H\rangle}$ such that

$$\mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{A}_2,G_2} \geq \mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{A},G} - \delta.$$

Applying our lifting theorem for classical PRGs to the quantum distinguisher $\mathcal{A}_2$, it follows that there exists some classical distinguisher $\mathcal{B}^H$ making $\mathsf{poly}(k, n, Q_{\mathcal{A}_2}, Q_G, \epsilon^{-1})$ such that

$$\mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{B},G_2} \geq \frac{1}{2}\mathsf{Adv}^{\mathsf{PRG}}_{\mathcal{A}_2,G_2}.$$

Putting the above two inequalities together we obtain the lemma. □

References

[1] Scott Aaronson and Andris Ambainis. The need for structure in quantum speedups. *Theory of Computing*, 10:133–166, 2014.

[2] Gorjan Alagic, Chen Bai, Jonathan Katz, and Christian Majenz. Post-quantum security of the Even-Mansour cipher. In *Advances in Cryptology—Eurocrypt 2022, Part III*, volume 13277 of *LNCS*, pages 458–487. Springer, 2022.

[3] Per Austrin, Hao Chung, Kai-Min Chung, Shiuan Fu, Yao-Ting Lin, and Mohammad Mahmoody. On the impossibility of key agreements from quantum random oracles. In *Advances in Cryptology—Crypto 2022, Part II*, volume 13508 of *LNCS*, page 165–194. Springer, 2022.

[4] James Bartusek, Fuyuki Kitagawa, Ryo Nishimaki, and Takashi Yamakawa. Obfuscation of pseudo-deterministic quantum circuits. In *Proc. 55th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1567–1578. ACM, 2023.

[5] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proc. 1st ACM Conference on Computer and Communications Security (CCS)*, pages 62–73. ACM, 1993.

[6] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random oracles in a quantum world. In *Advances in Cryptology—Asiacrypt 2011*, volume 7073 of *LNCS*, pages 41–69. Springer, 2011.

[7] Juliane Krämer and Patrick Struck. Encryption schemes using random oracles: From classical to post-quantum security. In *11th Intl. Conference on Post-Quantum Cryptography (PQCrypto)*, volume 12100 of *LNCS*, pages 539–558. Springer, 2020.

[8] Aran Nayebi, Scott Aaronson, Aleksandrs Belovs, and Luca Trevisan. Quantum lower bound for inverting a permutation with advice. *Quantum Information and Computation*, 15(11–12):901–913, 2015.

[9] Fang Song. A note on quantum security for post-quantum cryptography. In *6th Intl. Workshop on Post-Quantum Cryptography (PQCrypto) 2014*, volume 8772 of *LNCS*, pages 246–265. Springer, 2014.

[10] Takashi Yamakawa and Mark Zhandry. Classical vs. quantum random oracles. In *Advances in Cryptology—Eurocrypt 2021, Part II*, volume 12697 of *LNCS*, pages 568–597. Springer, 2021.

[11] Takashi Yamakawa and Mark Zhandry. Verifiable quantum advantage without structure. In *63rd Annual Symp. on Foundations of Computer Science (FOCS)*, pages 69–74. IEEE, 2022.

[12] Mark Zhandry. How to record quantum queries, and applications to quantum indifferentiability. In *Advances in Cryptology—Crypto 2019, Part II*, volume 11693 of *LNCS*, pages 239–268. Springer, 2019.

[13] Mark Zhandry. How to construct quantum random functions. *J. ACM*, 68(5):33:1–33:43, 2021.

[14] Jiang Zhang, Yu Yu, Dengguo Feng, Shuqin Fan, and Zhenfeng Zhang. On the (quantum) random oracle methodology: New separations and more, 2019. Available at `https://ia.cr/2019/1101`.