How to Construct Quantum Random Functions

MARK ZHANDRY, Stanford University, USA

Pseudorandom functions (**PRFs**) are one of the foundational concepts in theoretical computer science, with numerous applications in complexity theory and cryptography. In this work, we study the security of PRFs when evaluated on quantum superpositions of inputs. The classical techniques for arguing the security of PRFs do not carry over to this setting, even if the underlying building blocks are quantum resistant. We therefore develop a new proof technique to show that many of the classical PRF constructions remain secure when evaluated on superpositions.

CCS Concepts: • **Theory of computation → Cryptographic primitives**; **Quantum complexity theory;**

Additional Key Words and Phrases: Quantum, security proofs, pseudorandom function

ACM Reference format:
Mark Zhandry. 2021. How to Construct Quantum Random Functions. *J. ACM* 68, 5, Article 33 (August 2021), 43 pages.
https://doi.org/10.1145/3450745

1 INTRODUCTION

Quantum computers, which harness quantum physics to perform computations, promise to fundamentally alter the study of computer science. Owing to their enhanced computational power—most famously due to Grover's [27] and Shor's [39] algorithms—and inherently different operating model, new ideas are needed to reason about abilities and limitations of quantum computation.

Pseudorandom functions. In this work, we focus on the case of ***pseudorandom functions*** (**PRFs**). PRFs, as defined by Goldreich et al. [23], are keyed functions that "look like" uniformly random functions. That is, any computationally bounded adversary which can make arbitrary queries to the function, but does not know the secret key, will be unable to distinguish the function outputs from those of a truly random function. In the classical world, PRFs are one of the foundational objects in theoretical computer science, with numerous applications in complexity theory and cryptography. In complexity theory, PRFs are the canonical example of unlearnable functions, and are used by Razborov and Rudich [38] to explain why many circuit lower bound techniques are incapable of separating $\mathcal{P}$ from $\mathcal{NP}$. In cryptography, PRFs are one of the central building blocks, being used to encrypt and authenticate messages, among numerous other applications.

This work was supported by NSF and DARPA.
Author's address: M. Zhandry, 940 Stewart Drive, Sunnyvale, CA 94085; email: mzhandry@gmail.com.
Authors current address: Princeton University and NTT Research.

0004-5411/2021/08-ART33 $15.00
https://doi.org/10.1145/3450745

A fundamental question is how to build PRFs, and what computational assumptions are required to justify their existence. A number of works have shown how to build PRFs from hard algebraic problems [12, 14, 32, 34–36]. Moreover, one of the major successes of classical cryptography is showing that PRFs can be built from any one-way function [23, 29], one of the weakest cryptographic objects. One-way functions can in turn be built from numerous hard computational problems. On the other hand, many standardized PRFs such as DES or AES are directly conjectured to be secure PRFs, the conjecture being supported by extensive cryptanalysis.

PRFs in a quantum world. When moving to a world with quantum computers, pseudorandom functions will similarly be fundamental. At this point, we distinguish between two possible ways of formally specifying PRFs in the quantum setting. The first considers security against quantum algorithms, but restricts the algorithms to still making classical queries to the function. The second also considers quantum algorithms, but allows the algorithms to query the function on *quantum* states. Informally, such a query applies the map

$$\sum_{x,y} \alpha_{x,y}|x,y\rangle \mapsto \sum_{x,y} \alpha_{x,y}|x,y \oplus \mathsf{PRF}(x)\rangle$$

to the algorithm's state. The term on the left represents a superposition of inputs that assigns a weight to *every* input to the PRF. To disambiguate between the two notions, we call the weaker classical query notion "classical-query security," whereas we call the stronger quantum query notion "quantum-query security" or simply "quantum security." We stress that in either case, PRF remains a classically computable function.

The focus of this work will be the stronger quantum security notion for PRFs, which is far more useful. Quantum-secure PRFs represent natural examples of functions that cannot be learned efficiently using a quantum computer, even given evaluations on quantum inputs. Quantum-secure PRFs are also a useful tool in cryptographic security proofs [8] in the quantum setting. Moreover, they have been used to build cryptosystems secure against "superposition attacks" [3, 4, 17, 18, 20, 24, 28, 40, 46], namely, attacks where the adversary can interact with the honest parties using quantum states. Quantum-secure PRFs have also recently been used to build pseudorandom quantum states [15, 30], namely, efficiently sample-able states that are computationally indistinguishable from truly random states.

1.1 Proving Security in a Quantum World

In order to formally prove the security of cryptographic objects like pseudorandom functions, two ingredients are needed. First, the current state of complexity theory means that most cryptographic objects cannot be proved secure unconditionally. This means that some starting computational assumption is needed. Common computational assumptions in cryptography include the hardness of algebraic problems such as factoring, discrete logarithms, finding short vectors in lattices, solving multivariate polynomial equations, or certain problems regarding isogenies on elliptic curves. Alternatively, in order to give the most general and modular results, it is common to abstract the features of these algebraic assumptions. For example, each of the aforementioned computational problems readily yields a one-way function: a function that is easy to compute but computationally infeasible to invert. Then, one may make the *generic* assumption that *some* one-way function exists, rather than making an assumption about any concrete instantiation.

The second ingredient is a sound security proof, demonstrating that the chosen starting assumption implies the desired security of the cryptographic object in question. Such proofs are most often achieved by proving the contrapositive: showing that any efficient attack on

the cryptographic object yields an efficient algorithm violating the underlying computational assumption. Such contrapositive proofs are called reductions. Take, for example, the celebrated result of Håstad et al. [29] that one-way functions can be used to build **pseudorandom *generators* (PRGs)**—deterministic length-expanding functions which map a random seed to an output that is computationally indistinguishable from uniform. The proof takes any supposed efficient adversary A for the pseudorandom generator, and uses it to construct an efficient adversary B for the underlying one-way function. By the assumption that there is no such efficient adversary for the one-way function, they prove that the supposed adversary A actually cannot exist.

Proving quantum security. When moving to the quantum setting, the same two ingredients are needed. However, now the underlying computational assumption needs to be strengthened to require that the computational problem is hard for *quantum* computers. Due to Shor's polynomial-time quantum algorithm [39] for factoring and discrete logarithms, the hardness of these problems cannot be used for proving quantum security. On the other hand, finding shortest vectors, solving multivariate polynomial equations, or certain isogeny problems are widely conjectured to be hard for quantum computers. In particular, it is widely believed that one-way functions secure against quantum computers exist.

Using a quantum hardness assumption alone is not enough, as one must also make sure to use a reduction that is compatible with quantum algorithms. Concretely, a classical security reduction converts any efficient *classical* attack on the object into an efficient *classical* algorithm for the underlying hard problem. However, in order to prove quantum security using a quantum hard assumption, we need to convert any efficient *quantum* attack on the object into an efficient *quantum* algorithm for the hard problem. Such a quantum reduction is not necessarily implied by a classical reduction.

The good news is that most reductions in cryptography are essentially independent of the computational model of the attacker. Namely, most reductions treat the attacker as a black box that takes classical inputs and produces classical outputs, but the reductions are otherwise independent of the inner workings of the black box. The reduction simply runs the attacker black box as a subroutine, performing some additional operations in order to solve the underlying hard problem. These reductions, while typically stated as dealing with classical algorithms, immediately work for quantum adversaries—or essentially any sufficiently expressive computational model—as well. For these reductions, all that is needed is to replace every instance of the words "polynomial time" with "quantum polynomial time," and the reduction will work for quantum algorithms as well. For example, inspecting the proof that one-way functions imply pseudorandom generators shows that the reduction has this form. Thus, even though Håstad et al. [29] only claim to prove their result in the classical setting, it immediately carries over into the quantum setting:

Theorem 1.1 (Implicit in [29]). *Assuming the existence of one-way functions secure against* quantum *computers, there exist (length-doubling) pseudorandom generators secure against* quantum *computers.*

This observation regarding Theorem 1.1 was made by Aaronson and Christiano [1, 2].

The case of pseudorandom functions. For proving the security of PRFs, however, the situation is more delicate. An attacker for a PRF is no longer just an algorithm that takes a classical input and produces a classical output. Instead, an attacker is *interactive*, making various queries to the PRF and seeing the responses. In the classical setting, queries and responses are also classical. Existing security proofs for PRFs (e.g., [14, 23, 34]) still treat the adversary as a black box, but an interactive black box that exchanges classical messages.

When we move to the quantum setting, classical security proofs will still typically be able to justify the classical-query security of PRFs, since the classical proofs tend to be independent of the adversary's computational model. Concretely, by inspecting the classical construction of PRFs from pseudorandom generators, we see that:

Theorem 1.2 (Implicit in [23]). *Assuming the existence of length-doubling PRGs secure against* quantum *computers, there exist classical-query secure PRFs.*

However, when we consider the more powerful quantum security notion for PRFs, the classical security proofs no longer apply. Indeed, in the quantum security setting, the interface of the adversary has changed to allow quantum queries. Proofs can still treat the adversary as an interactive black box, but have to handle black boxes that exchange *quantum* messages. The existing security proofs generally do *not* work for adversaries with such a quantum interface, and as such cannot be used to prove PRFs to be quantum-secure, even if the starting assumption is quantum hard. As a result, prior to this work, the quantum security of any PRF construction could not be based on any standard quantum hardness assumption.

Example: The GGM construction. To illustrate what goes wrong, consider the construction of PRFs due to Goldreich et al. [23][1]. They show how to build a pseudorandom function PRF from any length-doubling pseudorandom generator G, a construction known as the GGM construction.

At a high level, implicit in the GGM construction is a binary tree of depth n. The root contains the key, and the values of a node's children are derived by applying G to the value at that node. Each child will contain half of the output; since G is length-doubling, each node's value will have the same length as the parent. The n-bit input to PRF corresponds to a leaf of this tree, and the output is the value contained in the leaf. The entire tree is implicit; only the root is stored. To evaluate PRF, we start at the root, and follow the path from root to the leaf corresponding to the input, generating the necessary part of tree on the fly.

The security proof consists of two main "hybrid" arguments: the first across levels of the tree, and the second across the nodes in a particular level. In the first step, one introduces a sequence of hybrid experiments where the first several levels of the tree are removed, and the nodes at the next lever are replaced with uniformly random values; all nodes after this level are derived using G as before. If one removes no levels, this corresponds to PRF. On the other hand, if one removes all levels but the leaves, then the leaves contain independent random values, corresponding to a uniformly random function. Thus, if an adversary can break PRF, it means it can distinguish the first and last of these hybrid experiments. By the triangle inequality, there are two adjacent hybrid experiments—corresponding to layers i and $i-1$—that are also distinguishable, though the distinguishing advantage is reduced by a factor of n corresponding to the number of hybrid experiments/levels of the tree. In other words, if A breaks PRF with probability ϵ, it will distinguish the two adjacent hybrid experiments with probability at least ϵ/n.

In the second step, one introduces another sequence of hybrid experiments, where the first several *nodes* within the $(i-1)$th layer are replaced with random. Done naively, the result is that there must now exist two adjacent hybrid experiments which A distinguishes with probability at least $\epsilon/n2^{i-1}$, since there are 2^{i-1} nodes in this level. This probability is potentially exponentially small, and is too small to reach a contradiction for standard polynomially hard PRGs. Instead, the more clever reduction from [23] uses the fact that a classical efficient adversary can only query PRF on polynomially many points, so the paths used to evaluate PRF on the various queries only "visit" polynomially many nodes in each level. Essentially, we can prune the nodes not visited, and therefore only need polynomially many hybrids for the second hybrid argument. The result is that

[1]To the best of our knowledge, this issue was first pointed out by Aaronson [1].

A actually distinguishes two adjacent hybrid experiments in this second step with probability at least ϵ/nq, where q is the number of queries made by A. From here, one can construct an adversary B that breaks G with probability ϵ/nq.

Moving to the quantum setting, A may query PRF on a superposition—a quantum state that assigns non-zero weight to potentially every single input. In response, A expects to see a superposition of potentially every single output of PRF. The result is that even answering a single quantum query could require visiting all nodes in the tree. This means that we can no longer prune the un-visited parts of the tree, and seemingly have to incur the entire 2^{i-1} loss in the second hybrid step, resulting in an adversary B with exponentially small probability of breaking G. The usual "polynomial" hardness notions for cryptographic primitives like PRGs only stipulate that polynomial-time adversaries have a "negligible" probability of success, meaning smaller than any inverse polynomial. The exponentially small success probability of B therefore does not yield the desired contradiction.

It is possible to revive security by relying on "sub-exponentially hard" PRGs that stipulate much smaller success probabilities (see Section 1.3 for a brief discussion). However, this style of argument seems incapable of proving security relative to standard polynomial hardness assumptions. All existing security proofs for pseudorandom functions from standard assumptions suffer from similar weaknesses. The prior work therefore leaves open the following natural and fundamental question:

Are there any PRFs that can be proven quantum-secure under standard quantum polynomial hardness assumptions? In particular, can quantum-secure PRFs be built from any quantum polynomially hard one-way function?

1.2 Our Results

First, we demonstrate that not all classically secure PRFs are quantum secure, even if the underlying computational assumptions are quantum hard. Concretely, we show that:

Theorem 3.1. *Assuming the existence of one-way functions secure against quantum computers, there exists a classical-query-secure PRF that is not quantum-secure.*

This, in particular, shows that the classical-query security of a PRF cannot generically imply its quantum security. As a result, one must examine concrete PRF constructions to determine if they are quantum-secure.

We next develop a new quantum-compatible approach to proving the security of PRFs. Interestingly, we demonstrate how to apply our approach to several *existing* PRF constructions, obtaining quantum security under the quantum hardness of the same computational problems as in the classical setting. At a high level, all existing proofs break down in the quantum setting for similar reasons as in the second step above. We show how to replace this step with a new proof that still makes sense when allowing quantum queries. Concretely, we prove the security of the following.

Quantum-secure PRFs from any one-way function. Our first and main result is to show that quantum-secure PRFs can be built from any quantum-secure one-way function. The construction is *identical* to the classical construction: we rely on Theorem 1.1 to first get a (length-doubling) pseudorandom generator against quantum algorithms, and then apply the GGM construction. For the second step of the proof, we provide a new security analysis, giving the following theorem.

Theorem 5.5. *The GGM construction is quantum-secure when instantiated with a (length-doubling) PRG that is secure against quantum computers.*

Combined with Theorem 1.1, this gives

Corollary 1.3. *Assuming the existence of one-way functions secure against* quantum *computers, there exist quantum-secure PRFs.*

PRFs with low circuit depth. We show the versatility of our proof techniques by also applying them to other classical constructions. The next construction we investigate is that of Naor and Reingold [34]. They show how to build a PRF from any pseudorandom synthesizer (PRS), which is a generic object of a similar flavor to PRGs. The advantage of [34] is that the depth of the evaluation circuit for the PRF grows logarithmically with the input size, as opposed to linearly as with the GGM construction.

As with the GGM construction, the classical proof from [34] fails in the quantum setting. We devise a new proof, giving

Theorem 6.3. *The Naor-Reingold construction is quantum-secure when instantiated with a PRS that is secure against quantum computers.*

Banerjee et al. [14] show how to construct pseudorandom synthesizers with logarithmic depth, under the assumed hardness of the **Learning With Errors** (**LWE**) problem [37]. LWE is the main contender for building quantum-resistant cryptography. It also has numerous other applications throughout cryptography such as giving fully homomorphic encryption [16, 21], functional encryption [25], and certain types of program obfuscation [26, 42].

The security proof in [14] readily extends to justify the security of their synthesizer construction against quantum attacks. Combined with Theorem 6.3, this gives a quantum-secure PRF computable in log-squared depth (NC^2) under the LWE assumption.

Banerjee et al. also give a direct log-squared depth PRF construction from LWE. We show that the same construction is also quantum-secure, assuming the quantum hardness of LWE, though our proof requires tweaking the parameters of their construction slightly.

Theorem 7.5 (Informal). *The Banerjee-Peikert-Rosen construction with slightly modified parameters is quantum secure, assuming LWE is hard for quantum computers.*

In the following subsections, we outline our main techniques.

1.3 Are Quantum Oracles Better Than Samples?

In all thee of the PRF constructions mentioned above, the proof can essentially be broken down into two steps. The first step converts any algorithm A breaking the PRF into an algorithm B for a problem which roughly has the following form. B is given oracle access to a function, where each output is sampled independently from a distribution D_1, or where each output is sampled independently from a distribution D_2. B's goal is to distinguish the two cases. The second step shows how to convert B into an algorithm C which distinguishes a single sample of D_1 from D_2. The second step shows that having classical access to an oracle generating samples is "no better" than having a single sample.

In the case of GGM, the first step corresponds to the hybrid over levels, and the second step corresponds to the hybrid across a level. The distribution D_1 corresponds to random outputs of the pseudorandom generator, and D_2 is uniform. In the Naor-Reingold and Banerjee-Peikert-Rosen constructions, the problem B solves is a bit different, but has a similar flavor.

In all three cases, the first step carries over essentially un-modified to the quantum query setting, resulting in an algorithm B which makes *quantum* queries to its oracles. However, in each of the three cases, the second existing proof of the second step breaks down, failing to give an algorithm C.

To complete the quantum security proofs, we then need to replace the construction of C with a quantum friendly proof. Our main technical result solves exactly this problem, showing that having *quantum* access to an oracle generating samples is still no better than having a single sample.

THEOREM 4.5. *Let D_1 and D_2 be efficiently sampleable distributions on a set $\mathcal{Y}$, and let $\mathcal{X}$ be some other (potentially exponential-sized) set. Let O_1 and O_2 be the distributions of functions from $\mathcal{X}$ to $\mathcal{Y}$ where for each $x \in \mathcal{X}$, $O_i(x)$ is chosen independently according to D_i. Then if A is an efficient quantum algorithm that uses quantum queries to distinguish oracles drawn from O_1 from oracles drawn from O_2, we can construct an efficient quantum algorithm B that distinguishes samples from D_1 and D_2.*

To complete the GGM proof in the quantum setting, we apply Theorem 4.5 with D_1 as the uniform distribution, and D_2 as the distribution of outputs of the pseudorandom generator.

The classical version of Theorem 4.5 is straightforward to prove. Any classical algorithm A making q queries to an oracle O only sees q outputs. Thus, given q samples from D_1 or D_2, we can lazily simulate the oracles O_1 or O_2, getting an algorithm that distinguishes q samples of D_1 from q samples of D_2. A simple hybrid argument shows how to get an algorithm that distinguishes one sample, with the distinguishing advantage reduced by a factor q.

In the quantum setting, any quantum algorithm A making even a single quantum query to O_i gets to "see" all the outputs at once, meaning we need exponentially many samples to simulate O_i exactly. This breaks the proof above since we can no longer lazily simulate the oracles O_1, O_2 using just q samples. We now discuss how Theorem 4.5 can be proved in certain special cases.

The statistical setting. Boneh et al. [8] prove a version of Theorem 4.5, for the case where D_1 and D_2 are *statistically* close.[2] However, the techniques are inherently limited to the statistical setting, as they crucially rely on the statistical distance between D_1 and D_2 being small to directly bound the distinguishing probability of O_1, O_2. In contrast, if D_1, D_2 are only computationally indistinguishable—as in the proofs of all of the PRF constructions—they may have large statistical distance, and instead a reduction is needed rather than a statistical argument. Proving Theorem 4.5 therefore requires entirely different techniques.

A simple case: (Sub)exponential hardness. Suppose that D_1, D_2 are *extremely* hard to distinguish: namely, any quantum polynomial time algorithm has at most a probability 2^{-n^c} of distinguishing, for some constant $c > 0$. Suppose further that $|\mathcal{X}| \leq 2^{n^d}$ for some constant $d < c$. We can then view the oracle seen by the adversary as a list of $|\mathcal{X}|$ samples from either D_1 or D_2. In this case, we can do a similar hybrid argument as in the classical setting, except perform the hybrid across all of $\mathcal{X}$. This will result in a quantum polynomial distinguisher for a single sample, but with a very large loss: if the original adversary A distinguished the two oracles with probability ϵ, the new polynomial-time adversary B will distinguish a single sample with probability $\epsilon 2^{-n^d}$. Fortunately, using the extreme hardness of distinguishing samples, we see that ϵ must be negligible.

Such an approach can be used to prove the security of the GGM construction assuming a subexponentially secure PRG, as discussed by Aaronson [1].

Another simple case: Random self-reductions. If (D_1, D_2) admits a *random self-reduction*, then it is straightforward to simulate these exponentially many samples given just a single sample, with a *tight* reduction. A random self-reduction is a randomized procedure R that roughly maps samples of D_1 to independent samples of D_1, and samples of D_2 to independent samples of D_2. More precisely, $R(s; z)$ takes as input a sample $s \in \mathcal{Y}$ and a random string z in some domain $\mathcal{Z}$, and outputs

[2]Moreover, they require D_1 to be the uniform distribution. On the other hand, their result shows that O_1 and O_2 are indistinguishable under polynomially many quantum queries, even to computationally unbounded adversaries.

another sample in $\mathcal{Y}$ with the property that the following two distributions are identical for both $i = 1$ and $i = 2$:

$$(s_1, s_2) : s_1 \leftarrow D_i, z \leftarrow \mathcal{Z}, s_2 = R(s_1, z) \text{ and } (s_1, s_2) : s_1, s_2 \leftarrow D_i. \tag{1.1}$$

If we have such a random self-reduction for (D_1, D_2), we can easily simulate O_i given only a *single* sample s from D_i as $O_i(x) = R(s; P(x))$ where $P : \mathcal{X} \to \mathcal{Z}$ is a random function. In order to make the simulation efficient, we will actually replace P with a $2q$-wise independent function[3] as in Zhandry [43].

For GGM, we can therefore already give a security proof if the underlying pseudorandom generator admits a random self-reduction. Such generators can be built from number-theoretic assumptions such as the hardness of factoring or the decisional Diffie-Hellman problem [13], but these problems are known to be broken by quantum computers. Certain lattice problems such as *Learning With Errors* also give pseudorandom generators with self-reductions, and are presumably resistant to quantum attacks. However, we note that the self-reductions are imperfect, as the self-reduction adds "noise" to the samples.[4] In other words, in order for R to output a sample from D_i, the input needs to be a sample from a "less noisy" distribution D'_i. On one hand, we can get a quantum-secure PRF by assuming D'_1 is indistinguishable from D'_2, which is a standard, though slightly stronger, assumption.

On the other hand, this simulation technique is far from general. For example, the existence of a random self-reduction of the form above implies that distinguishing the two distributions is in $\mathcal{NP} \cap co\text{-}\mathcal{NP}$. To see this, fix samples s_1^* from D_1 and s_2^* from D_2. Then $s = R(s_i, z)$ if and only if s is in the support of D_i, and hence z is a witness to s being sampled from D_i. As one-way functions are not believed to imply hardness in $\mathcal{NP} \cap co\text{-}\mathcal{NP}$ [10], this means that this approach cannot yield a quantum-secure PRF from any one-way function.

A general solution: Small-range distributions. We finally discuss our general solution. While we cannot in general lazily simulate the oracles O_i given q samples from D_i, we show that we can *approximately* simulate O_i given r samples, where r is a polynomial somewhat larger than q. The idea behind our simulation is natural: for each input, set the output to be one of the samples, chosen at random from the collection of samples.

We call functions generated in this way *small-range functions*, and the distribution induced by D_i we call *small-range distributions*. Small-range functions are very different from the corresponding large-range distribution O_i. In particular, while O_i has an exponentially large image size, the image size of the small-range function is at most the polynomial r, meaning it has far more collisions. Yet, given only oracle access, it is not obvious how to distinguish the two distributions, short of finding such a collision. Using quantum collision finding algorithms [9], we can find a collision using $O(r^{1/3})$ queries. We develop new quantum lower-bound techniques based on the polynomial method to show that this is optimal: no quantum algorithm making $o(r^{1/3})$ quantum queries can distinguish between a small-range distribution and the corresponding large-range distribution. Thus, by using enough samples, we can "approximately" simulate the needed large-range distribution.

The result is that any distinguisher A between the oracles O_1 and O_2 also distinguishes between two small-range distributions: the distribution induced by D_1 and the distribution induced by D_2. This means we can use A to distinguish a polynomial number of samples of D_1 from the same

[3] k-wise independent functions are those where the marginal distribution for any subset of k outputs is uniformly distributed.

[4] Another issue is that the distributions in Equation (1.1) are only statistically close, not identical. In other words, s_2 is not quite independent of s_1. It thus requires additional effort to show that O_i is simulated correctly.

number of samples of D_2. Like in the classical case, a simple hybrid argument shows how to distinguish just one sample.

1.4 Subsequent Work

Subsequent to the initial publication of our work, a number of works have built off of our techniques or used quantum-secure PRFs in their results.

Several works have studied cryptosystems in the setting of quantum superposition attacks. Quantum-secure PRFs are a useful tool in this setting. Boneh and Zhandry [17] show that quantum-secure PRFs give a version of quantum-secure **message authentication codes** (**MACs**). This result was strengthened to an even stronger quantum notion by Alagic et al. [3]. Zhandry [45] and Hosoyamada and Iwata [28] show how to construct pseudorandom *permutations* secure against quantum queries, assuming a quantum-secure PRF as a building block. In turn, Gagliardoni et al. [24] use such quantum-secure PRPs to build encryption secure in a strong quantum attack model. Song and Yun [40] show that certain "domain extenders" for PRFs work in the quantum-security setting, giving quantum-secure PRFs with different efficiency tradeoffs. Combining with our results, all these constructions can be instantiated with any one-way function secure against quantum computers. Alagic and Russel [4] use quantum-secure PRFs plus hidden shifts to build various constructions of symmetric-key cryptosystems.

Our technique of embedding a few samples into a large oracle has also been used by several works. Boneh and Zhandry [18] use this technique, combined with some new quantum techniques, to construct quantum-secure signatures from various building blocks. Ambainis et al. [5] use the technique to show that several classical proof systems and proofs of knowledge are *insecure* in the quantum setting.

Finally, our new technique for arguing quantum indistinguishability of oracle distributions was used by Zhandry [44] to prove that optimal query complexity bounds for the collision problem on random functions, as well as the element distinctness problem. It was also used more recently to argue about the indistinguishability of the "sponge" construction [19].

2 PRELIMINARIES AND NOTATION

We say that $\epsilon = \epsilon(n)$ is negligible if, for all polynomials $p(n)$, $\epsilon(n) < 1/p(n)$ for large enough n.

For an integer k, we will use non-standard notation and write $[k] = \{0, \ldots, k-1\}$ to be the set of non-negative integers less than k. We write the set of all n bit strings as $[2]^n$. Let $x = x_1 \ldots x_n$ be a string of length n. We write $x_{[a,b]}$ to denote the substring $x_a x_{a+1} \ldots x_b$.

2.1 Functions and Probabilities

Given two sets $\mathcal{X}$ and $\mathcal{Y}$, define $\mathcal{Y}^{\mathcal{X}}$ as the set of functions $f : \mathcal{X} \to \mathcal{Y}$. If a function f maps $\mathcal{X}$ to $\mathcal{Y} \times \mathcal{Z}$, we can think of f as two functions: one that maps $\mathcal{X}$ to $\mathcal{Y}$ and one that maps $\mathcal{X}$ to $\mathcal{Z}$. In other words, we can equate the sets of functions $(\mathcal{Y} \times \mathcal{Z})^{\mathcal{X}}$ and $\mathcal{Y}^{\mathcal{X}} \times \mathcal{Z}^{\mathcal{X}}$.

Given $f \in \mathcal{Y}^{\mathcal{X}}$ and $g \in \mathcal{Z}^{\mathcal{Y}}$, let $g \circ f$ be the composition of f and g. That is, $g \circ f(x) = g(f(x))$. If $\mathcal{F} \subseteq \mathcal{Y}^{\mathcal{X}}$, let $g \circ \mathcal{F}$ be the set of functions $g \circ f$ for $f \in \mathcal{F}$. Similarly, if $\mathcal{G} \subseteq \mathcal{Z}^{\mathcal{Y}}$, $\mathcal{G} \circ f$ is the set of functions $f \circ g$ where $g \in \mathcal{G}$. Define $\mathcal{G} \circ \mathcal{F}$ accordingly.

Given a distribution D and some event $\mathtt{event}$, we write $\Pr_{x \leftarrow D}[\mathtt{event}]$ to represent the probability that $\mathtt{event}$ happens when x is drawn from D. For a given set $\mathcal{X}$, we will sometimes abuse notation and write $\mathcal{X}$ to denote the uniform distribution on $\mathcal{X}$.

Given a distribution D on $\mathcal{Y}^{\mathcal{X}}$ and a function $g \in \mathcal{Z}^{\mathcal{Y}}$, define the distribution $g \circ D$ over $\mathcal{Z}^{\mathcal{X}}$ where we first draw f from D, and output the composition $g \circ f$. Given $f \in \mathcal{Y}^{\mathcal{X}}$ and a distribution E over $\mathcal{Z}^{\mathcal{X}}$, define $E \circ f$ and $E \circ D$ accordingly.

Given a distribution D on a set $\mathcal{Y}$, and another set $\mathcal{X}$, define $D^{\mathcal{X}}$ as the distribution on functions in $\mathcal{Y}^{\mathcal{X}}$ where the output for each input is chosen independently according to D.

The distance between two distributions D_1 and D_2 over a set $\mathcal{X}$ is

$$|D_1 - D_2| = \frac{1}{2}\sum_{x\in\mathcal{X}} |D_1(x) - D_2(x)|.$$

Notice that the distance is maximized when D_1, D_2 have disjoint supports; in this case $|D_1 - D_2| = 1$. On the other hand, $|D_1 - D_2| = 0$ if and only if $D_1 = D_2$. If $|D_1 - D_2| \leq \epsilon$, we say D_1 and D_2 are ϵ-close. If $|D_1 - D_2| \geq \epsilon$, we say they are ϵ-far.

2.2 Quantum Computation

Here, we give a brief background on quantum computation, and refer the reader to Nielsen and Chuang [33] for a more in-depth discussion.

A quantum system Q is defined over a finite set B of classical states. We will generally consider $B = \{0,1\}^n$. A **pure** state over Q is an L_2-normalized vector in $\mathbb{C}^{|B|}$, which assigns a (complex) weight to each element in B. Thus, the set of pure states forms a complex Hilbert space. A **qubit** is a quantum system defined over $B = \{0,1\}$. Given a quantum system Q_0 over B_0 and a quantum system Q_1 over B_1, we can define the product system $Q = Q_0 \times Q_1$ over $B = B_0 \times B_1 = \{(b_0, b_1) : b_0 \in B_0, b_1 \in B_1\}$. Given a state $v_0 \in Q_0$ and $v_1 \in Q_1$, we define the product state $v_0 \otimes v_1$ in the natural way. An n-qubit system is then $Q = Q_0^{\oplus n}$ where Q_0 is a single qubit.

Bra-ket notation. We will think of pure states as column vectors. The pure state that assigns weight 1 to x and weight 0 to each $y \neq x$ is denoted $|x\rangle$. The set $\{|x\rangle\}$ therefore gives an orthonormal basis for the Hilbert space of pure states; this basis is called the "computational basis." For a pure state $|\phi\rangle$, we will denote the conjugate transpose as the row vector $\langle\phi|$.

Evolution of quantum systems. A pure state $|\phi\rangle$ can be manipulated by performing a unitary transformation U to the state $|\phi\rangle$. We will denote the resulting state as $|\phi'\rangle = U|\phi\rangle$.

Measurements. A pure state $|\phi\rangle$ can be measured; the measurement outputs the value x with probability $|\langle x|\phi\rangle|^2$. The normalization of $|\phi\rangle$ ensures that the distribution over x is indeed a probability distribution. After measurement, the state "collapses" to the state $|x\rangle$. Notice that subsequent measurements will always output x, and the state will always stay $|x\rangle$.

If $Q = Q_0 \times Q_1$, we can perform a **partial measurement** in the system Q_0 or Q_1. If $|\phi\rangle = \sum_{x\in B_0, y\in B_1} \alpha_{x,y}|x,y\rangle$, partially measuring in Q_0 will give x with probability $p_x = \sum_{y\in B_1} |\alpha_{x,y}|^2$. $|\phi\rangle$ will then collapse to the state $\sum_{y\in B_1} \frac{\alpha_{x,y}}{\sqrt{p_x}}|x,y\rangle$. In other words, the new state has support only on pairs of the form (x,y) where x was the output of the measurement, and the weight on each pair is proportional to the original weight in $|\phi\rangle$. Notice that subsequent partial measurements over Q_0 will always output x, and will leave the state unchanged.

Efficient computation. A quantum computer will be able to perform a sequence of operations, where the set of allowed operations is a fixed, finite set G of unitary transformations, which we will call **gates**. For concreteness, we will use so-called Hadamard, phase, CNOT, and $\pi/8$ gates, but the precise choice is not important for this article, so long as the gate set is "universal" for quantum computing.

Let Q be a quantum system on n qubits. Each gate costs unit time to apply, and each partial measurement also costs unit time. Therefore, an efficient quantum algorithm will be able to make a polynomial-length sequence of operations, where each operation is either a gate from G or a partial measurement in the computational basis. Here, "polynomial" will generally mean polynomial in n.

Quantum queries. If a quantum algorithm makes queries to some function f, there are two scenarios we will consider. In one, the oracle only accepts classical queries, in which case the algorithm must measure its query. The other case is if the oracle accepts quantum queries. Here, the oracle accepts a quantum state, and applies the unitary U_f on a quantum system $Q = Q_{in} \otimes Q_{out}$, where $U_f|x, y\rangle = |x, y \oplus f(x)\rangle$. We will denote a quantum algorithm A given classical oracle access to an oracle O as A^O. If A has quantum access, we will denote this as $A^{|O\rangle}$.

2.2.1 Facts About Quantum Computation. Here, we recall some useful results for quantum computation. The following establishes that efficient quantum computation is at least as powerful as efficient classical computation.

Fact 1 ([33]). *Let C be a classical circuit of polynomial size. Then there is a unitary U_C composed of polynomially many quantum gates and acting on the system $Q = Q_{in} \otimes Q_{out} \oplus Q_{work}$, such that*

$$U_C|x, y, 0\rangle \mapsto |x, y \oplus C(x), 0\rangle.$$

Here, Q_{in} is a quantum system over the set of possible inputs, Q_{out} is a quantum system over the set of possible outputs, and Q_{work} is another quantum system that is just used for workspace, and is reset after use.

The following result is from Zhandry [43].

Theorem 2.1 ([43]). *Let A be a quantum algorithm making q quantum queries to an oracle $H \in \mathcal{Y}^{\mathcal{X}}$. If we draw H from some distribution D, then for every z, the quantity $\Pr_{H \leftarrow D}[A^{|H\rangle}() = z]$ is a linear combination of the quantities $\Pr_{H \leftarrow D}[H(x_i) = r_i \forall i \in \{1, \ldots, 2q\}]$ for all possible settings of the x_i and r_i.*

Efficiently simulating random functions. Note that a uniformly random function from $\mathcal{X}$ to $\mathcal{Y}$ will have a truth table of size $|\mathcal{Y}|^{|\mathcal{X}|}$, which will be exponentially large if $\mathcal{X}$ has exponential size. Thus, it is computationally infeasible to actually sample such a uniform random function.

Zhandry [43] uses Theorem 2.1 and k-wise independent functions to "simulate" quantum queries to random functions efficiently. A k-wise independent function is a family of functions $\mathcal{F}$ over some domain $\mathcal{X}$ and range $\mathcal{Y}$ such that the marginal distributions of any set of k outputs are distributed uniformly. In other words, for any vector of k distinct inputs $(x_1, \ldots, x_k)$ and any vector of (possibly overlapping) k outputs $(y_1, \ldots, y_k)$, we have

$$\Pr_{f \leftarrow \mathcal{F}}[f(x_1) = y_1, \ldots, f(x_k) = y_k] = \frac{1}{|\mathcal{Y}|^k}.$$

We note that k-wise independent hash functions are sometimes called "universal hash functions" or "strongly universal hash functions" (e.g., [41]), and can be efficiently constructed.

Corollary 2.2 ([43]). *For any sets $\mathcal{X}$ and $\mathcal{Y}$, we can efficiently simulate a random oracle from $\mathcal{X}$ to $\mathcal{Y}$ capable of handling q quantum queries, where q is a polynomial. More specifically, the behavior of any quantum algorithm making at most q queries to a $2q$-wise independent function is identical to its behavior when the queries are made to a random function.*

Given an efficiently sampleable distribution D over a set $\mathcal{Y}$, we can also simulate a random function drawn from $D^{\mathcal{X}}$ as follows: Let $\mathcal{Z}$ be the set of randomness used to sample from D, and let $f(r)$ be the element $y \in \mathcal{Y}$ obtained using randomness $r \in \mathcal{Z}$. Then $D^{\mathcal{X}} = f \circ \mathcal{Z}^{\mathcal{X}}$, so we first simulate a random function $O' \in \mathcal{Z}^{\mathcal{X}}$ using k-wise independent functions, and let $O(x) = f(O'(x))$.

2.3 Cryptographic Primitives

In this article, we always assume the adversary is a quantum algorithm. However, for any particular primitive, there may be multiple definitions of security, based on how the adversary is allowed to interact with the primitive. Here we define pseudorandom functions and two security notions, as well as two definitions of indistinguishability for distributions. We also give the definition for one-way functions. The definitions of pseudorandom generators and pseudorandom synthesizers appear in the relevant sections. For a more complete discussion of the definitions of cryptographic primitives, see, e.g., Katz and Lindell [31].

One-way functions. We start by defining one-way functions. The notion of a one-way function will not be used except in theorem statements, but we include it here for completeness. Our notion is identical to the classical notion, except that we require security to hold against quantum algorithms.

Definition 2.3 (One-way Function). A one-way function is a function $F : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ and $\mathcal{Y}$ are implicitly functions of a security parameter n. We require that F can be evaluated by a classical algorithm in time polynomial in n.

Definition 2.4 (One-way Function Security). A one-way function F is *secure* if no efficient quantum adversary A can invert F. Formally, for any efficient quantum adversary A, there exists a negligible function $\epsilon = \epsilon(n)$ such that

$$\Pr_{x \leftarrow \mathcal{X}}[F(\, A(F(x))\,) = F(x)] < \epsilon.$$

Pseudorandom functions. Next, we define PRFs.

Definition 2.5 (PRF). A PRF is a function $\mathsf{PRF} : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$, where $\mathcal{K}$ is the key-space, and $\mathcal{X}$ and $\mathcal{Y}$ are the domain and range. $\mathcal{K}$, $\mathcal{X}$, and $\mathcal{Y}$ are implicitly functions of the security parameter n. We require that PRF can be evaluated by a classical algorithm in time polynomial in n. We write $y = \mathsf{PRF}_k(x)$.

We now discuss two possible security notions. The first considers quantum attackers, but restricts them to querying the function on classical inputs.

Definition 2.6 (Classical-Query Security). A pseudorandom function PRF is classical-query-secure if no efficient quantum adversary A making *classical* queries can distinguish between a truly random function and the function PRF_k for a random k. That is, for every such A, there exists a negligible function $\epsilon = \epsilon(n)$ such that

$$\mathsf{Adv}_A := \left| \Pr_{k \leftarrow \mathcal{K}} \left[A^{\mathsf{PRF}_k}() = 1 \right] - \Pr_{O \leftarrow \mathcal{Y}^{\mathcal{X}}} \left[A^{O}() = 1 \right] \right| < \epsilon.$$

Here, Adv_A is called the *advantage* of A.

The second notion of security we consider, and the focus of this work, allows the quantum attacker to also make quantum queries to function. This definition appears, for example, in Boneh et al. [8].

Definition 2.7 (Quantum Security). A pseudorandom function PRF is quantum-secure if no efficient quantum adversary A making *quantum* queries can distinguish between a truly random function and the function PRF_k for a random k. That is, for every such A, there exists a negligible

function $\epsilon = \epsilon(n)$ such that

$$\left| \Pr_{k \leftarrow \mathcal{K}} \left[A^{|\mathrm{PRF}_k\rangle}() = 1 \right] - \Pr_{O \leftarrow \mathcal{Y}^{\mathcal{X}}} \left[A^{|O\rangle}() = 1 \right] \right| < \epsilon.$$

We similarly define the advantage of A for quantum security. We call such quantum-secure pseudorandom functions **Quantum Random Functions**, or **QPRFs**.

Indistinguishability of distributions. We now provide some definitions of indistinguishability for distributions. The standard notion of indistinguishability is that no efficient quantum algorithm can distinguish a sample of one distribution from a sample of the other. More formally, let $\mathcal{Y}$ be a set, which is implicitly indexed by a security parameter n. Let D_1, D_2 be two distributions over $\mathcal{Y}$, which are also indexed by n.

Definition 2.8 (Indistinguishability). D_1, D_2 are computationally (statistically, respectively) indistinguishable if no efficient (computationally unbounded, respectively) quantum algorithm A can distinguish a sample of D_1 from a sample of D_2. That is, for all such A, there is a negligible function ϵ such that

$$\left| \Pr_{y \leftarrow D_1} [A(y) = 1] - \Pr_{y \leftarrow D_2} [A(y) = 1] \right| < \epsilon.$$

We define the advantage of an algorithm A analogously as in Definition 2.6.

For our work, we will also need a new, seemingly stronger notion of security, which we call oracle-indistinguishability. The idea is that no efficient algorithm can distinguish between oracles whose outputs are distributed according to either D_1 or D_2.

Definition 2.9 (Oracle-Indistinguishability). D_1, D_2 are computationally (statistically, respectively) oracle-indistinguishable if, for all sets $\mathcal{X}$, no efficient (computationally unbounded, respectively) quantum algorithm B can distinguish $D_1^{\mathcal{X}}$ from $D_2^{\mathcal{X}}$ using a polynomial number of quantum queries. That is, for all such B and $\mathcal{X}$, there is a negligible function ϵ such that

$$\left| \Pr_{O \leftarrow D_1^{\mathcal{X}}} \left[B^{|O\rangle}() = 1 \right] - \Pr_{O \leftarrow D_2^{\mathcal{X}}} \left[B^{|O\rangle}() = 1 \right] \right| < \epsilon.$$

For this article, we will primarily be discussing computationally bounded adversaries, so we will normally take indistinguishability and oracle-indistinguishability to mean the computational versions.

3 SEPARATION RESULT

In this section, we show our separation result.

Theorem 3.1. *If one-way functions exist, then there are classical-query-secure PRFs that are not QPRFs.*

Proof. Our idea is to construct a PRF that embeds an oracle problem which is hard for classical query algorithms (even if the algorithm may use a quantum computer between queries), but is easy for quantum query algorithms. Our choice is period finding, which was used by Shor [39] to give efficient quantum algorithms for factoring integers and finding discrete logarithms. Concretely, we will construct a new pseudorandom function that is periodic with some large, secret period a, meaning $\mathrm{PRF}(x) = \mathrm{PRF}(x + a)$. Classical adversaries will not be able to detect the period, and thus cannot distinguish this new function from random. However, an adversary making quantum queries can detect the period, and thus distinguish our new function from random.

In more detail, let PRF be a classical-query-secure pseudorandom function with key-space $\mathcal{K}$, domain $\mathcal{X}$, and co-domain $\mathcal{Y}$. We can construct PRF from any one-way function secure against quantum queries, following Theorem 1.1 and Theorem 1.2. Interpret $\mathcal{X}$ as $[N]$, where N is the number of elements in $\mathcal{X}$. Assume without loss of generality that $\mathcal{Y}$ contains at least N^2 elements; if not, we can construct a new pseudorandom function with smaller domain but larger range in a standard way. We now construct a new pseudorandom function $\mathsf{PRF}'_{(k,a)}(x) = \mathsf{PRF}_k(x \bmod a)$ where

- The key space of PRF′ is $\mathcal{K}' = \mathcal{K} \times \mathcal{A}$ where $\mathcal{A}$ is the set of primes between $N/2$ and N. That is, a key for PRF′ is a pair (k, a) where k is a key for PRF, and a is a prime in the range $(N/2, N]$.
- The domain is $\mathcal{X}' = [N']$ where N' is the smallest power of 2 greater than $4N^2$.

The following two claims together prove Theorem 3.1:

Claim 1. *If* PRF *is classical-query-secure, then so is* PRF′.

Claim 2. *If* PRF *is quantum-secure, then* PRF′ *is not.*

Thus, one of PRF and PRF′ is classical-query-secure but not quantum-secure, as desired. We now prove Claims 1 and 2.

Proof of Claim 1. We prove that if PRF is classical-query-secure, so is PRF′. The idea is that, since PRF is a classical-query-secure pseudorandom function, we can replace it with a truly random function in the definition of PRF′, and no efficient adversary making classical queries will notice. But we are then left with a function that has a large random period where every value in the period is chosen independently at random. This function will look truly random unless the adversary happens to query two points that differ by a multiple of the period. But by the birthday bound, this will only happen with negligible probability.

We now give the proof. Suppose we have a quantum adversary A making classical queries that distinguishes PRF′ from a random function with non-negligible advantage ϵ. That is,

$$\left| \Pr_{k \leftarrow \mathcal{K}, a \leftarrow \mathcal{A}} \left[A^{\mathsf{PRF}'_{(k,a)}}() = 1 \right] - \Pr_{O \leftarrow \mathcal{Y}^{\mathcal{X}}} \left[A^{O}() = 1 \right] \right| = \epsilon.$$

This is equivalent to

$$\left| \Pr_{k \leftarrow \mathcal{K}, a \leftarrow \mathcal{A}} \left[A^{\mathsf{PRF}_k(\cdot \bmod a)}() = 1 \right] - \Pr_{O \leftarrow \mathcal{Y}^{\mathcal{X}}} \left[A^{O}() = 1 \right] \right| = \epsilon.$$

Now consider the quantity

$$\left| \Pr_{O \leftarrow \mathcal{Y}^{\mathcal{X}}, a \leftarrow \mathcal{A}} \left[A^{O(\cdot \bmod a)}() = 1 \right] - \Pr_{O \leftarrow \mathcal{Y}^{\mathcal{X}}} \left[A^{O}() = 1 \right] \right|.$$

The left-hand side is the case where O is a random function in $\mathcal{Y}^{\mathcal{X}}$, a is a random prime in $(N/2, N]$, and we give A the oracle $O'(x) = O(x \bmod a)$. As long as A never queries its oracle on two points x and x' such that $x \equiv x' \bmod a$, the outputs of each query will be independently sampled; hence, the oracle would be perfectly indistinguishable from a random function in this case. Thus, we just have to show that A is unlikely to query on x, x' such that $x - x'$ is a multiple of a. Toward that end, if A makes q queries, there are $\binom{q}{2}$ possible differences between query points. Each difference is at most $N' < 8N^2$, and each prime moduli a is greater than $N/2$. For $N > 64$, $(N/2)^3 > 32(N/2)^2 = 8N^2 > N'$, meaning the difference can only be divisible by at most two different prime moduli. By the prime number theorem, $|\mathcal{A}| \in \Omega(N/\log N)$. Each difference thus

has a probability at most $2/|\mathcal{A}| \in O(2 \log N/N)$ of being divisible by a, so the total probability of querying x and x' such that $x \equiv x' \bmod a$ is at most $O(q^2 \log N/N)$. Thus, this probability, and hence the advantage of A in distinguishing O' from a random oracle, is negligible. A simple hybrid argument then shows that

$$\left| \Pr_{k \leftarrow \mathcal{K}, a \leftarrow \mathcal{A}} \left[A^{\mathsf{PRF}_k(\cdot \bmod a)}() = 1 \right] - \Pr_{O \leftarrow \mathcal{Y}^{\mathcal{X}}, a \leftarrow \mathcal{A}} \left[A^{O(\cdot \bmod a)}() = 1 \right] \right| \geq \epsilon - O(q^2 \log N/N). \quad (3.1)$$

We now define a quantum algorithm B which distinguishes PRF from a random oracle. B makes classical queries to an oracle O. It chooses a random prime $a \in (N/2, N]$, and simulates A with the oracle $O'(x) = O(x \bmod a)$ by making queries to O. When $O = \mathsf{PRF}_k$, we get the left side of Equation (3.1), and when O is random, we get the right side. Thus,

$$\left| \Pr_{k \leftarrow \mathcal{K}} \left[B^{\mathsf{PRF}_k}() = 1 \right] - \Pr_{O \leftarrow \mathcal{Y}^{\mathcal{X}}} \left[B^{O}() = 1 \right] \right| \geq \epsilon - O(q^2 \log N/N).$$

Since N is exponential, B breaks the classical-query security of PRF. □

Proof of Claim 2. We now show that PRF and PRF′ cannot both be quantum-secure. The idea is that, if we allow quantum queries to PRF′, we can use the period finding algorithm of Boneh and Lipton [11] to find a. With a, it is easy to distinguish PRF′ from a random oracle by making two additional classical queries that differ by a multiple of the period. In order to prove that the period finding algorithm works, we need certain structural properties of PRF′. Fortunately, these properties are satisfied if PRF is quantum-secure.

We now give the proof. Suppose PRF is quantum secure. We first consider the case where PRF′ is built from a truly random function $O : [N] \to \mathcal{Y}$. That is, $\mathsf{PRF}'_a(x) = O(x \bmod a)$.

The following is a special case of the main theorem of Boneh and Lipton [11], which suffices for our purposes.

Theorem 3.2 (Special case of [11], Theorem 1). *There is a polynomial time quantum algorithm making quantum queries to an oracle $f : [N'] \to \mathcal{Y}$ such that the following holds. If f satisfies the two conditions*

- *there is a prime a such that $f(x) = f(x \bmod a)$ for all $x \in [N']$, and*
- *for each $y \in \mathcal{Y}$, there is at most a single $x \in [a]$ such that $f(x) = y$,*

then the algorithm outputs a with overwhelming probability.

Since $|\mathcal{Y}| \geq N^2$, the probability that there is any collision $x, x' \in [N], x \neq x'$ where $O(x) = O(x')$ is less than $1/2$. Thus, with probability at least $1/2$, $f(x) = O(x \bmod a)$ satisfies the requirements of Theorem 3.2. Thus, we get a distinguisher that works as follows, given access to an oracle O':

- Use Theorem 3.2 to find a.
- If $a \in (N/2, N]$, pick a random $x \in [N' - a]$, and verify that $O'(x) = O'(x + a)$. If so, output 1. Otherwise, output 0.

If $O' = O(x \bmod a)$, then with probability at least $1/2$, O will have no collisions, meaning we will find a with probability $1 - o(1)$. In this case, $O'(x) = O'(x + a)$ will always be true, so we output 1. If O' is random, then for any x, the probability that there is any $x' \in x + (N/2, N]$ with $O'(x') = O'(x)$ is negligible, so the random oracle will fail the test with all but negligible probability. Therefore, we distinguish PRF′ from random with advantage at least $1/2 - o(1)$.

We now switch to the true definition of PRF′. That is, we replace the random oracle O with PRF. Since PRF is quantum-secure, this only affects the behavior of our distinguisher negligibly. Therefore, our distinguisher still distinguishes PRF′ from random with advantage at least $1/2 - o(1)$. □

This completes the proof of Theorem 3.1. □

We have shown that for pseudorandom functions, security against classical queries does not imply security against quantum queries. In the next sections, we will show, however, that several of the standard constructions in the literature are nevertheless quantum-secure.

4 DISTINGUISHING ORACLE DISTRIBUTIONS

The goal of this section is give a proof for Theorem 4.5, which will be re-stated in Section 4.1. In order to prove Theorem 4.5, we first introduce a new tool, called *small-range distributions*, which allow for simulating a large oracle using very few samples from the underlying distribution of outputs. We then develop new quantum oracle techniques to prove that the simulated oracle is indistinguishable from the large oracle, showing that the simulation is effective.

Let $\mathcal{X}$ and $\mathcal{Y}$ be sets. Given a distribution D on $\mathcal{Y}$, define the small range distribution $\mathrm{SR}_r^D(\mathcal{X})$ as the following distribution on functions from $\mathcal{X}$ to $\mathcal{Y}$:

- For each $i \in [r]$, choose a random value $y_i \in \mathcal{Y}$ according to the distribution D.
- For each $x \in \mathcal{X}$, pick a random $i \in [r]$ and set $O(x) = y_i$.

We will often omit the domain $\mathcal{X}$ when it is clear from context. An alternate view of this distribution is to choose $g \leftarrow D^{[r]}$ and $f \leftarrow [r]^{\mathcal{X}}$, and output the composition $g \circ f$. That is, $\mathrm{SR}_r^D(\mathcal{X}) = D^{[r]} \circ [r]^{\mathcal{X}}$. In other words, we choose a random function f from $\mathcal{X}$ to $[r]$, and compose it with another random function g from $[r]$ to $\mathcal{Y}$, where outputs are distributed according to D. We call this distribution a small-range distribution because the set of images of any function drawn from the distribution is bounded to at most r points, which for $r << \mathcal{Y}$ will be a small subset of the co-domain. Notice that, as r goes to infinity, f will be injective with probability 1, and hence for each x, $g(f(x))$ will be distributed independently according to D. That is, $\mathrm{SR}_\infty^D(\mathcal{X}) = D^{\mathcal{X}}$.

We wish to show that $\mathrm{SR}_r^D(\mathcal{X})$ for "small" r is indistinguishable from the corresponding "large-range" function $D^{\mathcal{X}}$ where every output is sampled independently according to D. This will allow us to simulate the large-range $D^{\mathcal{X}}$ using relatively few samples from D.

To motivate our approach, we start by recalling a theorem of Zhandry [43]:

Theorem 4.1. *Fix q, and let D_λ be a family of distributions on $\mathcal{Y}^{\mathcal{X}}$ indexed by $\lambda \in [0,1]$. Suppose there is an integer d such that for every $2q$ pairs $(x_i, r_i) \in \mathcal{X} \times \mathcal{Y}$, the function $p(\lambda) = \Pr_{H \leftarrow D_\lambda}[H(x_i) = r_i \forall i \in \{1, \ldots, 2q\}]$ is a polynomial of degree at most d in λ. Then for any quantum algorithm A making q quantum queries, the output distributions under D_λ and D_0 are $2\lambda d^2$-close.*

We might be tempted to try to apply Theorem 4.1 to SR_r^D. Basically, we would set $\lambda = 1/r$ (since $r \geq 1$, $\lambda \in [0,1]$), and define $D_\lambda = \mathrm{SR}_r^D(\mathcal{X})$. Then, we would hope that the $p(\lambda)$ are polynomials of at most d in λ for some "small" d. We will show that this is true (namely, $d = q$), but unfortunately we still cannot apply Theorem 4.1. This is because the distributions D_λ are only well-defined for λ that are the reciprocal of positive integers, whereas Theorem 4.1 requires D_λ to be well-defined for all *real* $\lambda \in [0,1]$. Importantly, this means that the $p(\lambda)$ do not define probabilities for all λ (in other words, $p(\lambda) \notin [0,1]$ for some λ), a fact which the proof of Theorem 4.1 crucially relied on. Instead, we show a similar result that handles exactly the case where $\lambda = 1/r$.

Theorem 4.2. *Fix q, and let E_r be a family of distributions on $\mathcal{Y}^{\mathcal{X}}$ indexed by $r \in \mathbb{Z}^+ \bigcup \{\infty\}$. Suppose there is an integer d such that for every $2q$ pairs $(x_i, r_i) \in \mathcal{X} \times \mathcal{Y}$, the function*

$p(\lambda) = \Pr_{H \leftarrow E_{1/\lambda}}[H(x_i) = r_i \forall i \in \{1, \ldots, 2q\}]$ is a polynomial of degree at most d in λ. Then for any quantum algorithm A making q quantum queries, the output distributions under E_r and E_∞ are $\pi^2 d^3/3r$-close.

Let $D_\lambda = E_{1/\lambda}$. We see that the conditions of Theorems 4.2 and 4.1 are identical, with the following exception: Theorem 4.1 requires D_λ to be a distribution for all $\lambda \in [0, 1]$, while Theorem 4.2 only requires D_λ to be a distribution when $1/\lambda$ is an integer (and when $\lambda = 0$), exactly the case we need. The proof is thus similar in flavor to that of Theorem 4.1, with the following exception: the proof of Theorem 4.1 uses well-known bounds on polynomials p where $p(\lambda) \in [0, 1]$ for all $\lambda \in [0, 1]$. However, our polynomials p are only guaranteed to be in $[0, 1]$ when λ is the reciprocal of an integer. Thus, we need similar bounds on polynomials p where only $p(1/r)$ is required to be in $[0, 1]$ for integer r. Such polynomials are far less understood, and we need to prove suitable bounds under these relaxed assumptions. The proof is somewhat involved, and appears in Section 8.

Now we will apply Theorem 4.2 to our small-range distributions. We first need the following lemma:

Lemma 4.3. *Fix k. For any $\mathcal{X}$, the probabilities in each of the marginal distributions of $\mathrm{SR}_r^D(\mathcal{X})$ over k inputs are polynomials in $1/r$ of degree k.*

This lemma will be proved shortly. We can then use Theorem 4.2 to bound the ability of any quantum algorithm to distinguish $\mathrm{SR}_r^D(\mathcal{X})$ from $\mathrm{SR}_\infty^D(\mathcal{X}) = D^{\mathcal{X}}$.

Corollary 4.4. *The output distributions of a quantum algorithm making q quantum queries to an oracle either drawn from $\mathrm{SR}_r^D(\mathcal{X})$ or $D^{\mathcal{X}}$ are $\ell(q)/r$-close, where $\ell(q) = \pi^2(2q)^3/3 < 27q^3$.*

We observe that this bound is asymptotically tight: in Section 9, we show that the quantum collision finding algorithm of Brassard et al. [9] can be used to distinguish $\mathrm{SR}_r^D(\mathcal{X})$ from $D^{\mathcal{X}}$ with asymptotically optimal advantage. This also shows that Theorem 4.2 is tight. In particular, it demonstrates that the weaker bound obtained by relaxing the requirement $0 \le p(\lambda) \le 1$ to hold only over reciprocals of integers is inherent and not a fluke of the proof.

Proof of Lemma 4.3. Our goal is to show that, for each of the marginal distributions over k inputs to SR_r^D, each probability is a polynomial in $1/r$ of degree at most k.

The intuition is as follows. Fix some x_1, x_2. Let $O = O_2 \circ O_1$ where $O_1 \leftarrow [r]^{\mathcal{X}}$ and $O_2 \leftarrow D^{[r]}$. Notice that if $O_1(x_1) = O_1(x_2)$, then $O(x_1) = O(x_2)$, and this value is sampled randomly from D. On the other hand, if $O_1(x_2) \neq O_1(x_2)$, then $O(x_1)$ and $O(x_2)$ are independent samples from D. Let $D_=$ be the distribution over $\mathcal{Y}^2$ which samples $y \leftarrow D$ and outputs (y, y), and let $D_\neq$ be the distribution D^2 which outputs two independent samples from D. We then have that the marginal distribution of $(O(x_1), O(x_2))$ is the mixture of the distributions $D_=$ and $D_\neq$, given by

$$\Pr[O_1(x_1) = O_1(x_2)](D_=) + (1 - \Pr[O_1(x_1) = O_1(x_2)])(D_\neq).$$

The only dependence on r is through $\Pr[O_1(x_1) = O_1(x_2)]$, which is exactly $1/r$. Hence, all the marginal probabilities are degree $1 \le k = 2$ polynomials in $1/r$.

For higher k, we can similarly describe the marginal distributions $(O(x_1), \ldots, O(x_k))$ as a mixture of distributions, where the coefficients of the mixture are linear functions in the probabilities of various "equality patterns" amongst the $O_1(x_i)$, such as $O_1(x_1) = O_1(x_3) \wedge O_1(x_2) = O_1(x_4)$. The probabilities of these equality patterns all have the form $1/r^{k'}$ for $k' \le k$. This then gives the desired outcome.

We now give the formal proof. Fix some x_i and y_i for $i \in [k]$. We consider the probability that $O(x_i) = y_i$ for all $i \in [k]$. In the case $k = 1$, this probability is always a constant, as $O(x_i)$ is distributed exactly as D. We will therefore assume $k \ge 2$. We can assume without loss of generality that the x_i are distinct. Indeed, suppose instead that there are $i \neq j$ such that $x_i = x_j$. If $y_i \neq y_j$,

then the probability is 0 (O is not a function in this case), in particular a constant. If $y_i = y_j$, the $O(x_j) = y_j$ condition is redundant and can be removed, reducing this to the $k-1$ case. By induction on k, the resulting probability is a polynomial of degree at most $k - 1 < k$.

Recall that $\mathsf{SR}_r^D = D^{[r]} \circ [r]^{\mathcal{X}}$ and D is a distribution on $\mathcal{Y}$. Let $O_1 \leftarrow [r]^{\mathcal{X}}$ and $O_2 \leftarrow D^{[r]}$. Let O_1' be the restriction of O_1 to $\{x_0, \ldots, x_{k-1}\}$. Each O_1' then occurs with probability $1/r^k$. Now,

$$\begin{aligned}\Pr_{O\leftarrow \mathsf{SR}_r^D}[O(x_i) = y_i \forall i \in [k]] &= \Pr_{O_1\leftarrow [r]^{\mathcal{X}}, O_2 \leftarrow D^{[r]}}[O_2(O_1(x_i)) = y_i \forall i \in [k]] \\ &= \Pr_{O_1'\leftarrow [r]^{\{x_0,\ldots,x_{k-1}\}}, O_2 \leftarrow D^{[r]}}[O_2(O_1'(x_i)) = y_i \forall i \in [k]] \\ &= \frac{1}{r^k}\sum_{O_1'} \Pr_{O_2\leftarrow D^{[r]}}[O_2(O_1'(x_i)) = y_i \forall i \in [k]].\end{aligned}$$

We now associate with each O_1' a partition P of $[k]$ into $k' \le k$ non-empty subsets $P_1, \ldots, P_{k'}$ and a *strictly increasing* function $f : [k'] \to [r]$. The association is as follows: $O_1'(x_k) = f(j)$ where $x_k \in P_j$. In other words, f is defined as the unique strictly increasing function from $[k']$ that has the same range as O_1', and the sets P_j are the pre-images of $f(j)$. Thus, this association is a bijection. Using this association, we can write

$$\Pr_{O\leftarrow \mathsf{SR}_r^D}[O(x_i) = y_i \forall i \in [k]] = \frac{1}{r^k}\sum_{P=(P_j)}\sum_f \Pr[O_2(f(j)) = y_i \forall j, i \in P_j].$$

We now use the fact that O_2 is sampled independently for each distinct $f(j)$ (and hence, distinct j by the injectivity of f) to write

$$\Pr_{O\leftarrow \mathsf{SR}_r^D}[O(x_i) = y_i \forall i \in [k]] = \frac{1}{r^k}\sum_{P=(P_j)}\sum_f \prod_{j\in[k']} \Pr[O_2(f(j)) = y_i \forall i \in P_j].$$

Suppose for some j that the $y_i, i \in P_j$ are not all identical. Then $\Pr[O_2(f(j)) = y_i \forall i \in P_j] = 0$ since O_2 must be a function. On the other hand, if all the $y_i, i \in P_j$ are identical, then $\Pr[O_2(f(j)) = y_i \forall i \in P_j] = D(y_i)$ for an arbitrary choice of $i \in P_j$. Thus, we can write $\Pr[O_2(f(j)) = y_i \forall i \in P_j] = D(y_i)\sigma(P_j)$ where $\sigma(S)$ is 1 if y_i are all equal for $i \in S$, and 0 otherwise. Thus,

$$\Pr_{O\leftarrow \mathsf{SR}_r^D}[O(x_i) = y_i \forall i \in [k]] = \frac{1}{r^k}\sum_{P=(P_j)}\sum_f \prod_{j\in[k']} D(y_i)\sigma(P_j).$$

The summand does not depend on f. The number of f is just the number of ways of picking k' elements from $[r]$, or $\binom{r}{k'}$. Then we can write

$$\Pr_{O\leftarrow \mathsf{SR}_r^D}[O(x_i) = y_i \forall i \in [k]] = \frac{1}{r^k}\sum_{P=(P_j)}\binom{r}{k'}\prod_{j\in[k']} D(y_i)\sigma(P_j).$$

The P we are summing over are independent of r, as is the product in the above expression. $\binom{r}{k'}$ is a polynomial in r of degree at most $k' \le k$. Therefore, performing the sum, $\Pr_{O\leftarrow \mathsf{SR}_r^D}[O(x_i) = y_i \forall i \in [k]]$ is a polynomial of degree at most k in r, divided by r^k. The result is a polynomial of degree at most k in $1/r$. □

4.1 The Equivalence of Indistinguishability and Oracle-Indistinguishability

We now use the above techniques to explore the relationship between indistinguishability and oracle-indistinguishability and to prove Theorem 4.5, which is re-stated here using our notions of indistinguishability for distributions (Definitions 2.8 and 2.9).

Theorem 4.5. *Let D_1 and D_2 be efficiently sampleable distributions over a set $\mathcal{Y}$. Then D_1 and D_2 are indistinguishable if and only if they are also oracle-indistinguishable.*

Clearly, oracle-indistinguishability implies standard indistinguishability: if A distinguishes D_1 from D_2, then the algorithm $B^{|O\rangle}()$ that picks any $x \in \mathcal{X}$ and returns $A(O(x))$ breaks the oracle-indistinguishability.

In the other direction, in the classical world, if B makes q queries to O, we can simulate O "on the fly" using q samples from either D_1 or D_2. Then we can show that q samples of D_1 are indistinguishable from q samples of D_2 using a simple hybrid argument across the samples. The result is that if B distinguishes $D_1^{\mathcal{X}}$ from $D_2^{\mathcal{X}}$ with advantage ϵ, then we obtain an adversary A distinguishing D_1 from D_2 with advantage ϵ/q, breaking standard indistinguishability. In the quantum world, however, each query might be over a superposition of all exponentially many inputs in $\mathcal{X}$. Therefore, there will be exponentially many hybrids, resulting in B only distinguishing with advantage $\epsilon/|\mathcal{X}|$. This exponentially small advantage is negligible and therefore unable to give a contradiction.

In the statistical setting, this question has been addressed by Boneh et al. [8]. They show using information-theoretic techniques that if a (potentially computationally unbounded) quantum adversary making q queries distinguishes $D_1^{\mathcal{X}}$ from $D_2^{\mathcal{X}}$ with advantage ϵ, then D_1 and D_2 must be $\Omega(\epsilon^2/q^4)$-far. Their proof is inherently statistical, and even if one starts from a computationally bounded distinguisher for $D_1^{\mathcal{X}}$ from $D_2^{\mathcal{X}}$, the resulting distinguisher for D_1 and D_2 will be computationally *unbounded*. With our small-range distributions in hand, we now have the tools to extend their result to the computational setting (and improve quantitative bound for the statistical setting in the process) by proving Theorem 4.5.

Proof of Theorem 4.5. Let B be an (efficient) quantum adversary that distinguishes $D_1^{\mathcal{X}}$ from $D_2^{\mathcal{X}}$ with non-negligible advantage ϵ, for distributions D_1 and D_2 over $\mathcal{Y}$. That is, there is some set $\mathcal{X}$ such that

$$\left| \Pr_{O \leftarrow D_1^{\mathcal{X}}} \left[B^{|O\rangle}() = 1 \right] - \Pr_{O \leftarrow D_2^{\mathcal{X}}} \left[B^{|O\rangle}() = 1 \right] \right| = \epsilon.$$

Our goal is to construct an (efficient) quantum algorithm A that distinguishes a sample of D_1 from a sample of D_2. At a high level, we first replace $D_b^{\mathcal{X}}$ with $\mathsf{SR}_r^{D_b}$, for an appropriate choice of r. Since $\mathsf{SR}_r^{D_b}$ is indistinguishable from $D_b^{\mathcal{X}}$, but $D_0^{\mathcal{X}}, D_1^{\mathcal{X}}$ are distinguished by B, we must have that B still distinguishes $\mathsf{SR}_r^{D_1}$ from $\mathsf{SR}_r^{D_2}$. But now, $\mathsf{SR}_r^{D_b}$ can be simulated using only r samples from D_b, and as such we can get a distinguisher between D_1^r and D_2^r. Then, we perform the standard classical hybrid across the r samples to get a distinguisher for D_1 and D_2.

The one wrinkle in the above is the choice of r. If r is a polynomial, then $\mathsf{SR}_r^{D_b}$ can be distinguished from $D_b^{\mathcal{X}}$ with inverse polynomial advantage (see Section 9). But a fixed inverse polynomial indistinguishability is too large: for ϵ smaller than the indistinguishability bound, we would no longer be able to claim that $\mathsf{SR}_r^{D_1}$ and $\mathsf{SR}_r^{D_2}$ are still distinguished by B. On the other hand, if r is super-polynomial, then the final hybrid incurs a super-polynomial loss, and hence we fail to get an inverse polynomial advantage for distinguishing D_1 and D_2.

The solution is to choose r *dependent* on ϵ. Namely, we choose r sufficiently larger than $1/\epsilon$ such that we can still argue $\mathsf{SR}_r^{D_1}$ and $\mathsf{SR}_r^{D_2}$ are distinguished by B, but small enough to only incur a polynomial loss in the final hybrid.

We now give the proof. Choose r so that $\ell(q)/r = \epsilon/4$, where $\ell(q)$ is the polynomial from Corollary 4.4. That is, $r = 4\ell(q)/\epsilon$. No quantum algorithm can distinguish $\mathsf{SR}_r^{D_i}(\mathcal{X})$ from $D_i^{\mathcal{X}}$ with

advantage greater than $\ell(q)/r = \epsilon/4$. Thus, it must be that the quantity

$$\left| \Pr_{O \leftarrow \mathrm{SR}_r^{D_1}(X)} \left[B^{|O\rangle}() = 1 \right] - \Pr_{O \leftarrow \mathrm{SR}_r^{D_2}(X)} \left[B^{|O\rangle}() = 1 \right] \right|$$

is at least $\epsilon/2$. We now define $r+1$ hybrids H_i as follows: For $j = 0, \ldots, i-1$, draw y_j from D_1. For $j = i, \ldots, r-1$, draw y_j from D_2. Then give B the oracle O where for each x, $O(x)$ is a randomly selected y_j. H_r is the case where $O \leftarrow \mathrm{SR}_r^{D_1}$, and H_0 is the case where $O \leftarrow \mathrm{SR}_r^{D_2}$. Hence, H_0 and H_r are distinguished with advantage at least $\epsilon/2$. Let

$$\epsilon_i = \Pr_{O \leftarrow H_{i+1}} \left[B^{|O\rangle}() = 1 \right] - \Pr_{O \leftarrow H_i} \left[B^{|O\rangle}() = 1 \right]$$

be the (signed) advantage with which B distinguishes H_{i+1} from H_i. Then $|\sum_{i=1}^{r} \epsilon_i| \geq \epsilon/2$.

We construct an algorithm A that distinguishes between D_1 and D_2 with advantage $\epsilon/2r$. A essentially chooses a random i, and uses its input to simulate either H_i or H_{i+1}, depending on whether y comes from D_2 or D_1. In more detail, A, on input y, does the following:

- Choose a random $i \in [r]$.
- Construct a random oracle $O_0 \leftarrow [r]^X$.
- Construct random oracles $O_1 \leftarrow D_1^{\{0,\ldots,i-1\}}$ and $O_2 \leftarrow D_2^{\{i+1,\ldots,r-1\}}$.
- Construct the oracle O where $O(x)$ is defined as follows:
  — Compute $j = O_0(x)$.
  — If $j = i$, output y.
  — Otherwise, if $j < i$, output $O_1(j)$ and if $j > i$, output $O_2(j)$.
- Simulate B with the oracle O, and output the output of B.

Let A_i be the algorithm A using a given i. If $y \leftarrow D_1$, B sees hybrid H_{i+1}. If $y \leftarrow D_2$, B sees H_i. Therefore, we have that

$$\Pr_{y \leftarrow D_1}[A_i(y) = 1] - \Pr_{y \leftarrow D_2}[A_i(y) = 1] = \epsilon_i.$$

Averaging over all i, we get that A's distinguishing advantage is

$$|\Pr_{y \leftarrow D_1}[A(y) = 1] - \Pr_{y \leftarrow D_2}[A(y) = 1]| = \left| \frac{1}{r} \sum_{i=1}^{r} \epsilon_i \right| \geq \frac{\epsilon}{2r} = \frac{\epsilon^2}{8\ell(q)}.$$

Thus, A is a quantum algorithm that distinguishes D_1 from D_2 with non-negligible advantage. The only remaining piece is the efficiency of A. Unfortunately, A as described is *inefficient*, since the oracles O_0, O_1, O_2 are exponentially large to even write down. We solve this by using Corollary 2.2, which states that such oracles can be simulated given a polynomial upper bound on the number of quantum queries to the oracles. A makes a constant number of queries to O_0, O_1, O_2 for each query made by B, which in turn makes a polynomial number of queries. Thus, we apply Corollary 2.2 to get an efficient A with the same distinguishing advantage, breaking the indistinguishability of D_1 and D_2. □

Notice that by removing the requirement that B be an efficient algorithm, we get a proof for the statistical setting as well, so that if any computationally unbounded quantum algorithm making q quantum queries distinguishes D_1^X from D_2^X with advantage ϵ, then D_1 and D_2 must be $\Omega(\epsilon^2/\ell(q)) = \Omega(\epsilon^2/q^3)$-far, improving the result of Boneh et al. by a factor of q.

5 PSEUDORANDOM FUNCTIONS FROM PSEUDORANDOM GENERATORS

We recall the construction of pseudorandom functions from pseudorandom generators due to Goldreich et al. [22], the so-called GGM construction. We also prove its security in a new way that makes sense in the quantum setting, giving the first proof of quantum security for the construction. First, we define pseudorandom generators.

Definition 5.1 (PRG). A (length-doubling) PRG is a function $G : \mathcal{K} \to \mathcal{Y}$, where $\mathcal{Y} = \mathcal{K} \times \mathcal{K}$. $\mathcal{K}$ is implicitly indexed by the security parameter n. We require that G can be evaluated by a classical algorithm in time polynomial in n.

Definition 5.2 (Standard Security). A pseudorandom generator G is secure against quantum computers if the distributions $G \circ \mathcal{K}$ and $\mathcal{Y}$ are computationally indistinguishable. In this case, we say that G is standard-secure.

We note that other sets $\mathcal{Y}$ are possible for a pseudorandom generator, corresponding to different levels of "stretch." For the purposes of this work, however, we only need to discuss the length-doubling case. We also note that standard security implies that $|\mathcal{K}|$ is super-polynomial in n, since it is always possible to distinguish $G \circ \mathcal{K}$ from $\mathcal{Y}$ in time roughly $|\mathcal{K}|$ by trying all possible inputs to G.

We now give the GGM construction, which builds a PRF from any length-doubling PRG. The starting idea is that a length-doubling PRG is already a PRF for single-bit inputs. Namely, if we write $G(x) = (G_0(x), G_1(x))$, then the function mapping $(k, b) \mapsto G_b(x)$ is a PRF. Of course, single-bit inputs are too small, as we would like exponentially many inputs. The idea behind the GGM construction is to recursively derive k as a smaller PRF with input set $[2]^{n-1}$. Then, expand the input to $[2]^n$ by deriving k from the first $n-1$ bits of input using the smaller PRF and setting b to be the last bit.

Construction 1 (GGM-PRF). *Let $G : \mathcal{K} \to \mathcal{K}^2$ be a length-doubling pseudorandom generator. Write $G(x) = (G_0(x), G_1(x))$ where G_0, G_1 are functions from $\mathcal{K}$ to $\mathcal{K}$. Then, we define the GGM pseudorandom function* $\mathsf{PRF} : \mathcal{K} \times [2]^n \to \mathcal{K}$ *where*

$$\mathsf{PRF}_k(x) = G_{x_1}(\ldots G_{x_{n-1}}(G_{x_n}(k))\ldots).$$

That is, the function PRF takes a key k in $\mathcal{K}$ and an n-bit input string. It first applies G to k. It keeps the left or right half of the output depending on whether the last bit of the input is 0 or 1. What remains is an element in $\mathcal{K}$, so the function applies G again, keeps the left or right half depending on the second-to-last bit, and so on.

As described in the Introduction, the standard proof of security fails to prove quantum security. Using Theorem 4.5, we show how to work around this problem. As a first step, we define a stronger notion of security for pseudorandom generators, which we call oracle security. Justifying oracle security for a PRG will capture the main challenging step in proving quantum security of the GGM construction; fortunately, this step is implied by Theorem 4.5.

Definition 5.3 (Oracle Security). A pseudorandom generator $G : \mathcal{X} \to \mathcal{Y}$ is oracle-secure if the distributions $G \circ \mathcal{K}$ and $\mathcal{Y}$ are oracle-indistinguishable.

$G \circ \mathcal{K}$ is efficiently sampleable since we can sample a random value in $\mathcal{K}$ and apply G to it. Then, $G \circ \mathcal{K}$ and $\mathcal{Y}$ are both efficiently sampleable, so Theorem 4.5 immediately gives

Corollary 5.4. *If G is a secure PRG, then it is also oracle-secure.*

We now can prove Theorem 5.5, showing the quantum security of Construction 1 following a similar approach to the classical proof, replacing a key step with Corollary 5.4. We briefly recall Theorem 5.5 before proving it.

Theorem 5.5. *If G is a length-doubling standard-secure PRG, then* PRF *from Construction 1 is a QPRF.*

Proof. We adapt the security proof of Goldreich et al. to convert any adversary for PRF into an adversary for the oracle security of G. Then, Corollary 5.4 shows that this adversary is impossible under the assumption that G is standard-secure.

In slightly more detail, recall that the Goldreich et al. proof consists of two steps: a hybrid argument over the levels of the GGM tree, and then a hybrid argument within a single level. This second hybrid is what causes trouble in the quantum-security setting. Our new proof keeps the first hybrid over the levels of the tree, which works much the same way as the classical proof. Then, we observe that the second hybrid is exactly establishing the oracle security of the underlying PRG. We therefore replace this step with Corollary 5.4 to complete the proof.

We now give the details. Suppose a quantum adversary A distinguishes PRF from a random oracle with probability ϵ. Define hybrids H_i as follows: Pick a random function $P \leftarrow \mathcal{K}^{[2]^{n-i}}$ (i.e., random function from $(n-i)$-bit strings into $\mathcal{K}$) and give A the oracle

$$O_i(x) = G_{x_1}(\dots G_{x_i}(P(x_{[i+1,n]}))\dots).$$

H_0 is the case where A's oracle is random. When $i = n$, $P \leftarrow \mathcal{K}^{[2]^{n-i}}$ is a random function from the set containing only the empty string to $\mathcal{K}$, and hence is associated with the image of the empty string, a single random element in $\mathcal{K}$. Thus, H_n is the case where A's oracle is PRF. Let ϵ_i be the (signed) advantage A has in distinguishing H_i from H_{i+1}. That is,

$$\epsilon_i = \Pr\left[A^{|O_i\rangle}() = 1\right] - \Pr\left[A^{|O_{i+1}\rangle}() = 1\right].$$

Notice that there is no absolute value in the definition of ϵ_i. It follows that $|\sum_i \epsilon_i| = \epsilon$. We now construct a quantum algorithm B breaking the oracle security of G. B is given quantum access to an oracle $P : [2]^{n-1} \to \mathcal{K}^2$, and distinguishes $P \leftarrow (\mathcal{K}^2)^{[2]^{n-1}}$ from $P \leftarrow G \circ \mathcal{K}^{[2]^{n-1}}$. That is, B is given either a random function from $(n-1)$-bit strings into $\mathcal{K}^2$, or G applied to a random function from $(n-1)$-bit strings into $\mathcal{K}$. B distinguishes the two cases as follows:

- Pick a random i in $\{0, \dots, n-1\}$.
- Let $P^{(i)} : [2]^{n-i-1} \to \mathcal{K}^2$ be the oracle $P^{(i)}(x) = P(0^i x)$. Thus, $P^{(i)}$ is either a random function from $(n-1-i)$-bit strings into $\mathcal{K}^2$, or G applied to a random function.
- Write $P^{(i)}$ as $(P_0^{(i)}, P_1^{(i)})$ where $P_b^{(i)} : [2]^{n-i-1} \to \mathcal{K}$ are the left and right halves of the output of $P^{(i)}$.
- Construct the oracle $O : [2]^n \to \mathcal{K}$ where

$$O(x) = G_{x_1}(\dots G_{x_i}(P^{(i)}_{x_{i+1}}(x_{[i+2,n]}))\dots).$$

- Simulate A with oracle O, and output whatever A outputs.

Notice that each quantum query to O results in two quantum queries to P, one to compute $P^{(i)}_{x_{i+1}}(x_{[i+2,n]})$ and one to un-compute it after computing $O(x)$. Therefore, B makes at most twice the number of queries that A does.

Fix i, and let B_i be the algorithm B using this i. In the case where P is truly random, so is $P^{(i)}$, as are $P_0^{(i)}$ and $P_1^{(i)}$. Thus, $O = O_i$, the oracle in hybrid H_i. When P is drawn from $G \circ \mathcal{K}^{[2]^{n-1}}$, then

$P^{(i)}$ is distributed according to $G \circ \mathcal{K}^{[2]^{n-i-1}}$, and so $(P_0, P_1) \leftarrow G \circ \mathcal{K}^{[2]^{n-i-1}}$. Thus, $O = O_{i+1}$, the oracle in hybrid H_{i+1}. For fixed i, we then have that the quantity

$$\Pr_{P \leftarrow (\mathcal{K}^2)^{[2]^{n-1}}}\left[B_i^{|P\rangle}() = 1\right] - \Pr_{P \leftarrow G \circ \mathcal{K}^{[2]^{n-1}}}\left[B_i^{|P\rangle}() = 1\right]$$

is equal to ϵ_i. Averaging over all i and taking the absolute value, we have that the distinguishing probability of B,

$$\left|\Pr_{P \leftarrow (\mathcal{K}^2)^{[2]^{n-1}}}\left[B^{|P\rangle}() = 1\right] - \Pr_{P \leftarrow G \circ \mathcal{K}^{[2]^{n-1}}}\left[B^{|P\rangle}() = 1\right]\right|,$$

is equal to

$$\left|\frac{1}{n}\sum_i \epsilon_i\right| = \epsilon/n.$$

Thus, B breaks the oracle security of G with advantage only polynomially smaller than the advantage A distinguishes PRF from a random oracle. □

6 PSEUDORANDOM FUNCTIONS FROM SYNTHESIZERS

In this section, we show that the construction of pseudorandom functions from pseudorandom synthesizers due to Naor and Reingold [34] is quantum-secure. This construction is motivated by realizing PRFs computable in low circuit depth. We first recall the definition of a pseudorandom synthesizer.

Synthesizers. A synthesizer can be thought of as a special kind of pseudorandom function. Consider evaluating a PRF PRF on a polynomial-length sequence of random independent inputs $x_1, \ldots, x_q$, as well as a random key k: $(\mathsf{PRF}_k(x_1), \ldots, \mathsf{PRF}_k(x_q))$. Classical-query security of PRF immediately implies that these q outputs are indistinguishable from uniformly random and independent values.

A synthesizer essentially requires that the list remains indistinguishable from random, even if the adversary is given many lists for independently chosen keys k, but where the lists all share the same $x_1, \ldots, x_q$. That is, a single list of random inputs $x_1, \ldots, x_q$ and a single list of random keys $k_1, \ldots, k_q$ are chosen, and the adversary is either given all q^2 values $\mathsf{PRF}_{k_i}(x_j)$ or simply q^2 uniformly random values. The synthesizer is secure if the adversary cannot distinguish the two cases. Note that this definition is *weaker* than that of a pseudorandom function, since the inputs x_i are random instead of adversarially chosen. We now give the formal definition of a synthesizer, changing the syntax to reflect the symmetry between key and input in this setting.

Definition 6.1 (Synthesizer). A pseudorandom synthesizer is a function $S : \mathcal{X}^2 \to \mathcal{X}$. $\mathcal{X}$ is implicitly indexed by the security parameter n. We require that S can be evaluated by a classical algorithm in time polynomial in n.

We note that it is also possible to consider synthesizers where the range is not equal to $\mathcal{X}$; however, we only need to consider the case where the range is $\mathcal{X}$.

Definition 6.2 (Standard security). A pseudorandom synthesizer $S : \mathcal{X}^2 \to \mathcal{X}$ is standard-secure if, for any polynomial q and any efficient quantum adversary A, there exists a negligible function ϵ such that

$$\left|\Pr_{\substack{x_i \leftarrow \mathcal{X}, y_j \leftarrow \mathcal{X} \\ i,j \in [q]}}\left[A(\ (S(x_i, y_j))_{i,j \in [q]}\ ) = 1\right] - \Pr_{u_{i,j} \leftarrow \mathcal{X}, i,j \in [q]}\left[A(\ (u_{i,j})_{i,j \in [q]}\ ) = 1\right]\right| < \epsilon.$$

PRFs from synthesizers. We now recall the construction of a PRF from any synthesizer, as given by Naor and Reingold [34]. The intuition behind the construction is the following. A synthesizer can already be seen as giving a PRF with a very small input size as follows. The key consists of two vectors $(x_1, \ldots, x_q) \leftarrow \mathcal{X}^q$ and $(y_1, \ldots, y_q) \leftarrow \mathcal{X}^q$. The input set is $[q]^2$, and the evaluation of the PRF on input (i, j) is just $S(x_i, y_j)$.

This input size is too small, as it only allows for inputs from a polynomial-sized set. However, the important feature is that the total length of all outputs (which grows quadratically in q) is larger than the key length (which grows linearly in q). This allows for domain extension, roughly by setting the vectors $(x_1, \ldots, x_q)$ and $(y_1, \ldots, y_q)$ to themselves be the outputs of PRFs. That is, recursively construct two PRFs PRF_1 and PRF_2 with domain $\mathcal{Z}$. Then build a new PRF PRF with domain $\mathcal{Z}^2$, defined as $\mathsf{PRF}(z_1, z_2) = S(\mathsf{PRF}_1(z_1), \mathsf{PRF}_2(z_2))$.

The key consequence of the above construction is that every level of the recursion doubles the length of the input, while adding a fixed amount (essentially the depth of S) to the depth of the evaluation. Thus, the overall evaluation depth is a logarithmic factor deeper than the depth of S. By using, say, the log-depth synthesizer from the LWE problem (LWE) of Banerjee et al. [14], this gives a PRF with log-squared depth.

We now give the construction in detail.

Construction 2 (NR-PRF). *Given a pseudorandom synthesizer $S : \mathcal{X}^2 \to \mathcal{X}$, let ℓ be an integer and $n = 2^\ell$. We let $\mathsf{PRF}_k(x) = \mathsf{PRF}_k^{(\ell)}(x)$ where $\mathsf{PRF}^{(i)} : (\mathcal{X}^{2\times 2^i}) \times [2]^{2^i} \to \mathcal{X}$ is defined as*

$$\mathsf{PRF}^{(0)}_{a_{1,0}, a_{1,1}}(x) = a_{1,x},$$

$$\mathsf{PRF}^{(i)}_{A_1^{(i-1)}, A_2^{(i-1)}}(x) = S\left(\mathsf{PRF}^{(i-1)}_{A_1^{(i-1)}}(x_{[1,2^{i-1}]}),\ \mathsf{PRF}^{(i-1)}_{A_2^{(i-1)}}(x_{[2^{i-1}+1,2^i]})\right),$$

where

$$A_1^{(i-1)} = (a_{1,0}, a_{1,1}, a_{2,0}, a_{2,1}, \ldots, a_{2^{i-1},0}, a_{2^{i-1},1}),$$
$$A_2^{(i-1)} = (a_{2^{i-1}+1,0}, a_{2^{i-1}+1,1}, a_{2,0}, a_{2,1}, \ldots, a_{2^i,0}, a_{2^i,1}).$$

That is, PRF takes a key k consisting of $2 \times 2^\ell$ elements of $\mathcal{X}$, and takes bit strings x of length 2^ℓ as input. It uses x to select 2^ℓ of the elements in the key, and pairs them off. It then applies S to each of the pairs, obtaining $2^{\ell-1}$ elements of $\mathcal{X}$. Next, PRF pairs these elements and applies S to these pairs again, and continues in this way until there is one element left, which becomes the output.

6.1 Security of the Naor-Reingold PRF

Here, we prove the main theorem of this section, Theorem 6.3, which is reproduced here for convenience.

Theorem 6.3. *If S is a standard-secure synthesizer, then* PRF *from Construction 2 is a QPRF.*

The proof is very analogous to that of the security of the GGM construction: we define a new notion of security for synthesizers, called quantum security, and use the techniques of Naor and Reingold to prove that quantum security implies that Construction 2 is quantum secure. Unlike the GGM case, the equivalence of quantum and standard security for synthesizers is not an immediate consequence of Theorem 4.5, due to their being two correlated inputs to S. Nevertheless, we prove the equivalence, completing the proof of security for Construction 2. The proof follows similar ideas to the proof of Theorem 4.5, using small-range distributions to simulate the oracles using a polynomial number of samples.

We first give a new definition of security for synthesizers that is a simple syntactic change to Definition 6.2 and is readily seen to be equivalent. However, it will be convenient for our proof. The idea is that, rather than giving the adversary all q^2 outputs of S, we give the adversary an *oracle* indexing the values. The adversary can then query on any pair (i, j) and receive $S(x_i, y_j)$ in response. Of course, a classical query algorithm can simply query the entire q^2 inputs (since q^2 is polynomial) to get the list of outputs. Therefore, there is no difference in the definitions. This remains true, even if we allow the adversary to query the oracle on quantum superpositions.

Definition 6.4 (Standard Security, Restated). A pseudorandom synthesizer $S : \mathcal{X}^2 \to \mathcal{X}$ is standard-secure if, for any set $\mathcal{Z}$ where $|\mathcal{Z}| \in n^{O(1)}$, no efficient quantum algorithm A making quantum queries can distinguish $O(z_1, z_2) = S(O_1(z_1), O_2(z_2))$ where $O_b \leftarrow \mathcal{X}^{\mathcal{Z}}$ from $O \leftarrow \mathcal{X}^{\mathcal{Z}\times\mathcal{Z}}$. That is, for any such A, there exists a negligible ϵ such that

$$\left| \Pr_{O_1, O_2 \leftarrow \mathcal{X}^{\mathcal{Z}}} \left[A^{|S(O_1(\cdot), O_2(\cdot))\rangle}() = 1 \right] - \Pr_{P \leftarrow \mathcal{X}^{\mathcal{Z}\times\mathcal{Z}}} \left[A^{|O(\cdot,\cdot)\rangle}() = 1 \right] \right| < \epsilon.$$

In order to prove security of the Naor-Reingold PRF, we define a new notion of security for synthesizers that we call quantum security. The definition is identical to Definition 6.4, except that there is no bound on the size of $\mathcal{Z}$; in particular, $\mathcal{Z}$ can be exponential.

Definition 6.5 (Quantum Security). A pseudorandom synthesizer $S : \mathcal{X}^2 \to \mathcal{X}$ is quantum-secure if, for any set $\mathcal{Z}$, no efficient quantum algorithm A making quantum queries can distinguish $O(z_1, z_2) = S(O_1(z_1), O_2(z_2))$ where $O_b \leftarrow \mathcal{X}^{\mathcal{Z}}$ from $O \leftarrow \mathcal{X}^{\mathcal{Z}\times\mathcal{Z}}$. That is, for any such A, there exists a negligible ϵ such that

$$\left| \Pr_{O_1, O_2 \leftarrow \mathcal{X}^{\mathcal{Z}}} \left[A^{|S(O_1(\cdot), O_2(\cdot))\rangle}() = 1 \right] - \Pr_{P \leftarrow \mathcal{X}^{\mathcal{Z}\times\mathcal{Z}}} \left[A^{|O(\cdot,\cdot)\rangle}() = 1 \right] \right|.$$

Our first step to proving Theorem 6.3 is to show that standard security and quantum security are equivalent.

Lemma 6.6. *If S is standard-secure, then it is also quantum-secure.*

Proof. Let B be an adversary breaking the quantum security of S with non-negligible advantage ϵ, meaning

$$\epsilon = \left| \Pr_{O_1 \leftarrow \mathcal{X}^{\mathcal{Z}}, O_2 \leftarrow \mathcal{X}^{\mathcal{Z}}} [B^{S(O_1, O_2)}() = 1] - \Pr_{O \leftarrow \mathcal{Y}^{\mathcal{Z}\times\mathcal{Z}}} [B^{O}() = 1] \right|.$$

At a high level, we will replace O_1 and O_2 with $\mathsf{SR}^{\mathcal{X}}_r$, for an appropriate choice of r, meaning the oracle on the left-hand side becomes $S \circ (\mathsf{SR}^{\mathcal{X}}_r \times \mathsf{SR}^{\mathcal{X}}_r)$. We will also replace the oracle O with a suitable oracle distribution so that the right-hand side has a similar structure. Our reduction will then simulate both sides, using an oracle from $[r]^2 \to \mathcal{Y}$ that corresponds to the oracles in the standard-security experiment for S.

One minor technical subtlety is that, in the proof, we will find it convenient to take uniform random values over infinite sets. Of course, such a distribution is not well-defined. Instead, we will introduce a parameter s, take random values over the set $[s]$, and let s go to infinity.

Let us now define a new oracle distribution, which we will denote AR_s, which stands for *almost random*. AR_s is defined as follows:

- Pick random oracles P_1 and P_2 from $[s]^{\mathcal{Z}}$.
- Pick a random oracle Q from $\mathcal{Y}^{[s]^2}$.
- Output the oracle $O : \mathcal{Z}^2 \to \mathcal{Y}$ where $O(z_1, z_2) = Q(P_1(z_1), P_2(z_2))$.

Notice that as s goes to ∞, P_1 and P_2 become injective with probability approaching 1, and thus AR_∞ is the uniform distribution on $\mathcal{Y}^{\mathcal{Z}^2}$. AR_s is thus an approximation to the uniform distribution on oracles, with a structure that will be amenable to our proof. We define $\epsilon(s)$ as

$$\epsilon(s) = \left| \Pr_{O_1 \leftarrow \mathcal{X}^{\mathcal{Z}}, O_2 \leftarrow \mathcal{X}^{\mathcal{Z}}}[B^{S(O_1,O_2)}() = 1] - \Pr_{O \leftarrow \mathsf{AR}_s}[B^O() = 1] \right|.$$

Then $\epsilon = \lim_{s\to\infty} \epsilon(s)$. Notice that each query to O ($S(O_1, O_2)$, respectively) can be simulated using two queries to each of P_1, P_2 (O_1, O_2, respectively). Let q be the number of queries made by B. Then the number of queries made to each of O_1, O_2, P_1, P_2 is bounded by $2q$.

Let r be an integer such that $\ell(2q)/r = \epsilon/8$ where q is the number of queries made by B and $\ell(\cdot)$ is the polynomial from Corollary 4.4. We now replace O_1, O_2 with $\mathsf{SR}_r^{\mathcal{X}}$, and P_1, P_2 (as a part of AR_s) with $\mathsf{SR}_r^{[s]}$. By Corollary 4.4, each of these changes will only change the behavior of B by $\epsilon/8$. Thus, a simple argument shows that

$$\left| \Pr_{O_i \leftarrow \mathsf{SR}_r^{\mathcal{X}}}[B^{S(O_1,O_2)}() = 1] - \Pr_{P_i \leftarrow \mathsf{SR}_r^{[s]}, Q \leftarrow \mathcal{Y}^{[s]^2}}[B^{Q(P_1,P_2)}() = 1] \right| \geq \epsilon(s) - \epsilon/2.$$

Notice that we can think of the oracle $Q(P_1, P_2)$ for $P_i \leftarrow \mathsf{SR}_r^{[s]}$ as the oracle

$$O'(z_1, z_2) = Q(V_1 \circ R_1(z_1), V_2 \circ R_2(z_2)) = Q'(R_1(z_1), R_2(z_2)),$$

where $V_i \leftarrow [s]^{[r]}$, $R_i \leftarrow [r]^{\mathcal{Z}}$, and $Q'(w_1, w_2) = Q(V_1(w_1), V_2(w_2))$. As s goes to ∞, V_i becomes injective with probability converging to one, so Q' approaches a random function from $[r]^2 \to \mathcal{Y}$. Thus, we have that

$$\left| \Pr_{O_i \leftarrow \mathsf{SR}_r^{\mathcal{X}}}[B^{S(O_1,O_2)}() = 1] - \Pr_{R_i \leftarrow [r]^{\mathcal{Z}}, Q' \leftarrow \mathcal{Y}^{[r]^2}}[B^{Q'(R_1,R_2)}() = 1] \right| \geq \epsilon/2.$$

We now describe a new algorithm A which tries to break the standard security of S according to Definition 6.4. A takes as input a quantum-accessible oracle O from $[r]^2$ to $\mathcal{Y}$. A constructs two random oracles $R_1 \leftarrow [r]^{\mathcal{Z}}$ and $R_2 \leftarrow [r]^{\mathcal{Z}}$, gives B the oracle $O'(z_1, z_2) = O(R_1(z_1), R_2(z_2))$, and simulates B. If $O = S(T_1, T_2)$ for random oracles $T_i \leftarrow \mathcal{X}^{[r]}$, then the oracle seen by A is $O'(z_1, z_2) = S(O_1(z_1), O_2(z_2))$, where O_1 and O_2 are drawn from $\mathsf{SR}_r^{\mathcal{X}}$. If O is a random oracle, then the oracle seen by A is $O'(z_1, z_2) = O(R_1(z_1), R_2(z_2))$, where $R_i \leftarrow [r]^{\mathcal{Z}}$. The advantage of A in distinguishing these two cases is therefore at least $\epsilon/2$. If ϵ is non-negligible, then there is a polynomial bounding r infinitely often, and in these cases, A breaks the standard security of S. The only remaining piece is the efficiency of A: As in the proof of Theorem 4.5, A is not efficient due to sampling exponential-sized random oracles. But as in Theorem 4.5, these oracles can be replaced by k-wise independent functions using Corollary 2.2. Thus, we get an efficient A. □

We are now ready to prove that Construction 2 is quantum secure. The proof will follow that of Naor and Reingold [34], but replacing a key step of their proof with Lemma 6.6.

Proof of Theorem 6.3. Let A be a quantum adversary breaking the quantum security of PRF with advantage ϵ. That is,

$$\left| \Pr_{k \leftarrow \mathcal{X}^{2n}}[A^{\mathsf{PRF}_k}() = 1] - \Pr_{O \leftarrow \mathcal{X}^{[2]^n}}[A^O() = 1] \right| = \epsilon.$$

Let hybrid H_i be the game where the oracle seen by A is PRF, except that each instance of $\mathsf{PRF}^{(i)}$ is replaced with a truly random function from $[2]^{2^i}$ into $\mathcal{X}$. Since $\mathsf{PRF}^{(0)}$ is already a random function, H_0 is equivalent to the case where the oracle is PRF. Similarly, H_ℓ is by definition the

case where the oracle is truly random. Thus, a simple hybrid argument shows that there is an i such that A can distinguish H_i from H_{i-1} with probability at least ϵ/ℓ.

We now describe an algorithm B which breaks the quantum security of S. B is given an oracle from P from $(\mathcal{X} \times [2^{\ell-i}])^2$ into $\mathcal{X}$, which is either $S(Q_1, Q_2)$ for random oracles $Q_b \leftarrow \mathcal{X}^{\mathcal{X}\times[2^{\ell-i}]}$, or it is given a truly random oracle. It then constructs oracles

$$P_y(x_1, x_2) = P((x_1, y), (x_2, y)).$$

Notice that there are $2^{\ell-i}$ possible y values, and that for fixed y, P_y is either a random oracle from $\mathcal{X}^2$ into $\mathcal{X}$, or it is $S(Q_{y,1}, Q_{y,2})$ for random oracles $Q_{y,b}$ from $\mathcal{X}$ to $\mathcal{X}$. We then construct the oracle O which is PRF, except that we stop the recursive construction at $\mathsf{PRF}^{(i)}$. There are $2^{\ell-i}$ different instances of $\mathsf{PRF}^{(i)}$, so we use the $2^{\ell-i}$ different P_y oracles in their place. If P is $S(Q_1, Q_2)$, this corresponds to hybrid H_{i-1}, whereas if P is a random oracle, this corresponds to H_i. Thus, B distinguishes the two cases with probability ϵ/ℓ.

However, under the assumption that S is standard secure, Lemma 6.6 shows that it is quantum secure, meaning the algorithm B is impossible. Therefore, PRF is quantum-secure. □

7 DIRECT CONSTRUCTION OF PSEUDORANDOM FUNCTIONS

In this section, we present the construction of pseudorandom functions from Banerjee et al. [14]. We show that this construction is quantum-secure.

Let p, q be integers with $q > p$. Let $\lfloor x \rceil_p$ be the map from $\mathbb{Z}_q$ into $\mathbb{Z}_p$ defined by first rounding x to the nearest multiple of q/p, and then interpreting the result as an element of $\mathbb{Z}_p$. More precisely, $\lfloor x \rceil_p = \lfloor (p/q)x \rceil \mod p$ where the multiplication and division in $(p/q)x$ are computed in $\mathbb{R}$.

Construction 3. *Let p, q, m, ℓ be integers with $q > p$. Let $\mathcal{K} = \mathbb{Z}_q^{n\times m} \times (\mathbb{Z}^{n\times n})^\ell$. We define $\mathsf{PRF} : \mathcal{K} \times [2]^\ell \to \mathbb{Z}_p^{m\times n}$ as follows: For a key $k = (\mathbf{A}, \{\mathbf{S}_i\})$, let*

$$\mathsf{PRF}_k(x) = \left\lfloor \mathbf{A}^T \cdot \prod_{i=1}^{\ell} \mathbf{S}_i^{x_i} \right\rceil_p .$$

We note that we follow the convention from Banerjee et al. that $\mathbf{A}$ is transposed in the definition of PRF. The function PRF uses for a key an $n \times m$ matrix $\mathbf{A}$ and ℓ different $n \times n$ matrices $\mathbf{S}_i$, where elements are integers mod q. It uses its ℓ-bit input to select a subset of the $\mathbf{S}_i$, which it multiplies together. The product is then multiplied by the transpose of $\mathbf{A}$, whose result is rounded mod p.

Next, we prove that Construction 3 is secure when the secret key $(\mathbf{A}, \{\mathbf{S}_i\})$ is drawn from an appropriate distribution.

7.1 Security of the BPR PRF

First, we need to define the LWE (LWE) problem, as defined by Regev [37]. Consider the basic linear algebra task of determining if a vector u lies in the (column-)span of a random matrix. This task is solvable in polynomial time using, say, Gaussian elimination. The LWE problem considers essentially the same task, but where the vector u is "noisy." That is, each entry of u has been perturbed by some independent *small* noise. The addition of even very small noise appears to make the problem significantly more challenging. In fact, as Regev shows, the LWE problem is at least as hard as the hardness of certain worst-case lattice problems, which in turn are widely believed to be hard even for quantum computers.

Banerjee et al. define a variant of the basic LWE problem, which they explain is equivalent to the basic version, but is more convenient for their security proof. We will use the same variant. Here, rather than a random matrix being chosen all at once, the adversary is given rows of the matrix,

together with the corresponding entry of u. The adversary can request as many row/entry pairs as he would like. Additionally, rather than a single u vector, there are now w such vectors which are all different linear combinations of the columns of the random matrix. Each sample gives the corresponding entries of each of the u vectors. Finally, for notational convenience, m samples are given at a time, rather than a single sample, and these samples are arranged in a matrix. We now give the definition.

Definition 7.1 (LWE). Let $q \geq 2$ be an integer, n a security parameter, and $m = poly(n)$ and $w = poly(n)$ be integers. For a distribution χ over $\mathbb{Z}$ and a secret matrix $\mathbf{S} \in \mathbb{Z}_q^{n\times w}$, the LWE distribution $\mathsf{LWE}_{\mathbf{S},\chi}$ is the distribution over $\mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^{m\times w}$ defined as follows:

- Choose a random matrix $\mathbf{A} \leftarrow \mathbb{Z}_q^{n\times m}$.
- Choose a random error matrix $\mathbf{E} \leftarrow \chi^{m\times w}$.
- Output $(\mathbf{A}^T, \mathbf{B}^T = \mathbf{A}^T \cdot \mathbf{S} + \mathbf{E} \mod q)$.

The LWE problem is then to distinguish between a polynomial number of samples from $\mathsf{LWE}_{\mathbf{S},\chi}$ for a fixed $\mathbf{S} \leftarrow \chi^{n\times w} \mod q$ from the same number of samples from the uniform distribution. The LWE problem is hard if, for all efficient quantum adversaries A, the probability A distinguishes these two cases is negligible in n.

Here again, we follow the convention from Banerjee et al. that $\mathbf{A}$ is transposed in the output of the LWE samples. Analogous to Constructions 1 and 2, we will prove the security of Construction 3 by first defining an oracle version of the LWE problem. Proving the hardness of this oracle version captures the main difficulty of translating the classical proof to the quantum setting. We therefore prove the equivalent of the plain and oracle versions of LWE. Then we follow a very similar proof as in Banerjee et al., except that we swap out a key step with our new notion of hardness of LWE.

We now define the oracle-LWE problem.

*Definition 7.2 (Oracle-*LWE*).* The oracle-LWE problem is to distinguish an oracle O whose outputs are generated by $\mathsf{LWE}_{\mathbf{S},\chi}$ (where the same $\mathbf{S} \leftarrow \chi^{n\times w} \mod q$ is used for all points) from a truly random oracle O. We say that LWE is oracle-hard if, for all efficient adversaries A making quantum queries, A cannot distinguish these two distributions with more than negligible probability.

Lemma 7.3. *If* LWE *is hard, it is also oracle-hard.*

Proof. The proof is essentially the same as in Theorem 4.5, except that there is a common secret $\mathbf{S}$ across all the samples. Let A be an adversary breaking the oracle-hardness of LWE using q quantum queries with advantage ϵ. Let r be an integer such that $\ell(q)/r \approx \epsilon/4$, where $\ell(q)$ is the polynomial from Corollary 4.4. We then construct an algorithm B, which takes as input r pairs $(\mathbf{A}_i^T, \mathbf{B}_i^T)$, and distinguishes when the pairs come from $\mathsf{LWE}_{\mathbf{S},\chi}$ for some fixed $\mathbf{S} \leftarrow \chi^{n\times w}$ from when the pairs are random. B works as follows:

- Construct the oracle O where $O(x) = (\mathbf{A}_{P(i)}^T, \mathbf{B}_{P(i)}^T)$. Here, $P(i)$ is a random function with range $[r]$.
- Simulate A with oracle O, and output the output of A.

Using the same analysis as in the proof of Theorem 4.5, we get that B distinguishes the two cases with advantage $\epsilon/2$. If ϵ is non-negligible, then there is a polynomial that bounds r infinitely often, and in these cases, the number of samples received by B is a polynomial, and hence B breaks the hardness of LWE. □

Next, we need to define the discrete Gaussian distribution, which is the noise distribution most often used in LWE.

Definition 7.4 (Discrete Gaussian). Let $D_{\mathbb{Z},r}$ denote the discrete Gaussian distribution over $\mathbb{Z}$, where the probability of x is proportional to $e^{-\pi x^2/r^2}$.

We are now ready to state and prove the security of Construction 3.

Theorem 7.5 (Formal Version). *Let $\chi = D_{\mathbb{Z},r}$, and $q \geq p \cdot \ell(Cr\sqrt{n+\ell})^\ell n^{\omega(1)}$ for some suitable universal constant C. Let* PRF *be as in Construction 3, and suppose each S_i is drawn from $\chi^{n\times n}$. If the* LWE *problem is hard for modulus q and distribution χ, then* PRF *from Construction 3 is a QPRF.*

Proof. The proof is very similar to that of Banerjee et al. Notice that our theorem requires $q \geq p \cdot \ell(Cr\sqrt{n+\ell})^\ell n^{\omega(1)}$, whereas the original only requires $q \geq p \cdot \ell(Cr\sqrt{n})^\ell n^{\omega(1)}$. We will explain why this is later, but note here that it does not fundamentally change the asymptotic nature of q and p.

The proof idea, as in Banerjee et al., is the following: if, before rounding, we add error to the subset product in Construction 3, the error will most likely get eliminated by the rounding. The only case where the error affects the output of rounding is when the pre-rounded value was close to a rounding boundary. By the choice of large q, it is very unlikely to be close to a rounding boundary. The proof then proceeds by adding a certain error to the evaluations, and then interpreting the result as being derived from LWE samples. The LWE samples are then replaced with truly random outputs. This process is repeated several times until all outputs are truly random.

Much like the classical proofs for the GGM and Naor-Reingold constructions, the classical proof of Construction 3 relies on the PRF queries being classical. In this case, each PRF query will require another PRF sample. For quantum security, we will modify the proof accordingly, using quantum queries to LWE oracle, and then relying on LWE being oracle-hard.

We now give the proof. Let A be an adversary that distinguishes PRF from a random function with advantage ϵ. First, consider the case where A sees a truly random function $U : [2]^k \to \mathbb{Z}_p^{m\times n}$. Let p_0 be the probability A outputs 1 in this case.

Replace U with $\lfloor U' \rceil_p$ where U' is a truly random function from $[2]^k \to \mathbb{Z}_q^{m\times n}$, and let p_1 be the probability A outputs 1 in this case.

Claim 3. *$|p_0 - p_1|$ is negligible.*

Proof. The distribution of outputs of U' is somewhat biased if p does not exactly divide q. However, the bias of each output is negligible because $q \geq pn^{\omega(1)}$. By Theorem 4.5, the ability of A to distinguish these two cases, and thus $|p_0 - p_1|$, is negligible. □

For a set S, let $\lfloor S \rceil_p$ be the set obtained by rounding each element of S and discarding duplicates. Now, let $B = \ell(Cr\sqrt{n+k})^\ell$. Let $\mathrm{BAD}(y)$ be the event that

$$\lfloor y + [-B, B]^{m\times n} \rceil_p \neq \{\lfloor y \rceil_p\}.$$

That is, $\mathrm{BAD}(y)$ is the event that y is very close to another element in $\mathbb{Z}_q$ that rounds to a different value in $\mathbb{Z}_p$. Banerjee et al. show that for each x, the probability that $\mathrm{BAD}(U'(x))$ occurs is negligible. Therefore, according to Theorem 4.5, $\mathrm{BAD}(U'(x))$ as an oracle with outputs in {True, False} is indistinguishable from the oracle that always outputs False, even under quantum queries. Hence, it is impossible for an algorithm making quantum queries to U' to find an x such that $\mathrm{BAD}(U'(x))$ occurs, except with negligible probability.

Now we define a class of functions $G : \mathcal{K} \times [2]^k \to \mathbb{Z}_q^{m \times n}$ to be PRF without rounding. That is,

$$G_k(x) = \mathbf{A}^T \cdot \prod_{i=1}^{\ell} \mathbf{S}_i^{x_i}.$$

Then $\mathsf{PRF}_k(x) = \lfloor G_k(x) \rceil_p$. We also define a related class of functions $\tilde{G}$ where $\tilde{G} = \tilde{G}^{(\ell)}$ and

- $\tilde{G}^{(0)}$ is a function from $[2]^0$ into $\mathbb{Z}_q^{m \times n}$ defined as follows: pick a random $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, and set $\tilde{G}^{(0)}(\emptyset) = \mathbf{A}^T$.
- $\tilde{G}^{(i)}$ is a function from $[2]^i$ into $\mathbb{Z}_q^{m \times n}$ defined as follows: sample $\tilde{G}^{(i-1)}$, pick $\mathbf{S}_i \leftarrow \chi^{n \times n}$ and for each $x' \in [2]^{i-1}$, pick $\mathbf{E}_{x'} \leftarrow \chi^{m \times n}$. Then

$$\tilde{G}^{(i)}(x = x'x_i) = \tilde{G}^{(i-1)}(x') \cdot \mathbf{S}_i^{x_i} + x_i \cdot \mathbf{E}_{x'} \mod q.$$

Now consider giving A the oracle $\lfloor \tilde{G}(x) \rceil_p$, and let p_2 be the probability A outputs 1 in this case. For now, let us assume $\tilde{G}(x)$ and $U'(x)$ (with no rounding) are indistinguishable. We will return to proving this later.

Claim 4. *Assuming that the oracles $\tilde{G}(x)$ and $U'(x)$ are indistinguishable under quantum queries, $|p_2 - p_1|$ is negligible.*

Proof. $|p_2 - p_1|$ is just A's distinguishing probability between $\lfloor \tilde{G}(x) \rceil_p$ and $\lfloor U'(x) \rceil_p$, the same oracles that are assumed to be indistinguishable, just with a final rounding. This rounding can only decrease the distinguishing ability. Concretely, if $|p_2 - p_1|$ is non-negligible, we let B be a distinguisher for $\tilde{G}(x)$ and $U'(x)$ which simply answers all queries from A by making the query to its oracle, and rounding the result. B's advantage will be the same as A's. □

Additionally, assuming $\tilde{G}(x)$ and $U'(x)$ are indistinguishable, it is also impossible to make queries to $\tilde{G}(x)$ and output an x such that $\mathsf{BAD}(\tilde{G}(x))$ occurs, except with negligible probability.

Next, we replace $\tilde{G}$ with G, giving the oracle $\lfloor G(x) \rceil_p$. Let p_3 be the probability A outputs 1 in this case.

Claim 5. *$|p_3 - p_2|$ is negligible.*

Proof. Banerjee et al. show that as long as $\mathsf{BAD}(\tilde{G}(x))$ does not occur, $\lfloor \tilde{G}(x) \rceil_p = \lfloor G_k(x) \rceil_p = \mathsf{PRF}_k(x)$ with all but negligible probability. Plugging in our modification of the parameters in the statement of Theorem 7.5 (namely, replacing $\sqrt{n}$ with $\sqrt{n+\ell}$) results in this probability actually being $2^{-\ell}\sigma$ for some negligible σ. Summing over all 2^ℓ different x, we get that, except with negligible overall probability, $\mathsf{PRF}_k(x) = \lfloor \tilde{G}(x) \rceil_p$ for all x where $\mathsf{BAD}(\tilde{G}(x))$ does not occur.

Now, if $|p_3 - p_2|$ is non-negligible this means A distinguishes $\lfloor G(x) \rceil_p$ from $\lfloor \tilde{G}(x) \rceil_p$ with non-negligible advantage. By the above, except with negligible probability over the choice of $\tilde{G}$, these two oracles only differ on inputs where $\mathsf{BAD}(\tilde{G}(x))$ occurs. Our goal is to turn A into an algorithm which actually finds such a bad x with non-negligible probability, contradicting the facts proved above.

In this classical setting, this is straightforward: since the oracles are identical except on bad x, a classical query algorithm must query on such a bad x to have any chance at distinguishing. We can therefore come up with an adversary B which makes queries to $\tilde{G}(x)$ and simulates A by forwarding A's queries to its oracle and rounding the result. Then at a randomly chosen query, B will abort and output A's query.

In the quantum case, the proof is somewhat analogous, but more care is needed. The following strategy is implicit in Boneh et al. [8], which uses the following notation and lemmas from Bennett et al. [7]. Define the Euclidean distance $||\phi\rangle - |\psi\rangle|$ between two quantum states as $(\sum_x |\alpha_x - \beta_x|^2)^{\frac{1}{2}}$ where $|\phi\rangle = \sum_x \alpha_x |x\rangle$ and $|\psi\rangle = \sum_x \beta_x |x\rangle$.

Lemma 7.6 ([7] Theorem 3.1). *Let $|\varphi\rangle$ and $|\psi\rangle$ be quantum states with Euclidean distance at most ϵ. Then, performing the same measurement on $|\varphi\rangle$ and $|\psi\rangle$ yields distributions with statistical distance at most 4ϵ.*

For an oracle algorithm A making queries to an oracle O, let $|\phi_t\rangle$ be the state submitted for the tth query. Define $q_r(|\phi_t\rangle)$ to be the magnitude squared of r in the tth query, which we will call the *query probability* of r. If we sum over all queries, we get the total query probability of r.

Lemma 7.7 ([7] Theorem 3.3). *Let A be a quantum algorithm running making T queries to an oracle O. Let $\epsilon > 0$ and let $S \subseteq [1,T] \times \{0,1\}^n$ be a set of time-string pairs such that $\sum_{(t,r)\in S} q_r(|\phi_t\rangle) \leq \epsilon$. If we modify O into an oracle O' which is identical to O except possibly at times-string pairs $(t,r) \in S$, then the Euclidean distance between the final states of A when invoking O and O' is at most $\sqrt{T\epsilon}$.*

To use these lemmas, fix a key k, and sample $\tilde{G}$, setting the $\mathbf{A}$ and $\mathbf{S}$ matrices to be those from the key k. If A distinguishes $\mathsf{PRF}_k(x) = \lfloor G(x) \rceil$ from $\lfloor \tilde{G}(x) \rceil_p$ with non-negligible probability, Lemma 7.6 implies the Euclidean distance between the final states is non-negligible.

Let W be the set of x where $\mathsf{PRF}_k(x)$ and $\lfloor \tilde{G}(x) \rceil_p$. Consider running A using PRF_k, and let $\tau \equiv \sum_{x\in W,t} q_x(|\phi_t\rangle)$ be the total query probability on such x. Then we invoke Lemma 7.7 to conclude that τ is non-negligible.

We can then *find* an $x \in W$ as follows: run A, forwarding queries to $\tilde{G}(x)$ and rounding the result.[5] However, at a randomly chosen query, abort A, and measure and output the query. If the query chosen is t, the probability of outputting such an x is $\sum_{x\in W} q_x(|\phi_t\rangle)$. Over all T queries, the probability of outputting such an x is then τ/T, which is non-negligible.

Finally, since $x \in W$ implies that $\mathsf{BAD}(\tilde{G}(x))$ occurs, we therefore have that our new algorithm outputs an x such that $\mathsf{BAD}(\tilde{G}(x))$ occurs with non-negligible probability, a contradiction. □

Hence, we have shown that PRF is indistinguishable from a random function. It remains to show that U' and $\tilde{G}$ are oracle-indistinguishable. This is the main part of our proof that differs from Banerjee et al.'s proof. We show that U' and $\tilde{G}$ are indistinguishable given that the LWE problem is oracle-hard. Using Lemma 7.3, we reach the same conclusion assuming LWE is hard, thus completing the theorem.

Let B be an adversary that distinguishes U' from $\tilde{G}$ with advantage ϵ. Define hybrid H_i as the case where B is given the oracle O_i where $O_i = \tilde{G}$, except that, in the recursive definition of $\tilde{G}$, $\tilde{G}^{(i)}$ is replaced with a truly random function. H_0 corresponds to the correct definition of $\tilde{G}$, and H_ℓ corresponds to U'. Thus, there exists an i such that B distinguishes H_i from H_{i-1} with advantage ϵ/ℓ.

Construct an adversary C with access to an oracle $P : [2]^{i-1} \to \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^{m\times n}$. P is either a random function or each output is chosen according to the LWE distribution using a common random secret $\mathbf{S}$. In other words, $P(x) = (\mathbf{A}^T, \mathbf{B}^T)$, where either $\mathbf{A}(x)$ and $\mathbf{B}(x)$ are chosen at random for all x, or there is a secret $\mathbf{S} \leftarrow \chi^{n\times n}$ and $\mathbf{B}(x)^T = \mathbf{A}(x)^T\mathbf{S} + \mathbf{E}(x) \mod q$ where $\mathbf{E}(x) \leftarrow \chi^{m\times n}$.

[5] In order to simulate the A's queries correctly, we will need to make two queries to $\tilde{G}(x)$: one to obtain $\tilde{G}(x)$, and then another to un-compute $\tilde{G}(x)$ after producing the rounded value.

For each $j > i$, C constructs random oracles $Q_j : [2]^{j-1} \to \mathbb{Z}^{m \times n}$ where $Q_j(x) \leftarrow \chi^{m \times n}$. C also generates $\mathbf{S}_j \leftarrow \chi^{n \times n}$ for $j > i$. Then C works as follows:

- Let $\tilde{G}^{(i)}(x = x'x_i) = \begin{cases} \mathbf{A}(x')^T & \text{if } x_i = 0 \\ \mathbf{B}(x')^T & \text{if } x_i = 1 \end{cases}$.
- Let $\tilde{G}^{(j)}(x = x'x_j) = \tilde{G}^{(j-1)}(x') \cdot \mathbf{S}_j^{x_j} + x_j \cdot Q_j(x') \mod q$ for $j > i$.
- Let $O(x) = \tilde{G}^{(\ell)}(x)$.
- Run B with oracle O.

When P is a random oracle, this corresponds to H_i. When P is the LWE oracle, this corresponds to H_{i-1}. Thus, C distinguishes these two cases with advantage at least ϵ/ℓ. Under the assumption that LWE is oracle hard, this quantity, and hence ϵ, are negligible. We then use Lemma 7.3 to complete the theorem. □

8 PROOF OF THEOREM 4.2

Here, we prove Theorem 4.2. Here, we state a more general version of the theorem, which considers potentially stronger restrictions on the polynomials in question. Our reasons for giving this more general version is that an analogous generalization of Theorem 4.1 was crucially used by Zhandry [43]. While we do not need the generalization for our results, we provide the generalization in case it is useful beyond the scope of this work.

Theorem 4.2. *Fix $q > 0$ and $\Delta > 0$, and let E_r be a family of distributions on $\mathcal{Y}^{\mathcal{X}}$ indexed by $r \in \mathbb{Z}^+ \bigcup \{\infty\}$. Suppose there is an integer d such that for every $2q$ pairs $(x_i, r_i) \in \mathcal{X} \times \mathcal{Y}$, the function $p(\lambda) = \Pr_{H \leftarrow E_{1/\lambda}}[H(x_i) = r_i \forall i \in \{1, \ldots, 2q\}]$ satisfies*

- *$p(\lambda)$ is a polynomial of degree at most d in λ and*
- *$p^{(i)}(0)$, the ith derivative of p at 0, is 0 for each $i \in \{1, \ldots, \Delta - 1\}$.*

Then for any quantum algorithm A making q quantum queries, the output distributions under E_r and E_∞ are $2^{2-\Delta}\zeta(2\Delta)(1/r)^{\Delta}(d)^{3\Delta}$-close, where ζ is the Riemann Zeta function.

Setting $\Delta = 1$ gives the original statement of the theorem, using the fact that $\zeta(2) = \pi^2/6$.

Let $\lambda = 1/r$. We follow a similar proof technique as in Zhandry [43]. By Theorem 2.1, for a q-query quantum algorithm A, $\Pr_{H \leftarrow E_r}[A^H() = z]$ is a linear combination of the $\Pr_{H \leftarrow E_r}[H(x_i) = r_i \forall i \in \{1, \ldots, 2q\}]$. Thus, for any z, $\Pr_{H \leftarrow E_{1/\lambda}}[A^H() = z]$ is a polynomial in λ of degree d with the first $\Delta - 1$ derivatives at $\lambda = 0$ being 0.

Now, suppose that A distinguishes $E_{1/\lambda}$ from E_∞ with advantage $\epsilon(\lambda)$. That is,

$$\sum_z \left| \Pr_{H \leftarrow E_{1/\lambda}}[A^H() = z] - \Pr_{H \leftarrow E_\infty}[A^H() = z] \right| = \epsilon(\lambda).$$

We would like to have that ϵ is a polynomial in λ, so that we can analyze the degree constraints on ϵ in order to get the desired bound. The quantity in the absolute value is indeed a polynomial by assumption; if the absolute value were removed, the entire sum would as well be a polynomial. Unfortunately, the absolute value in the sum means that ϵ is not a polynomial. Worse, the sign changes of the summands will occur at different points, meaning ϵ is not even the absolute value of a polynomial.

We will resolve this issue by focusing on the z for which the term in the absolute value is already positive. The hope would be that we can then remove the absolute value to get a polynomial. The caveat is that which z correspond to positive terms will also be a function of λ. Therefore, the set that gets summed over will not be fixed, meaning the sum is not guaranteed to be a polynomial.

The solution is to introduce another parameter λ_0. We will consider the sum over z that give positive terms for this particular λ_0, but then consider the sum as a function of an independent parameter λ. If we remove the absolute values, we get a polynomial as desired. We then analyze this polynomial, and at the end set $\lambda = \lambda_0$ to get the desired result.

In more detail, let $\mathcal{Z}_\lambda$ be the set of z such that z is a more likely output under $E_{1/\lambda}$ than E_∞. That is, $\Pr_{H\leftarrow E_{1/\lambda}}[A^H() = z] > \Pr_{H\leftarrow E_\infty}[A^H() = z]$. Then we have that

$$\left(\Pr_{H\leftarrow E_{1/\lambda}}[A^H() \in \mathcal{Z}_\lambda] - \Pr_{H\leftarrow E_\infty}[A^H() \in \mathcal{Z}_\lambda]\right) = -\left(\Pr_{H\leftarrow E_{1/\lambda}}[A^H() \notin \mathcal{Z}_\lambda] - \Pr_{H\leftarrow E_\infty}[A^H() \notin \mathcal{Z}_\lambda]\right)$$

since the terms in parentheses sum to 0. On the other hand, the absolute values of the terms in parentheses sum to ϵ. Therefore, we have that

$$\Pr_{H\leftarrow E_{1/\lambda}}[A^H() \in \mathcal{Z}_\lambda] - \Pr_{H\leftarrow E_\infty}[A^H() \in \mathcal{Z}_\lambda] = \epsilon(\lambda)/2.$$

Fix λ_0, and consider the quantity

$$p_{\lambda_0}(\lambda) \equiv \Pr_{H\leftarrow E_{1/\lambda}}[A^H() \in \mathcal{Z}_{\lambda_0}] = \sum_{z\in\mathcal{Z}_{\lambda_0}} \Pr_{H\leftarrow E_{1/\lambda}}[A^H() = z].$$

Then $p_\lambda(\lambda) - p_\lambda(0) = \epsilon(\lambda)/2$. Further, for each λ_0, p_{λ_0} is a degree-d polynomial in λ such that $p^{(i)}_{\lambda_0}(0) = 0$ for $i \in \{1,\ldots,\Delta-1\}$. It also lies in the range $[0,1]$ when $\lambda = 0$ or $1/\lambda \in \mathbb{Z}^+$. Next, we will use the following theorem.

Theorem 8.1. *Let $p(\lambda)$ be a polynomial in λ of degree d such that $p^{(i)}(0) = 0$ for $i \in \{1,\ldots,\Delta-1\}$, $0 \le p(0) \le 1$, and $0 \le p(1/r) \le 1$ for all $r \in \mathbb{Z}^+$. Then $|p(1/r) - p(0)| < 2^{1-\Delta}\zeta(2\Delta)(1/r)^\Delta d^{3\Delta}$ for all $r \in \mathbb{Z}^+$, where ζ is the Riemann Zeta function.*

Before proving this theorem, we use it to finish the proof of Theorem 4.2. For each λ_0, p_{λ_0} satisfies the conditions of Theorem 8.1, so we must have that $p_{\lambda_0}(\lambda) - p_{\lambda_0}(0) < 2^{1-\Delta}\zeta(2\Delta)\lambda^\Delta d^{3\Delta}$. But then setting $\lambda_0 = \lambda$, we get that

$$\epsilon(\lambda) = 2(p_\lambda(\lambda) - p_\lambda(0)) < 2^{2-\Delta}\zeta(2\Delta)\lambda^\Delta d^{3\Delta}.$$

Replacing $1/\lambda$ with r, we have shown that the output distributions of any q query quantum algorithm A under E_r and E_∞ are $2^{2-\Delta}\zeta(2\Delta)(1/r)^\Delta d^{3\Delta}$-close, as desired.

Proof of Theorem 8.1. We have a polynomial p of degree d with $p^{(i)}(0) = 0$ for $i \in \{1,\ldots,\Delta-1\}$. Further, for $r \in \mathbb{Z}^+ \cap \{\infty\}$, $0 \le p(1/r) \le 1$. Our goal is to show that $p(\lambda)$ approaches $p(0)$ reasonably fast as $\lambda \to 0$, using the fact that $p(\lambda)$ is constrained to lie in $[0,1]$ on all reciprocals of integers. Our strategy, roughly, is to interpolate p, using special interpolation points that are reciprocals of integers. This gives an expression for p in terms of Lagrange polynomials and the values of p at the interpolation points. We can bound the interpolation points to be in $[0,1]$ and then analyze the rate the Lagrange polynomials approach 0 to obtain the desired bounds.

The proof is rather technical. Our proof will gradually reduce the task of proving Theorem 8.1 to smaller and smaller tasks. We now give the proof.

Let $s(\lambda) = \frac{p(\lambda)-p(0)}{\lambda^\Delta}$. Then s is a degree $(d-\Delta)$ polynomial. We will now interpolate this polynomial at $d-\Delta+1$ points: let

$$\lambda_i = \frac{1}{\left\lfloor \frac{(d-\Delta+1)^3}{2i^2} \right\rfloor}.$$

Then we can use the Lagrange interpolating polynomials to interpolate $s(\lambda)$. Let $s_i = s(\lambda_i)$. Then,

$$s(\lambda) = \sum_{i=1}^{d-\Delta+1} s_i \ell_i(\lambda),$$

where $\ell_i(\lambda)$ is the Lagrange polynomial

$$\ell_i(\lambda) = \prod_{j=1, j\neq i}^{d-\Delta+1} \left(\frac{\lambda - \lambda_j}{\lambda_i - \lambda_j}\right).$$

Then we get

$$\begin{aligned} p(\lambda) - p(0) &= \lambda^\Delta \sum_{i=1}^{d-\Delta+1} \frac{p(\lambda_i) - p(0)}{\lambda_i^\Delta} \ell_i(\lambda) \\ &= \sum_{i-1}^{d-\Delta+1} a_i(\lambda)(p(\lambda_i) - p(0)), \end{aligned}$$

where

$$a_i(\lambda) = \left(\frac{\lambda}{\lambda_i}\right)^\Delta \ell_i(\lambda).$$

Now, observe that $1/\lambda_i$ are integers, so $0 \leq p(\lambda_i) \leq 1$ by assumption. Since $0 \leq p(0) \leq 1$ as well, we must have that $|p(\lambda_i) - p(0)| \leq 1$. Therefore,

$$|p(\lambda) - p(0)| \leq \sum_{i=1}^{d-\Delta+1} |a_i(\lambda)|.$$

We now need to bound this sum.

Claim 6. *If $\lambda \leq \lambda_i$ for all i, then $\sum_i |a_i(\lambda)| < 2^{1-\Delta}\zeta(2\Delta)\lambda^\Delta d^{3\Delta}$.*

Before proving this claim, we note that it proves Theorem 8.1 when $\lambda \leq \lambda_i$ for all i (equivalently, $\lambda \leq \lambda_1$). If $\lambda > \lambda_1$, then the bound we are trying to prove is at least

$$2^{1-\Delta}\zeta(2\Delta)\left(\frac{(d-\Delta+1)^3}{\lfloor (d-\Delta+1)^3/2 \rfloor}\right)^\Delta > 2\zeta(2\Delta) > 2.$$

This is already trivially satisfied by the assumption that $p(1/r) \in [0, 1]$. □

Proof of Claim 6. First, notice that

$$\left|\frac{a_i(\lambda)}{\lambda^\Delta}\right| = \left(\frac{1}{\lambda_i}\right)^\Delta \prod_{j=1, j\neq i}^{d-\Delta+1} \left(\frac{\left|\lambda - \lambda_j\right|}{\left|\lambda_i - \lambda_j\right|}\right) \leq \left(\frac{1}{\lambda_i}\right)^\Delta \prod_{j=1, j\neq i}^{d-\Delta+1} \left(\frac{\lambda_j}{\left|\lambda_i - \lambda_j\right|}\right),$$

where the inequality follows from our assumption that $0 \leq \lambda \leq \lambda_j \forall j$, meaning that $|\lambda - \lambda_j| \leq \lambda_j$. Now, observe that $\lambda_i \geq \frac{2i^2}{(d-\Delta+1)^3}$ and that

$$\begin{aligned} \left|\lambda_i - \lambda_j\right| &= \lambda_i\lambda_j \left|\left\lfloor \frac{(d-\Delta+1)^3}{2i^2} \right\rfloor - \left\lfloor \frac{(d-\Delta+1)}{2j^2} \right\rfloor\right| \\ &\geq \frac{2i^2}{(d-\Delta+1)^3}\lambda_j \left(\left|\frac{(d-\Delta+1)^3}{2i^2} - \frac{(d-\Delta+1)^3}{2j^2}\right| - 1\right), \end{aligned}$$

which can be simplified to

$$|\lambda_i - \lambda_j| \geq \lambda_j \frac{\left|i^2 - j^2\right| - \frac{2i^2j^2}{(d-\Delta+1)^3}}{j^2}.$$

We notice that the numerator is minimized by making i and j as large as possible, which is when they are $d-\Delta+1$ and $d-\Delta$. In this case, the quantity becomes $\lambda_j(3-\frac{2}{d-\Delta+1})/j^2$, which is greater than 0 as long as $d-\Delta+1 \geq 1$ (if $d-\Delta<0$, then $p(\lambda)$ is a constant, so the theorem is trivial). Thus,

$$\left|\frac{a_i(\lambda)}{\lambda^\Delta}\right| \leq \left(\frac{1}{\lambda_i}\right)^\Delta \prod_{j=1,j\neq i}^{d-\Delta+1}\left(\frac{j^2}{|i^2-j^2|-\frac{2i^2j^2}{(d-\Delta+1)^3}}\right).$$

The $(1/\lambda_i)^\Delta$ term is bounded by $(\frac{(d-\Delta+1)^3}{2i^2})^\Delta$. We now bound the other term.

Claim 7. *For all integers D and i such that $i \leq D$, we have $\alpha_{D,i} \leq 2$ where*

$$\alpha_{D,i} = \prod_{j=1,j\neq i}^{D}\left(\frac{j^2}{|i^2-j^2|-\frac{2i^2j^2}{D^3}}\right).$$

Before proving Claim 7, we demonstrate how it proves Claim 6. Claim 7 gives

$$\left|\frac{a_i(\lambda)}{\lambda^\Delta}\right| \leq \left(\frac{1}{\lambda_i}\right)^\Delta \prod_{j=1,j\neq i}^{d-\Delta+1}\left(\frac{j^2}{|i^2-j^2|-\frac{2i^2j^2}{(d-\Delta+1)^3}}\right) \leq \left(\frac{(d-\Delta+1)^3}{2i^2}\right)^\Delta \times 2 \leq 2\left(\frac{d^3}{2i^2}\right)^\Delta.$$

This gives

$$|a_i(\lambda)| \leq \lambda^\Delta d^{3\Delta} 2^{1-\Delta}\frac{1}{i^{2\Delta}}.$$

Summing over all i from 1 to $d-\Delta+1$ gives

$$\sum_{i=1}^{d-\Delta+1}|a_i(\lambda)| \leq \lambda^\Delta d^{3\Delta}2^{1-\Delta}\sum_{i=1}^{d-\Delta+1}\frac{1}{i^{2\Delta}}.$$

The sum on the right-hand side is the truncated p series for $p=2\Delta$. This series sums to $\zeta(p)$ where ζ is the Riemann Zeta function, so the truncation is strictly less than this value. Therefore,

$$\sum_{i=1}^{d-\Delta+1}|a_i(\lambda)| < \lambda^\Delta d^{3\Delta}2^{1-\Delta}\zeta(2\Delta),$$

as desired. □

It now remains to prove Claim 7.

Proof of Claim 7. First, rewrite $\alpha_{D,i}$ as

$$\alpha_{D,i} = \prod_{j=1,j\neq i}^{D}\frac{j^2}{|i^2-j^2|}\prod_{j=1,j\neq i}^{D}\frac{1}{1-\frac{2i^2j^2}{|i^2-j^2|D^3}} = \beta_{D,i}\gamma_{D,i},$$

where

$$\beta_{D,i} = \prod_{j=1,j\neq i}^{D}\frac{j^2}{|i^2-j^2|} \text{ and } \gamma_{D,i} = \prod_{j=1,j\neq i}^{D}\frac{1}{1-\frac{2i^2j^2}{|i^2-j^2|D^3}}.$$

We first bound $\beta_{D,i}$:

$$\begin{aligned}
\beta_{D,i} &= \prod_{j=1, j\neq i}^{D} \frac{j^2}{|i^2 - j^2|} \\
&= \prod_{j=1}^{i-1} \frac{j^2}{(i+j)(i-j)} \prod_{j=i+1}^{D} \frac{j^2}{(i+j)(j-i)} \\
&= \left(\frac{((i-1)!)^2}{\frac{(2i-1)!}{i!}(i-1)!}\right)\left(\frac{\left(\frac{D!}{i!}\right)^2}{\frac{(D+i)!}{(2i)!}(D-i)!}\right) \\
&= \left(\frac{2(i!)^2}{(2i)!}\right)\left(\frac{(D!)^2(2i)!}{(i!)^2(D-i)!(D+i)!}\right) \\
&= 2(D!)^2/(D-i)!(D+i)! = 2\frac{\binom{2D}{D+i}}{\binom{2D}{D}}.
\end{aligned}$$

Now we compute the Taylor series expansion of $\ln\left(\frac{\beta_{D,i}}{2}\right) = \ln\binom{2D}{D+i} - \ln\binom{2D}{D}$ around $i = 0$:

$$\ln\left(\frac{\beta_{D,i}}{2}\right) = \sum_{m=1}^{\infty} \frac{i^m}{m!} \left(\left(\frac{d}{dj}\right)^m \ln\binom{2D}{D+j}\right)\Bigg|_{j=0}.$$

Next, we will need to use the polygamma function $\psi^{(m)}(x) = \left(\frac{d}{dx}\right)^m \ln\Gamma(x)$ where $\Gamma(x)$ is the Gamma function, $\Gamma(x) = (x-1)!$ for integers x. See [6] for background on Γ and $\psi^{(m)}$. Recalling that $\binom{n}{k} = n!/k!(n-k)!$, we therefore have that

$$\left(\frac{d}{dk}\right)^m \ln\binom{n}{k} = -\psi^{(m-1)}(k+1) - (-1)^m\psi^{(m-1)}(n-k+1).$$

Plugging in $k = n/2$ would give

$$\left(\frac{d}{dk}\right)^m \ln\binom{n}{k}\Bigg|_{k=n/2} = -(1+(-1)^m)\psi^{(m-1)}(n/2+1).$$

This allows us to write

$$\ln\left(\frac{\beta_{D,i}}{2}\right) = -\sum_{m=1}^{\infty}(1+(-1)^m)\psi^{(m-1)}(D+1)\frac{i^m}{m!} = -2\sum_{\ell=1}^{\infty}\psi^{(2\ell-1)}(D+1)\frac{i^{2\ell}}{(2\ell)!}.$$

Next, we discard all terms in the sum except $\ell = 1$. We use that $\psi^{(m)}(x) \geq 0$ for all odd m and non-negative x to conclude that this only increases the right-hand side above. Then, we use that $\psi^{(1)}(x+1) \geq \frac{1}{x} - \frac{1}{2x^2}$ for $x > 1$ to bound

$$\ln\left(\frac{\beta_{D,i}}{2}\right) \leq -\psi^{(1)}(D+1)i^2 \leq -i^2\left(\frac{1}{D} - \frac{1}{2D^2}\right).$$

We now bound the second term $\gamma_{D,i} = \prod_{j=1, j\neq i}^{D} \frac{1}{1-\frac{2i^2j^2}{|i^2-j^2|D^3}}$. First, let $x_{i,j} = \frac{2i^2j^2}{|i^2-j^2|D^3}$. Notice that $x_{i,j} = x_{j,i}$, and that for $j < i$, $x_{i,j} \leq x_{i,i-1}$. Lastly, notice that $x_{i,i-1} \leq x_{D,D-1}$. Therefore, for all i, j,

$$x_{i,j} \leq x_{D,D-1} = \frac{2(D-1)^2}{D(2D-1)} = 1 - \frac{1}{D} - \frac{D-1}{D(2D-1)} \leq 1 - \frac{1}{D}.$$

Next, we need the following claim:

Claim 8. *For $D \geq 4$ and $x \leq 1 - 1/D$, $\ln(\frac{1}{1-x}) \leq x + x^2 \ln D$.*

Proof. Let $z(x) = \frac{1}{x^2}\ln(\frac{1}{1-x}) - \frac{1}{x}$. This function has the Taylor series $\sum_{\ell=0}^{\infty} x^\ell/(\ell+2)$. In particular, z monotonically increases with x. Therefore, it suffices to show that $z(1-1/D) \leq \ln D$. Evaluating $z(1-1/D)$ gives

$$z(1-1/D) = D\frac{D\ln D - D + 1}{(D-1)^2} = \ln D - \frac{D(D-1) - (2D-1)\ln D}{(D-1)^2}.$$

Therefore, we just need to show that $D(D-1) \geq (2D-1)\ln D$. Toward that end, note that $(2D-1)\ln D \leq 2D\ln D$. Therefore, all we need is $D - 1 \geq 2\ln D$, which is true for $D \geq 4$. □

We now return to the proof of Claim 7. For now, we assume $D \geq 4$. Then we have that

$$\ln \gamma_{D,i} \leq \sum_{j=1, j\neq i}^{D} x_{i,j} + x_{i,j}^2 \ln D.$$

We now bound the sums $\sum_{j=1,j\neq i}^{D} x_{i,j}$ and $\sum_{j=1,j\neq i}^{D} x_{i,j}^2$. We have that

$$\begin{aligned}
\sum_{j=1,j\neq i}^{D} x_{i,j} &= \sum_{j=1,j\neq i}^{D} \frac{2i^2j^2}{|i^2-j^2|D^3} = \frac{i^2}{D^3}\left(\sum_{j=1}^{i-1}\frac{2j^2}{(i^2-j^2)} + \sum_{j=i+1}^{D}\frac{2j^2}{(j^2-i^2)}\right)\\
&= \frac{i^2}{D^3}\left(\sum_{j=1}^{i-1}\left(-2 + \frac{i}{(i-j)} + \frac{i}{(i+j)}\right) + \sum_{j=i+1}^{D}\left(2 + \frac{i}{(j-i)} - \frac{i}{(i+j)}\right)\right)\\
&= \frac{i^2}{D^3}\left(-2(i-1) + i\sum_{j_1=1}^{i-1}\frac{1}{j_1} + i\sum_{j_2=i+1}^{2i-1}\frac{1}{j_2} + 2(D-i) + i\sum_{j_3=1}^{D-i}\frac{1}{j_3} - i\sum_{j_4=2i+1}^{D+i}\frac{1}{j_4}\right),
\end{aligned}$$

where in the last line we used the substitutions $j_1 = i - j, j_2 = j + i, j_3 = j - i, j_4 = j + i$. Now we rewrite $\sum_{j_1=1}^{i-1}\frac{1}{j_1} + \sum_{j_2=i+1}^{2i-1}\frac{1}{j_2}$ as $\sum_{j=1}^{2i}\frac{1}{j} - \frac{3}{2i}$. Now we can rearrange the sums to give

$$\begin{aligned}
\sum_{j=1,j\neq i}^{D} x_{i,j} &= \frac{i^2}{2D^3}\left(1 + 4D - 8i + i\left(\sum_{j=1}^{D-i}\frac{1}{j} - \sum_{j=1}^{D+i}\frac{1}{j} + 2\sum_{j=1}^{2i}\frac{1}{j}\right)\right)\\
&= \frac{i^2}{2D^3}(1 + 4D - 8i + i(H_{D-i} - H_{D+i} + 2H_{2i})),
\end{aligned}$$

where $H_n = \sum_{j=1}^{n} j^{-1}$ are the Harmonic numbers. Using a similar analysis, we can write

$$\sum_{j=1,j\neq i}^{D} x_{i,j}^2 = \frac{i^4}{4D^6}\left(-3 + 16D + 12i(H_{D-i} - H_{D+i}) + 4i^2\left(H_{D-i}^{(2)} + H_{D+i}^{(2)}\right)\right),$$

Here, $H_n^{(2)} = \sum_{j=1}^{n} j^{-2}$ are generalized Harmonic numbers.

This gives us upper bounds on $\ln \gamma_{D,i}$ and $\ln(\beta_{D,i}/2)$, which we put together to get an upper bound on $\ln(\alpha_{D,i}/2)$ for $D \geq 4$:

$$\begin{aligned}\ln\left(\frac{\alpha_{D,i}}{2}\right) &\leq \frac{i^2}{2D^3}(1 + 5D - 2D^2 - 8i + 2i(H_{D-i} - H_{D+i} + 2H_{2i})) \\ &\quad + \frac{i^4 \ln D}{4D^6}\left(-3 + 16D + 12i(H_{D-i} - H_{D+i}) + 4i^2\left(H^{(2)}_{D-i} + H^{(2)}_{D+i}\right)\right) \\ &= \frac{i^2}{4D^6}\left[\left(4D^3 i(H_{D-i} - H_{D+i} + 2H_{2i}) + 10D^4 - 16D^3 i + 4i^4\left(H^{(2)}_{D-i} + H^{(2)}_{D+i}\right)\ln D\right)\right. \\ &\quad \left. -4D^5 + \left(2D^3 + 16Di^2 \ln D + 12i^3(H_{D-i} - H_{D+i})\ln D\right) - 3i^2 \ln D\right].\end{aligned}$$

In order to prove Claim 7, we need to show that the quantity $\ln(\alpha_{D,i}/2)$ on the left is less than 0. We can already see that, for large enough D, since $1 \leq i \leq D$ and $H_n = O(\log n)$ and $H^{(2)}_n = O(1)$, that the $-4D^5$ term will dominate, and therefore the whole quantity will indeed be less than 0. This shows that the claim holds for sufficiently large D, and therefore that there is *some* constant bounding $\alpha_{D,i}$. This is enough to conclude a weaker version of Theorem 4.5 where the given constant is replaced by the existence of *some* constant.

The remaining proof will give an explicit threshold for D, after which the expression above is indeed less than 0. For all smaller D, we will explicitly calculate the value of $\alpha_{D,i}$ in order to complete the proof.

We first look at the quartic term of the right side (where powers of i and D sum to 4):

$$4D^3 i(H_{D-i} - H_{D+i} + 2H_{2i}) + 10D^4 - 16D^3 i + 4i^4(H^{(2)}_{D-i} + H^{(2)}_{D+i})\ln D.$$

Let $r = i/D$. We use the fact that $\log n \leq H_n \leq \log n + 1$ and $H^{(2)}_n \leq \pi^2/6$ to bound this above by

$$D^4\left(4r(3 + \ln(1-r) - \ln(1+r) + 2\ln(2r) + 2\ln(2D)) + 10 - 16r + \frac{4}{3}\pi^2 r^4 \ln D\right).$$

Rearranging gives

$$D^4\left(4r\left(-1 + \ln\left(\frac{16(1-r)r^2}{1+r}\right) + 2\ln D\right) + 10 + \frac{4}{3}\pi^2 r^4 \ln D\right).$$

In the range $r \in [0,1]$, $16(1-r)r^2/(1+r)$ obtains a maximum value of $8(5\sqrt{5} - 11) < 1.443 < 1.649 < e^{1/2}$. This means $\ln(\frac{8(1-r)r^2}{1+r}) < \frac{1}{2}$. We also use the fact that $\pi^2 \approx 9.87 < 99/10$. Therefore, we upper bound the quantity as

$$D^4\left(\left(8 + \frac{4}{3}\pi^2\right)\ln D + 8\right) \leq D^4\left(\frac{106}{5}\ln D + 8\right).$$

We now bound the cubic term:

$$2D^3 + 16Di^2 \ln D + 12i^3(H_{D-i} - H_{D+i})\ln D \leq D^3(2 + 16r^2 \ln D + 12r^3(1 + \ln(1-r) - \ln(1+r))\ln D).$$

Using Taylor expansions, we see that $1 + \ln(1-r) - \ln(1+r) \leq 1 - 2r - \frac{2}{3}r^3$, allowing us to bound this as

$$2D^3(1 + 2r^2(4 + 3r - 6r^2 - 2r^4)\ln D).$$

$2r^2(4 + 3r - 6r^2 - 2r^4)$ obtains a maximum value less than 3, meaning we can bound this as $2D^3(1 + 3\ln D)$.

Putting this all together, we have that, for $D \geq 4$,

$$\ln\left(\frac{\alpha_{D,i}}{2}\right) \leq \frac{r^2}{4D^2}\left(-4D^3 + D^2\left(\frac{106}{5}\ln D + 8\right) + 2D(3\ln D + 1)\right).$$

Table 1. The Values of $\alpha_{D,i}$ for $D < 18$, Rounded to the Nearest 0.01

	i													
D	1	2	3	4	5	6	7	8	9	10	11	12	13	14
17	1.9	1.64	1.27	0.9	0.58	0.34	0.18	0.09	0.04	0.01	0.01	0.00	0.00	0.00
16	1.9	1.62	1.25	0.87	0.54	0.31	0.16	0.07	0.03	0.01	0.00	0.00	0.00	0.00
15	1.89	1.6	1.21	0.83	0.51	0.28	0.14	0.06	0.02	0.01	0.00	0.00	0.00	0.00
14	1.89	1.58	1.18	0.79	0.47	0.25	0.12	0.05	0.02	0.01	0.00	0.00	0.00	0.00
13	1.88	1.56	1.15	0.75	0.44	0.23	0.10	0.04	0.02	0.00	0.00	0.00	0.00	
12	1.87	1.53	1.11	0.71	0.40	0.20	0.09	0.03	0.01	0.00	0.00	0.00		
11	1.86	1.51	1.07	0.66	0.36	0.17	0.07	0.03	0.01	0.00	0.00			
10	1.85	1.48	1.02	0.61	0.32	0.15	0.06	0.02	0.01	0.00				
9	1.84	1.45	0.97	0.57	0.28	0.12	0.05	0.02	0.00					
8	1.83	1.41	0.92	0.52	0.25	0.10	0.04	0.00						
7	1.82	1.37	0.87	0.47	0.22	0.10	0.01							
6	1.81	1.33	0.82	0.43	0.22	0.01								
5	1.79	1.30	0.79	0.46	0.04									
4	1.79	1.29	0.86	0.10										
3	1.82	1.43	0.23											
2	**2**	**0.5**												
1	**1**													

	i		
D	15	16	17
17	0.00	0.00	0.00
16	0.00	0.00	
15	0.00		

[1] Values in bold are exact.

Define the function

$$f(D) = -4D^3 + D^2\left(\frac{106}{5}\ln D + 8\right) + 2D(3\ln D + 1),$$

so that $\ln(\frac{\alpha_{D,i}}{2}) \leq \frac{r^2}{4D^2} f(D)$. We now show that $f(D) < 0$ for $D \geq 18$. We have the following derivatives:

$$f'(D) = \frac{-60D^2 + D(186 + 212\ln D) + (30\ln D + 40)}{5},$$
$$f''(D) = \frac{-120D^2 + D(212\ln D + 398) + 30}{5D},$$
$$f'''(D) = \frac{-120D^2 + 212D - 30}{5D^2}.$$

Notice that $-120D^2 + 212D - 30 = 0$ at $\frac{1}{60}(53 \pm \sqrt{1909})$, which is approximately $0.155, 1.612$. For D between these two values, $-120D^2 + 212D - 30$, and hence $f'''(D)$, is greater than zero. For D outside of these values (in particular, for $D \geq 2$), $f'''(D) < 0$.

Now notice that $f''(7) \approx -5.04 < 0$. Since $f'''(D) < 0$ for $D \geq 2$, this means $f''(D) < 0$ for $D \geq 7$. Next, notice that $f'(13) \approx -107.2 < 0$. Since $f''(D) < 0$ for $D \geq 8$, this means $f'(D) < 0$ for $D \geq 13$. Finally, notice that $f(18) \approx -534.5 < 0$. Since $f'(D) < 0$ for $D \geq 13$, we therefore have that $f(D) < 0$ for $D \geq 18$.

Therefore, we have proved $\alpha_{D,i} \leq 2$ for $D \geq 18$. The remaining 153 cases for $D < 18$ can easily (though tediously) be verified manually. For completeness, the values are included in Table 1. This completes the proof of Claim 7. □

9 A QUANTUM DISTINGUISHER FOR SMALL-RANGE DISTRIBUTIONS

In this section, we give a quantum distinguisher that distinguishes $\mathrm{SR}_r^{\mathcal{Y}}$ from a random function with probability (asymptotically) matching the bound of Corollary 4.4. Our algorithm is basically the collision finding algorithm of Brassard et al. [9], with a check at the end to verify that a collision is found. The algorithm has oracle access to a function O from $\mathcal{X}$ to $\mathcal{Y}$, which is either $\mathrm{SR}_r^{\mathcal{Y}}$ or a random function. It is given as input the integer r, the number of queries q, and operates as follows:

- Let $p = (q-1)/2$. Pick a set S of p points in $\mathcal{X}$ at random, and check that there is no collision on S by making p classical queries to O. Sort the elements of S, and store the pairs $(s, O(s))$ as a table for efficient lookup.
- Construct the oracle $O'(x) = \begin{cases} 1 & \text{if } x \notin S \text{ and } O(x) = O(s) \text{ for some } s \in S \\ 0 & \text{otherwise.} \end{cases}$
- Run Grover's algorithm [27] on O' for p iterations to look for a point x such that $O'(x) = 1$.
- Check that there is an $s \in S$ such that $O(x) = O(s)$ by making one more classical query to O.

Before analyzing this construction, we explain what Grover's algorithm does. It takes as input an oracle O' mapping some space $\mathcal{X}$ into $[2]$, and tries to find an x such that $O'(x) = 1$. Specifically, if N points map to 1, then after q queries to O', Grover's algorithm will output an x such that $O'(x) = 1$ with probability $\Theta(q^2 N/|\mathcal{X}|)$.

We now analyze this construction. The first step takes p queries to O. If we find a collision, we are done. Otherwise, we have p points that map to p different values. Call this set of values T. The oracle O' outlined in the second step makes exactly one query to O for each query to O'. The number of points in x such that $O'(x) = 1$ is the number of points x in $\mathcal{X} \setminus S$ (which is $|X| - p$) such that $O(x) \in T$. In the random oracle case, the probability that $O(x)$ is one of p random values is $p/|\mathcal{Y}|$, so the expected number of such x is $(|\mathcal{X}| - p)p/|\mathcal{Y}|$. Thus, after p iterations, Grover's algorithm will output such an x with probability $\Theta(p^3(|\mathcal{X}| - p)/|\mathcal{X}||\mathcal{Y}|))$. In the $\mathrm{SR}_r^{\mathcal{Y}}$ case, since there are only r possible outputs, the probability that x maps to T is p/r, so the expected number of such x is $p(|\mathcal{X}| - p)/r$. Thus, Grover's algorithm will output such an x with probability $\Theta(p^3(|\mathcal{X}| - p)/r|\mathcal{X}|)$.

The difference in these probabilities is $\Theta((p^3(1/r - 1/|\mathcal{Y}|)(|\mathcal{X}| - p)/|\mathcal{X}|)$. If we let $|\mathcal{Y}|$ be at least $2r$ and $|\mathcal{X}|$ at least $2p + 1 = q$, we see that we distinguish $\mathrm{SR}_r^{\mathcal{Y}}$ from random with probability $\Omega(p^3/r) = \Omega(q^3/r)$, thus matching the bound of Corollary 4.4. This shows that the corollary is optimal, and hence Theorem 4.2 is optimal for the case $\Delta = 0$.

ACKNOWLEDGMENTS

We would like to thank Dan Boneh for his guidance and many insightful discussions. This work was supported by NSF and DARPA. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of Defense or the U.S. Government.

REFERENCES

[1] Scott Aaronson. 2009. Quantum copy-protection and quantum money. In *Proceedings of the 24th Annual IEEE Conference on Computational Complexity (CCC'09)*. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5231194.

[2] Scott Aaronson and Paul Christiano. 2012. Quantum money from hidden subspaces. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing*, Howard J. Karloff and Toniann Pitassi (Eds.). ACM Press, New York, NY, 41–60. https://doi.org/10.1145/2213977.2213983

[3] Gorjan Alagic, Christian Majenz, Alexander Russell, and Fang Song. 2020. Quantum-access-secure message authentication via blind-unforgeability. In *Advances in Cryptology (EUROCRYPT'20), Part III*, Lecture Notes in

Computer Science, Vol. 12107, Anne Canteaut and Yuval Ishai (Eds.). Springer, Heidelberg, Germany, 788–817. https://doi.org/10.1007/978-3-030-45727-3_27

[4] Gorjan Alagic and Alexander Russell. 2017. Quantum-secure symmetric-key cryptography based on hidden shifts. In *Advances in Cryptology (EUROCRYPT'17), Part III*, Lecture Notes in Computer Science, Vol. 10212, Jean-Sébastien Coron and Jesper Buus Nielsen (Eds.). Springer, Heidelberg, Germany, 65–93. https://doi.org/10.1007/978-3-319-56617-7_3

[5] Andris Ambainis, Ansis Rosmanis, and Dominique Unruh. 2014. Quantum attacks on classical proof systems: The hardness of quantum rewinding. In *Proceedings of the 55th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, Philadelphia, PA, 474–483. https://doi.org/10.1109/FOCS.2014.57

[6] Milton Abramowitz and Irene Stegun. 1964. *Handbook of Mathematical Functions, With Formulas, Graphs, and Mathematical Tables*. Dover Publications, Inc.

[7] Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. 1997. Strengths and weaknesses of quantum computing. *SIAM J. Comput.* 26 (1997), 1510–1523. http://arxiv.org/abs/quant-ph/9701001.

[8] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. 2011. Random oracles in a quantum world. In *Advances in Cryptology (ASIACRYPT'11)*, Lecture Notes in Computer Science, Vol. 7073, Dong Hoon Lee and Xiaoyun Wang (Eds.). Springer, Heidelberg, Germany, 41–69. https://doi.org/10.1007/978-3-642-25385-0_3

[9] Gilles Brassard, Peter Høyer, and Alain Tapp. 1997. Quantum algorithm for the collision problem. *ACM SIGACT News (Cryptology Column)* 28 (1997), 14–19. http://arxiv.org/abs/quant-ph/9705002.

[10] Manuel Blum and Russell Impagliazzo. 1987. Generic oracles and oracle classes (extended abstract). In *Proceedings of the 28th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, Los Angeles, CA, 118–126. https://doi.org/10.1109/SFCS.1987.30

[11] Dan Boneh and Richard J. Lipton. 1995. Quantum cryptanalysis of hidden linear functions (extended abstract). In *Advances in Cryptology (CRYPTO'95)*, Lecture Notes in Computer Science, Vol. 963, Don Coppersmith (Ed.). Springer, Heidelberg, Germany, 424–437. https://doi.org/10.1007/3-540-44750-4_34

[12] Dan Boneh, Hart William Montgomery, and Ananth Raghunathan. 2010. Algebraic pseudorandom functions with improved efficiency from the augmented cascade. In *Proceedings of the 17th Conference on Computer and Communications Security (ACM CCS'10)*, Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov (Eds.). ACM Press, Chicago, Illinois, 131–140. https://doi.org/10.1145/1866307.1866323

[13] Dan Boneh. 1998. The decision Diffie-Hellman problem. In *Proceedings of the 3rd Algorithmic Number Theory Symposium (ANTS'98)*, Lecture Notes in Computer Science, Vol. 1423. Springer, Heidelberg, Germany. Invited paper.

[14] Abhishek Banerjee, Chris Peikert, and Alon Rosen. 2012. Pseudorandom functions and lattices. In *Advances in Cryptology (EUROCRYPT'12)* Lecture Notes in Computer Science, Vol. 7237, David Pointcheval and Thomas Johansson (Eds.). Springer, Heidelberg, Germany, 719–737. https://doi.org/10.1007/978-3-642-29011-4_42

[15] Zvika Brakerski and Omri Shmueli. 2019. (Pseudo) random quantum states with binary phase. In *Proceedings of the 17th Theory of Cryptography Conference (TCC'19), Part I*, Lecture Notes in Computer Science, Vol. 11891, Dennis Hofheinz and Alon Rosen (Eds.). Springer, Heidelberg, Germany, 229–250. https://doi.org/10.1007/978-3-030-36030-6_10

[16] Zvika Brakerski and Vinod Vaikuntanathan. 2011. Efficient fully homomorphic encryption from (standard) LWE. In *Proceedings of the 52nd Annual Symposium on Foundations of Computer Science*, Rafail Ostrovsky (Ed.). IEEE Computer Society Press, Palm Springs, CA, 97–106. https://doi.org/10.1109/FOCS.2011.12

[17] Dan Boneh and Mark Zhandry. 2013. Quantum-secure message authentication codes. In *Advances in Cryptology (EUROCRYPT'13)* Lecture Notes in Computer Science, Vol. 7881, Thomas Johansson and Phong Q. Nguyen (Eds.). Springer, Heidelberg, Germany, 592–608. https://doi.org/10.1007/978-3-642-38348-9_35

[18] Dan Boneh and Mark Zhandry. 2013. Secure signatures and chosen ciphertext security in a quantum computing world. In *Advances in Cryptology (CRYPTO'13), Part II*, Lecture Notes in Computer Science, Vol. 8043, Ran Canetti and Juan A. Garay (Eds.). Springer, Heidelberg, Germany, 361–379. https://doi.org/10.1007/978-3-642-40084-1_21

[19] Jan Czajkowski, Andreas Hülsing, and Christian Schaffner. 2019. Quantum indistinguishability of random sponges. In *Advances in Cryptology (CRYPTO'19), Part II*, Lecture Notes in Computer Science, Vol. 11693, Alexandra Boldyreva and Daniele Micciancio (Eds.). Springer, Heidelberg, Germany, 296–325. https://doi.org/10.1007/978-3-030-26951-7_11

[20] Ivan Damgård, Jakob Funder, Jesper Buus Nielsen, and Louis Salvail. 2014. Superposition attacks on cryptographic protocols. In *Proceedings of the 7th International Conference on Information Theoretic Security (ICITS'13)*, Lecture Notes in Computer Science, Vol. 8317, Carles Padró (Ed.). Springer, Heidelberg, Germany, 142–161. https://doi.org/10.1007/978-3-319-04268-8_9

[21] Craig Gentry. 2009. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, Michael Mitzenmacher (Ed.). ACM Press, 169–178. https://doi.org/10.1145/1536414.1536440

[22] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. 1984. How to construct random functions (extended abstract). In *Proceedings of the 25th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, 464–479. https://doi.org/10.1109/SFCS.1984.715949

[23] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. 1986. How to construct random functions. *J. ACM* 33, 4 (Oct. 1986), 792–807.

[24] Tommaso Gagliardoni, Andreas Hülsing, and Christian Schaffner. 2016. Semantic security and indistinguishability in the quantum world. In *Advances in Cryptology (CRYPTO'16), Part III*, Lecture Notes in Computer Science, Vol. 9816, Matthew Robshaw and Jonathan Katz (Eds.). Springer, Heidelberg, Germany, 60–89. https://doi.org/10.1007/978-3-662-53015-3_3

[25] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. 2013. Reusable garbled circuits and succinct functional encryption. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing*, Dan Boneh, Tim Roughgarden, and Joan Feigenbaum (Eds.). ACM Press, 555–564. https://doi.org/10.1145/2488608.2488678

[26] Rishab Goyal, Venkata Koppula, and Brent Waters. 2017. Lockable obfuscation. In *Proceedings of the 58th Annual Symposium on Foundations of Computer Science*, Chris Umans (Ed.). IEEE Computer Society Press, 612–621. https://doi.org/10.1109/FOCS.2017.62

[27] Lov K. Grover. 1996. A fast quantum mechanical algorithm for database search. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*. ACM Press, 212–219. https://doi.org/10.1145/237814.237866

[28] Akinori Hosoyamada and Tetsu Iwata. 2019. 4-round Luby-Rackoff construction is a qPRP. In *Advances in Cryptology (ASIACRYPT'19), Part I* Lecture Notes in Computer Science, Vol. 11921, Steven D. Galbraith and Shiho Moriai (Eds.). Springer, Heidelberg, Germany, 145–174. https://doi.org/10.1007/978-3-030-34578-5_6

[29] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. 1999. A pseudorandom generator from any one-way function. *SIAM J. Comput.* 28, 4 (1999), 1364–1396.

[30] Zhengfeng Ji, Yi-Kai Liu, and Fang Song. 2018. Pseudorandom quantum states. In *Advances in Cryptology (CRYPTO'18), Part III*, Lecture Notes in Computer Science, Vol. 10993, Hovav Shacham and Alexandra Boldyreva (Eds.). Springer, Heidelberg, Germany, 126–152. https://doi.org/10.1007/978-3-319-96878-0_5

[31] Jonathan Katz and Yehuda Lindell. 2014. *Introduction to Modern Cryptography* (2nd ed.). Chapman & Hall/CRC.

[32] Allison B. Lewko and Brent Waters. 2009. Efficient pseudorandom functions from the decisional linear assumption and weaker variants. In *Proceedings of the 16th Conference on Computer and Communications Security (ACM CCS'09)*, Ehab Al-Shaer, Somesh Jha, and Angelos D. Keromytis (Eds.). ACM Press, 112–120. https://doi.org/10.1145/1653662.1653677

[33] Michael A. Nielsen and Isaac Chuang. 2000. Quantum computation and quantum information. *Am. J. Phys.* 70, 5 (2000), 558. https://doi.org/10.1119/1.1463744

[34] Moni Naor and Omer Reingold. 1995. Synthesizers and their application to the parallel construction of pseudo-random functions. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, 170–181. https://doi.org/10.1109/SFCS.1995.492474

[35] Moni Naor and Omer Reingold. 1997. Number-theoretic constructions of efficient pseudo-random functions. In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, 458–467. https://doi.org/10.1109/SFCS.1997.646134

[36] Moni Naor, Omer Reingold, and Alon Rosen. 2000. Pseudo-random functions and factoring (extended abstract). In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*. ACM Press, 11–20. https://doi.org/10.1145/335305.335307

[37] Oded Regev. 2005. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, Harold N. Gabow and Ronald Fagin (Eds.). ACM Press, 84–93. https://doi.org/10.1145/1060590.1060603

[38] Alexander A. Razborov and Steven Rudich. 1994. Natural proofs. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*. ACM Press, 204–213. https://doi.org/10.1145/195058.195134

[39] Peter W. Shor. 1994. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society Press, 124–134. https://doi.org/10.1109/SFCS.1994.365700

[40] Fang Song and Aaram Yun. 2017. Quantum security of NMAC and related constructions—PRF domain extension against quantum attacks. In *Advances in Cryptology (CRYPTO'17), Part II*, Lecture Notes in Computer Science, Vol. 10402, Jonathan Katz and Hovav Shacham (Eds.). Springer, Heidelberg, Germany, 283–309. https://doi.org/10.1007/978-3-319-63715-0_10

[41] Philipp Woelfel. 1999. Efficient strongly universal and optimally universal hashing. In *Proceedings of the 24th International Symposium on Mathematical Foundations of Computer Science (MFCS'99)*, Lecture Notes in Computer Science, Vol. 1672. 262–272. https://doi.org/10.1007/3-540-48340-3_24

[42] Daniel Wichs and Giorgos Zirdelis. 2017. Obfuscating compute-and-compare programs under LWE. In *Proceedings of the 58th Annual Symposium on Foundations of Computer Science*, Chris Umans (Ed.). IEEE Computer Society Press, 600–611. https://doi.org/10.1109/FOCS.2017.61

[43] Mark Zhandry. 2012. Secure identity-based encryption in the quantum random oracle model. In *Advances in Cryptology (CRYPTO'12)*, Lecture Notes in Computer Science, Vol. 7417, Reihaneh Safavi-Naini and Ran Canetti (Eds.). Springer, Heidelberg, Germany, 758–775. https://doi.org/10.1007/978-3-642-32009-5_44

[44] Mark Zhandry. 2015. A note on the quantum collision and set equality problems. *Quant, Inf, Comput,* 15, 7& 8 (2015). Full version available at http://arxiv.org/abs/1312.1027.

[45] Mark Zhandry. 2016. The Magic of ELFs. Cryptology ePrint Archive, Report 2016/114. https://eprint.iacr.org/2016/114.

[46] Mark Zhandry. 2016. A Note on Quantum-Secure PRPs. Cryptology ePrint Archive, Report 2016/1076. https://eprint.iacr.org/2016/1076.

Received August 2018; revised September 2020; accepted February 2021