Quantum Depth in the Random Oracle Model

Quantum Depth in the Random Oracle Model

Atul Singh Arora,[1] Andrea Coladangelo,[2] Matthew Coudron,[3]
Alexandru Gheorghiu,[4] Uttam Singh,[5] and Hendrik Waldner[6]

Abstract

We give a comprehensive characterization of the computational power of shallow quantum circuits combined with classical computation. Specifically, for classes of *search problems*, we show that the following statements hold, relative to a *random oracle*:

(a) $\mathsf{BPP}^{\mathsf{QNC}^{\mathsf{BPP}}} \neq \mathsf{BQP}$. This refutes Jozsa's conjecture [Joz05] in the random oracle model. As a result, this gives the first *instantiatable* separation between the classes by replacing the oracle with a cryptographic hash function, yielding a resolution to one of Aaronson's ten semi-grand challenges in quantum computing [Aar05].

(b) $\mathsf{BPP}^{\mathsf{QNC}} \nsubseteq \mathsf{QNC}^{\mathsf{BPP}}$ and $\mathsf{QNC}^{\mathsf{BPP}} \nsubseteq \mathsf{BPP}^{\mathsf{QNC}}$. This shows that there is a subtle interplay between classical computation and shallow quantum computation. In fact, for the second separation, we establish that, for some problems, the ability to perform adaptive measurements in a *single* shallow quantum circuit, is more useful than the ability to perform *polynomially many* shallow quantum circuits without adaptive measurements.

(c) There exists a 2-message *proof of quantum depth* protocol. Such a protocol allows a classical verifier to efficiently certify that a prover must be performing a computation of some minimum quantum depth. Our proof of quantum depth can be instantiated using the recent proof of quantumness construction by Yamakawa and Zhandry [YZ22].

[1] Institute for Quantum Information and Matter, California Institute of Technology
Department of Computing and Mathematical Sciences, California Institute of Technology
atul.singh.arora@gmail.com

[2] University of California, Berkeley
Simons Institute for the Theory of Computing
andrea.coladangelo@gmail.com

[3] National Institute of Standards and Technology (NIST)/ Joint Center for Quantum Information and Computer Science (QuICS)
Department of Computer Science, University of Maryland
mcoudron@umd.edu

[4] Department of Computer Science and Engineering, Chalmers University of Technology
Institute for Theoretical Studies, ETH Zürich
alexandru.gheorghiu@chalmers.se

[5] Center for Theoretical Physics, Polish Academy of Sciences
uttam@cft.edu.pl

[6] Department of Computer Science, University of Maryland
Max Planck Institute for Security and Privacy
hwaldner@umd.edu

Contents

1 Introduction

High depth circuits are believed to be strictly more powerful than low depth circuits, in the sense that having deeper circuits allows one to solve a larger set of problems. Indeed, this is a well established fact for both classical and quantum circuits of depth sub-logarithmic in the size of the input [FSS84; Has86; BGK18; WKST19]. However, for circuits of (poly)logarithmic depth and general polynomial depth, proving any sort of *unconditional* separation is challenging [RR94]. In fact, there is not even an unconditional proof that the set of problems that can be solved by polylog-depth classical circuits, NC, is a *strict subset* of the set of problems solvable by poly-depth classical circuits, P (or BPP when allowing for randomness). The same is believed to be the case for the quantum analogues of these classes, QNC and BQP, respectively. Nevertheless, the strict containments $\mathsf{NC} \subsetneq \mathsf{P}$ and $\mathsf{QNC} \subsetneq \mathsf{BQP}$ are known to hold in the oracle setting and, in particular, relative to a *random oracle* [Mil92].[1] This is a strong indication that there are problems in P (BQP) which cannot be parallelized so as to be solvable in NC (QNC). Under the *random oracle heuristic*, by replacing the random oracle with a cryptographic hash function, one can even provide concrete instantiations of such problems. A further indication of the separation between low and high depth computations is provided by certain inherently sequential cryptographic constructions such as time-lock puzzles and verifiable delay functions [RSW96; BBBF18].

The study of circuit depth can also yield insights into the subtle relationship between quantum and classical computation by considering *hybrid circuit models* that combine quantum and classical computation [CCL20; CM20; AGS22; HG22]. In this setting, one can ask the question: how powerful are poly-depth classical circuits, when augmented with polylog-depth quantum circuits? Could it be the case that interspersing BPP with QNC computations captures the full power of BQP computations? Jozsa famously conjectured that the answer is yes [Joz05]. Indeed, there is some evidence to support this conjecture, as the quantum Fourier transform, a central building block for many quantum algorithms, was shown to be implementable with log-depth quantum circuits [CW00]. This also implies that Shor's algorithm can be performed by a $\mathsf{BPP}^{\mathsf{QNC}}$ machine, a polynomial-time classical computer having the ability to invoke a (poly)log depth quantum computer.[2] Moreover, in the oracle setting, a number of problems yielding exponential separations between quantum and classical computation require only constant quantum-depth to solve, providing further support for Jozsa's conjecture [Sim97; Aar10; AA15].

Despite the evidence in support of Jozsa's conjecture, it was recently shown that, in the oracle setting, the conjecture is false [CCL20; CM20]. Specifically, the results of [CCL20] (hereafter referred to as CCL) and [CM20] (hereafter referred to as CM) considered two ways of interspersing poly-depth classical computation with d-depth quantum computation. The first is $\mathsf{BPP}^{\mathsf{QNC_d}}$, denoting problems solvable by a BPP machine that can invoke d-depth quantum circuits (whose outputs are measured in the computational basis). The second, $\mathsf{QNC_d}^{\mathsf{BPP}}$, denotes problems solvable by a d-depth quantum circuit that can invoke a BPP machine at each layer in the computation.[3] Later, borrowing terminology from [CCL20; AGS22], we will refer to the former circuit model as $\mathsf{CQ_d}$ and the latter as $\mathsf{QC_d}$. However, for the purposes of this introduction, we will stick to the more familiar notation using complexity classes. Intuitively, $\mathsf{BPP}^{\mathsf{QNC_d}}$ captures the setting of a classical computer that can invoke a d-depth quantum computater several times. Examples of this include quantum machine learning algorithms such as VQE or QAOA [PMS+14; FGG14], though as mentioned, Shor's algorithm is also of this type. On the other hand, $\mathsf{QNC_d}^{\mathsf{BPP}}$ captures a d-depth *measurement-based quantum computation* [RB01; BBD+09], where intermediate measurements are performed after each layer in the quantum computation. The outcomes of those measurements are processed by a poly-depth classical computation and the results are "fed" into the next quantum layer. CCL and CM showed that there exists an oracle relative to which $\mathsf{BPP}^{\mathsf{QNC_d}} \cup \mathsf{QNC_d}^{\mathsf{BPP}} \subsetneq \mathsf{BQP}$, for any $d = \text{polylog}(n)$, with n denoting the size of the input. Notably, each work considered a different oracle for showing the separation. For CM, the oracle is the same one as for Childs' glued trees problem [CCD+03]. For CCL, the oracle is a modified version of the oracle used for Simon's problem [Sim97], where the modification involves performing a sequence of permutations, allowing them to enforce high quantum depth.

CCL and CM were the first results to provide a convincing counterpoint to Jozsa's conjecture. However, the main drawback of the CCL and CM results is that they are relative to oracles that are highly structured and it is unclear if they can be explicitly instantiated based on some cryptographic assumption. Indeed,

[1] Technically [Mil92] only shows the strict containment $\mathsf{NC} \subsetneq \mathsf{P}$, relative to a random oracle. However, the quantum version $\mathsf{QNC} \subsetneq \mathsf{BQP}$ can also be shown as a straightforward extension of that result.

[2] Note that here and throughout the paper, the QNC oracle can output a string, unlike a decision oracle which outputs a bit.

[3] Note that the BPP oracle is not invoked coherently. Instead, it is invoked on outcomes resulting from intermediate measurements performed in the layers of the $\mathsf{QNC_d}$ circuit.

in his "Ten Semi-Grand Challenges for Quantum Computing Theory", Aaronson emphasizes this important distinction, and asks whether there is some *instantiatable* function that separates the hybrid models from BQP. In this work, we resolve Aaronson's question in the affirmative for the *search* variants of these classes.

In contrast to separations between different models of computation running in polynomial time, such as P and NP or BPP and BQP, where several plausible candidates exist for separating the classes, the case for depth separations is much more subtle. As was already observed in [BGJ+16], no standard cryptographic assumption is known to yield a separation between NC and P. The best candidates for such a separation are sequential compositions of hash functions (under the random oracle heuristic) as shown in [Mil92] and the iterated exponentiation scheme of Rivest, Shamir and Wagner [RSW96]. Thus, informally, the best we could hope for in terms of an instantiatable separation between the hybrid models and BQP is a separation in the random oracle model which could then be instantiated using cryptographic hash functions.

Our work is concerned not only with separations between the hybrid models and BQP in the random oracle model, but also with giving a comprehensive characterization of quantum depth in that model. To that end, we first re-examine Jozsa's conjecture and argue that the natural class associated to "d-depth quantum computation combined with polynomial-time classical computation" is not $\mathsf{BPP}^{\mathsf{QNC_d}} \cup \mathsf{QNC_d}^{\mathsf{BPP}}$, but $\mathsf{BPP}^{\mathsf{QNC_d}^{\mathsf{BPP}}}$. This is because, if one has the ability to perform $\mathsf{QNC_d}^{\mathsf{BPP}}$ computations, certainly it should also be possible to repeat this polynomially-many times as well as perform classical processing in between the runs. Note that $\mathsf{BPP}^{\mathsf{QNC_d}} \cup \mathsf{QNC_d}^{\mathsf{BPP}} \subseteq \mathsf{BPP}^{\mathsf{QNC_d}^{\mathsf{BPP}}}$. The separation we then obtain, relative to a random oracle, is $\mathsf{BPP}^{\mathsf{QNC_d}^{\mathsf{BPP}}} \subsetneq \mathsf{BQP}$, for any fixed $d \leq \mathrm{poly}(n)$. Going beyond this separation, we also show that the hybrid models $\mathsf{BPP}^{\mathsf{QNC_d}}$ and $\mathsf{QNC_d}^{\mathsf{BPP}}$ are separate from each other in both directions, relative to a random oracle (in fact, we show that $\mathsf{BPP}^{\mathsf{QNC}_{\mathcal{O}(1)}} \notin \mathsf{QNC_d}^{\mathsf{BPP}}$ and $\mathsf{QNC}_{\mathcal{O}(1)}^{\mathsf{BPP}} \notin \mathsf{BPP}^{\mathsf{QNC_d}}$), illustrating the subtle interplay between short-depth quantum computation and classical computation. Lastly, by combining the techniques that we develop with previous results on *proof of quantumness* protocols, we obtain *proof of quantum depth* protocols—protocols in which a BPP verifier, exchanging 2 messages[4] with an untrusted quantum prover, can certify that the prover has the ability to perform quantum computations of a minimum depth.

1.1 Main Results

We now state our results more formally and provide some intuition about the proofs. From here on, we abuse the notation slightly and use the standard *decision* complexity class names to refer to their *search* variants.

1.1.1 Lower bounds on quantum depth

We first show the following separation.

Theorem 1 (informal)**.** *Fix any function $d \leq \mathrm{poly}(n)$. Then, relative to a random oracle,*[5] *it holds that* $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}} \subsetneq \mathsf{BQP}$.

As motivated earlier, we take the class $\mathsf{BPP}^{\mathsf{QNC_d}^{\mathsf{BPP}}}$ to capture computations performed by a combination of d-depth quantum computation and polynomial-depth classical computation. The interpretation of our result is that $\mathsf{BPP}^{\mathsf{QNC_d}^{\mathsf{BPP}}}$ can be separated from BQP *using the least structured oracle possible*, a random oracle. Together with the (quantum) random oracle heuristic, by instantiating the oracle with a cryptographic hash function like SHA-2 or SHA-3, this yields the first plausible instantiation of a problem solvable in BQP but not in $\mathsf{BPP}^{\mathsf{QNC_d}^{\mathsf{BPP}}}$. This provides a resolution to Aaronson's challenge. The main technical innovation that allows us to achieve the separation is a general lifting lemma that takes any problem separating BPP from BQP in the random oracle model, which additionally satisfies a property that we call *classical query soundness*, and constructs a problem separating $\mathsf{BPP}^{\mathsf{QNC_d}^{\mathsf{BPP}}}$ and BQP. We show that several known problems satisfy this property. Our lifting lemma is inspired by [CCL20], and crucially extends their analysis beyond highly structured oracles. We describe this lifting lemma more precisely in Subsection 1.2.1.

1.1.2 Proofs of quantum depth

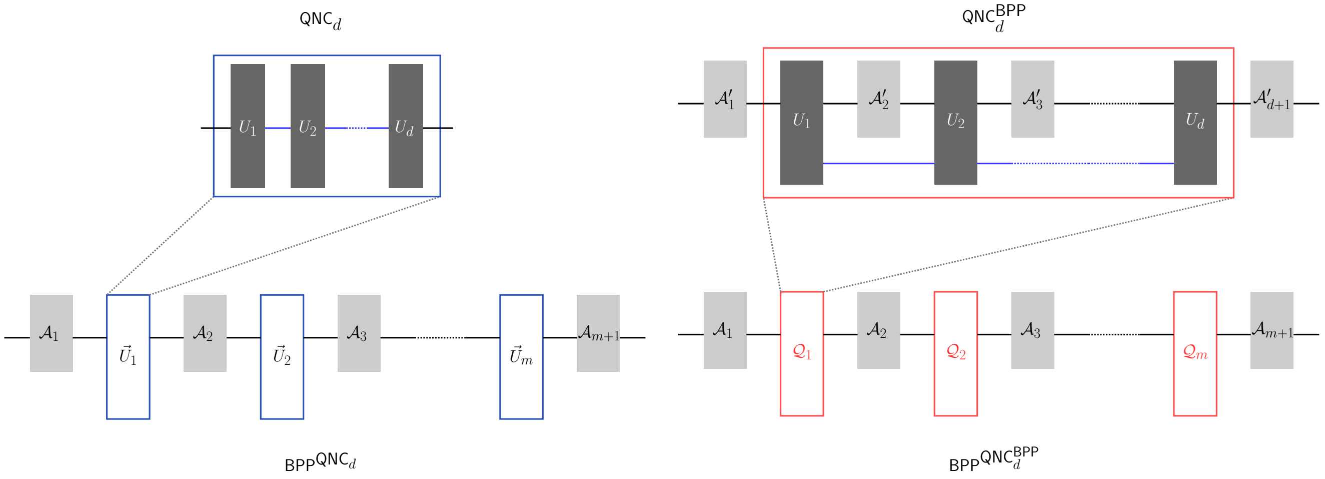
It is natural to wonder whether Theorem 1 yields an efficient test to certify quantum depth, i.e. a *proof of quantum depth*. A proof of quantum depth is a more fine-grained version of a proof of quantumness: rather

[4] 2 messages in total or a 1 round protocol.

[5] Here, as well as in all subsequent results, the statements hold with probability 1 over the choice of the random oracle. In addition, queries to the oracle are viewed as having depth 1.

$$\begin{tikzcd}
\mathsf{QNC}_d \arrow[r, "\text{adaptive measurements}", "\text{w/ BPP processing}"'] \arrow[d, "\text{poly-many invokes}"', "\text{w/ BPP processing}"'] & \mathsf{QNC}_d^{\mathsf{BPP}} \arrow[d, "\text{poly-many invokes}", "\text{w/ BPP processing}"] \\
\mathsf{BPP}^{\mathsf{QNC}_d} & \mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}
\end{tikzcd}$$

(a) Motivating the various hybrid quantum depth classes.



(b) Illustration of QNC_d and $\mathsf{QNC}_d^{\mathsf{BPP}}$ circuits.

(c) Illustration of $\mathsf{QNC}_d^{\mathsf{BPP}}$ and $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$ circuits.

Figure 1: The four hybrid quantum depth classes we consider. Blue wires carry qubits, black wires carry bits. Measurements are implicit and performed in the standard basis. U_is denote depth 1 unitaries, $\mathcal{A}_i$ and $\mathcal{A}'_i$ denote poly time classical algorithms.

than distinguishing between quantum and classical computation, a proof of quantum depth protocol can distinguish between provers having large or small quantum depth. We show that instantiating our lifting lemma with a problem whose solution is *efficiently verifiable* immediately yields a proof of quantum depth. One such problem[6] is due to Yamakawa and Zhandry [YZ22]. More precisely, we have the following.

Theorem 2 (informal)**.** *Let n be the security parameter and fix any function $d \leq \mathrm{poly}(n)$. In the random oracle model, there exists a two-message protocol between a poly-time classical verifier and a quantum prover such that,*

- *Completeness: There is a* BQP *prover which makes the verifier accept with probability at least* $1-\mathrm{negl}(n)$
- *Soundness: No malicious* $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$ *prover can make the verifier accept with probability greater than* $\mathrm{negl}(n)$*.*

We emphasise that considering protocols with more than two messages leads to difficulties in formalising the notion of quantum depth. For instance, one can construct protocols where the prover is forced to hold r single qubit states and subsequently measures them. Information about the basis in which to measure each of these qubits is sent one at a time by the verifier over r messages (the verifier waits for the response to each measurement, before sending the next basis). The measurement results are used by the verifier to ensure soundness (each qubit is measured in its preparation basis and so the outcomes are completely determined). It is not hard to show that if the prover measures these qubits without knowing the measurement basis, it cannot succeed except with negligible probability. If one attempts to model the prover as a $\mathsf{BPP}^{\mathsf{QNC}_d}$ or $\mathsf{QNC}_d^{\mathsf{BPP}}$ circuit, then, because of the delay between messages, it appears that $d \geq r$ is necessary. However, this can be seen as an artefact of the modelling choice: in practice, the prover only needs d single qubit

[6] We remark that, if one is only concerned with the complexity-theoretic separation of Theorem 1, and not with efficient verification, then a much simpler problem suffices (see $\mathsf{CollisionHashing}$ in Table 3).

quantum computers with quantum depth 1 where the last gate can be delayed until the appropriate message is received in order to pass the test. Essentially, this approach only tests the prover's ability to maintain the coherence of the qubits it received, without actually testing the depth of the circuit it has to perform. In Subsection 1.3, we discuss a possible resolution that captures quantum depth in the interactive setting.

1.1.3 Tighter bounds

While Theorem 1 establishes that $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$ does not capture the computational power of BQP for any fixed $d \leq \mathrm{poly}(n)$, it is not a priori clear if, for instance, $\mathsf{BPP}^{\mathsf{QNC}_{2d+\mathcal{O}(1)}^{\mathsf{BPP}}}$ is strictly larger than $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$. Indeed, we show that the answer is affirmative.

Theorem 3 (informal)**.** *Fix any function $d \leq \mathrm{poly}(n)$. Relative to a random oracle, it holds that*[7] $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}} \subsetneq \mathsf{BPP}^{\mathsf{QNC}_{2d+\mathcal{O}(1)}^{\mathsf{BPP}}}$.

Formally, Theorem 1 treats a call to the quantum random oracle as a depth-1 quantum gate. In practice, if instead the gate requires depth ℓ, then d can be replaced by $d\ell$. We remark that there exist hash functions that are thought to be quantum-secure which require only logarithmic depth to evaluate [Ajt96; PS19]. Further, there is reason to believe that such hash functions could also be constructed in $\ell = \mathcal{O}(1)$ depth. In particular, if one is only concerned with specific cryptographic properties (such as collision resistance), then generic constructions are known which convert log-depth hash functions into ones that require only constant depth [AIK06].

1.1.4 Separations between hybrid quantum depth classes

While both $\mathsf{BPP}^{\mathsf{QNC}}$ and $\mathsf{QNC}^{\mathsf{BPP}}$ capture some notion of a hybrid between efficient classical computation and shallow quantum computation, the relationship between the two is not immediately clear. To get a slightly better intuition about the two models, one can think of $\mathsf{BPP}^{\mathsf{QNC}}$ as capturing an efficient computation that contains *polynomially many* shallow quantum circuits (separated by measurements and classical computation). On the other hand, one can think of $\mathsf{QNC}^{\mathsf{BPP}}$ as a *single* shallow quantum circuit, where one is allowed to make partial measurements of some of the wires, and choose the next gates *adaptively*. While it may not be surprising that there exist problems that can be solved in $\mathsf{BPP}^{\mathsf{QNC}}$ but not in $\mathsf{QNC}^{\mathsf{BPP}}$, it turns out that the two classes are in fact incomparable—each class contains problems that the other does not, relative to a random oracle.

Theorem 4 (informal)**.** *Fix any function $d \leq \mathrm{poly}(n)$. Relative to a random oracle, it holds that* $\mathsf{BPP}^{\mathsf{QNC}_{\mathcal{O}(1)}} \nsubseteq \mathsf{QNC}_d^{\mathsf{BPP}}$ *and* $\mathsf{QNC}_{\mathcal{O}(1)}^{\mathsf{BPP}} \nsubseteq \mathsf{BPP}^{\mathsf{QNC}_d}$.

The second separation is arguably more surprising. It says that, relative to a random oracle, there are problems that can be solved by a *single* shallow (in fact, constant-depth) quantum circuit *with* adaptive measurements but cannot be solved by circuits with *polynomially many* shallow quantum circuits *without* adaptive measurements. The problem that shows $\mathsf{QNC}_{\mathcal{O}(1)}^{\mathsf{BPP}} \nsubseteq \mathsf{BPP}^{\mathsf{QNC}_d}$ is a variant of the proof of quantumness from [BKVV20]. The key technical innovation to achieve this separation is a theorem that characterises the structure of strategies that succeed in the protocol of [BKVV20] (this is discussed further in Section 1.2.2). This "structure theorem" crucially strengthens a similar theorem from [CGV22], and may be of independent interest.

Finally, we examine the relationship between $\mathsf{BPP}^{\mathsf{QNC}_d} \cup \mathsf{QNC}_d^{\mathsf{BPP}}$ and $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$. By definition, it is manifest that $\mathsf{BPP}^{\mathsf{QNC}_d} \cup \mathsf{QNC}_d^{\mathsf{BPP}} \subseteq \mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$. Even though $\mathsf{QNC}_d^{\mathsf{BPP}}$ and $\mathsf{BPP}^{\mathsf{QNC}_d}$ are incomparable, it is conceivable that their union captures any reasonable notion of quantum depth d. We show that this is not the case.

Theorem 5 (informal)**.** *Fix any function $d \leq \mathrm{poly}(n)$. Relative to a random oracle, it holds that* $\mathsf{BPP}^{\mathsf{QNC}_{\mathcal{O}(1)}^{\mathsf{BPP}}} \nsubseteq \mathsf{BPP}^{\mathsf{QNC}_d} \cup \mathsf{QNC}_d^{\mathsf{BPP}}$.

In words, the latter theorem asserts that a computation consisting of polynomially many layers of *constant*-depth quantum circuits with adaptive control cannot be simulated by quantum circuits with d depth which are either adaptive (but consisting of a single d-depth quantum circuit) or consisting of many d-depth quantum circuits (but without adaptive control).

[7] and more generally, that $\mathsf{QNC}_{2d+\mathcal{O}(1)} \nsubseteq \mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$.

1.1.5 Summary

Result	Remarks
$\mathsf{BPP}^{\mathsf{QNC}^{\mathsf{BPP}}} \subsetneq \mathsf{BQP}$	Refutes Jozsa's conjecture in the random oracle model
$\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}} \subsetneq \mathsf{BPP}^{\mathsf{QNC}_{2d+\mathcal{O}(1)}^{\mathsf{BPP}}}$	Fine grained advantage of quantum depth

Table 1: (Simplified) Bounds on quantum depth. Separations are with respect to the random oracle and $d \le \mathrm{poly}(n)$ is any fixed function of the input size.

Result	Physical Interpretation
$\mathsf{BPP}^{\mathsf{QNC}_{\mathcal{O}(1)}} \nsubseteq \mathsf{QNC}^{\mathsf{BPP}}$	Running *poly many constant* depth quantum circuits (with *no* adaptive measurements) cannot be simulated by running a *single* log depth quantum circuit *with* adaptive measurements.
$\mathsf{QNC}_{\mathcal{O}(1)}^{\mathsf{BPP}} \nsubseteq \mathsf{BPP}^{\mathsf{QNC}}$	Running a *single constant* depth quantum circuit *with* adaptive measurements cannot be simulated by running *poly many* log depth quantum circuits (with *no* adaptive measurements).
$\mathsf{BPP}^{\mathsf{QNC}_{\mathcal{O}(1)}^{\mathsf{BPP}}} \nsubseteq \mathsf{BPP}^{\mathsf{QNC}} \cup \mathsf{QNC}^{\mathsf{BPP}}$	Evidence that it is not enough to consider $\mathsf{BPP}^{\mathsf{QNC}}$ and $\mathsf{QNC}^{\mathsf{BPP}}$ when studying quantum depth. Running *poly many constant* depth quantum circuits *with* adaptive measurements cannot be simulated using either (a) *poly many* log depth quantum circuits with *no* adaptive measurements, or by (b) a *single* log *depth* quantum circuit *with* adaptive measurements.

Table 2: (Simplified) Separations of hybrid quantum depth with respect to the random oracle. The results hold, not only for log but for any fixed polynomially-bounded function.

Table 1 lists our lower bounds on quantum depth, and Table 2 lists the separations among the hybrid classes.

1.2 Main technical contributions

1.2.1 Lifting Lemmas

One of the main technical contributions of our work is to prove *two* general lifting lemmas. These lemmas take problems, defined relative to a random oracle, that are classically hard (in a stronger sense, defined next) and create new problems which are, in addition, hard for specific hybrid quantum depth classes. We describe these lifting lemmas a bit more precisely.

We say that a problem (defined with respect to the random oracle) is *classical query sound* if the following holds: any (potentially unbounded time) algorithm which makes only polynomially many *classical* queries to the random oracle (i.e. no superposition queries), succeeds at solving the problem with at most negligible probability. It turns out that the problem introduced by YZ satisfies this property. Another problem which satisfies this property is inspired by the proof of quantumness protocol defined by Brakerski et al. [BKVV20] (hereafter referred to as BKVV).[8] For such problems, the following holds.

Lemma 6 (informal, simplified)**.** *There is a procedure*[9] *that takes a classical query sound problem* $\mathcal{P} \in \mathsf{BQP}$ *and creates a new problem* $\mathcal{P}' := d\text{-}\mathsf{Rec}[\mathcal{P}]$*, such that* $\mathcal{P}' \notin \mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$ *and* $\mathcal{P}' \in \mathsf{BQP}$*.*

[8] Which we refer to as $\mathsf{CollisionHashing}$ later.

[9] $d\text{-}\mathsf{Rec}[\cdot]$ is meant to be short for d-Recursive.

Observe that this lemma makes the problem hard for *the most general notion of quantum depth* we have considered. To give some intuition about how it is derived, suppose we have a problem $\mathcal{P}$ which is classical query sound and denote the random oracle as H. Then $\mathcal{P}' = d\text{-}\mathsf{Rec}[\mathcal{P}]$ is the same problem, defined with respect to *a sequential composition of* $d+1$ *random oracles*, $\tilde{H} = H_d \circ \cdots \circ H_0$. In essence, we have substituted H with $\tilde{H}$. This new problem will retain classical query soundness, as $\tilde{H}$ behaves like a random oracle. But in addition, we have now made it so that querying $\tilde{H}$ effectively requires depth $d+1$. As QNC_d has depth d, only the BPP parts of $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$ will be able to query $\tilde{H}$. We can therefore simulate the $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$ algorithm with an exponential time algorithm that is limited to polynomially many queries to $\tilde{H}$. By classical query soundness, such an algorithm cannot solve $\mathcal{P}'$, which yields the desired result.

This was a simplified description of our result. In fact, we show a more refined statement that relates the depth required to solve $\mathcal{P}'$ to the depth required to solve $\mathcal{P}$. In addition, arguing that $\tilde{H}$ behaves like a random oracle and that QNC_d cannot query $\tilde{H}$ requires a careful and more involved analysis. We use Lemma 6 to establish Theorem 3.

Our second lifting lemma produces a problem that is hard for $\mathsf{QNC}_d^{\mathsf{BPP}}$, starting from a problem that satisfies what we call *offline soundness*. Consider a two phase algorithm consisting of: *an online phase* which is a poly-time classical algorithm *with access* to the random oracle followed by *an offline phase* which is an unbounded(-time) algorithm with *no access* to the random oracle. Then, *offline soundness* requires that no such two phase algorithm succeeds at solving the problem with non-negligible probability. It turns out, again, that both YZ and BKVV satisfy this property.

Lemma 7 (informal)**.** *There is a procedure*[10] *which takes a problem* $\mathcal{P} \in \mathsf{QNC}_{\mathcal{O}(1)}$ *with offline soundness and creates a new problem* $\mathcal{P}' := d\text{-}\mathsf{Ser}[\mathcal{P}]$ *such that* $\mathcal{P}' \notin \mathsf{QNC}_d^{\mathsf{BPP}}$ *and* $\mathcal{P}' \in \mathsf{BPP}^{\mathsf{QNC}_{\mathcal{O}(1)}}$*.*

Again, we actually show a slightly more general upper bound which depends on the depth required to solve $\mathcal{P}$. We use Lemma 7 to establish $\mathsf{BPP}^{\mathsf{QNC}_{\mathcal{O}(1)}} \nsubseteq \mathsf{QNC}_d^{\mathsf{BPP}}$ (first separation of Theorem 4). Establishing the other direction ($\mathsf{QNC}_{\mathcal{O}(1)}^{\mathsf{BPP}} \nsubseteq \mathsf{BPP}^{\mathsf{QNC}_d}$) is quite involved and relies heavily on the structure of the problem we consider (explained below). Consequently, it is unclear whether there exists a general lifting lemma that yields hardness for $\mathsf{BPP}^{\mathsf{QNC}_d}$.

We remark that, by using Lemma 7 to lift the problem that yields $\mathsf{QNC}_{\mathcal{O}(1)}^{\mathsf{BPP}} \nsubseteq \mathsf{BPP}^{\mathsf{QNC}_d}$, we also obtain Theorem 5, i.e. $\mathsf{BPP}^{\mathsf{QNC}_1^{\mathsf{BPP}}} \nsubseteq \mathsf{BPP}^{\mathsf{QNC}_d} \cup \mathsf{QNC}_d^{\mathsf{BPP}}$.

1.2.2 A structure theorem for [BKVV20]

Another technical contribution of this work, which may be of independent interest, is to prove a theorem characterizing the structure of strategies that are successful at the proof of quantumness from [BKVV20]. This theorem is a crucial strengthening of a theorem from [CGV22]. We employ this theorem as an intermediate step to establish the hybrid separation, $\mathsf{QNC}_{\mathcal{O}(1)}^{\mathsf{BPP}} \nsubseteq \mathsf{BPP}^{\mathsf{QNC}_d}$.

Recall, informally, that the proof of quantumness from [BKVV20] requires the prover to succeed at the following task: given access to a 2-to-1 function g, and to a random oracle H with a one-bit output, find a pair (y, r) such that

$$r \cdot (x_0 \oplus x_1) \oplus H(x_0) \oplus H(x_1) = 0,$$

where $\{x_0, x_1\} = g^{-1}(y)$. This can be solved in $\mathsf{QNC}_{\mathcal{O}(1)}$ as follows:

(i) Evaluate g on a uniform superposition of inputs, yielding $\sum_x |x\rangle |g(x)\rangle$,

(ii) Measure the image register obtaining some outcome y and a state $(|x_0\rangle + |x_1\rangle)|y\rangle$,

(iii) Query a phase oracle for H to obtain $((-1)^{H(x_0)}|x_0\rangle + (-1)^{H(x_1)}|x_1\rangle)|y\rangle$,

(iv) Make a Hadamard basis measurement of the first register, obtaining outcome r.

Informally, our structure theorem establishes that querying at a superposition of pre-images is essentially *the only way to succeed* (provided finding a collision for g is hard—this is the case when g is a trapdoor claw-free function, as in [BKVV20], but more generally our theorem also holds e.g. when g is a uniformly random 2-to-1 function). Denote by n the bit-length of strings in the domain of g.

[10] d-$\mathsf{Ser}[\cdot]$ is meant to be short for d-Serial.

Theorem 8 (informal)**.** *Let P be any* BQP *prover that succeeds with* $1 - \mathrm{negl}(n)$ *probability at the proof of quantumness protocol from [BKVV20], by making q queries to the oracle H. Then, with* $1-\mathrm{negl}(n)$ *probability over pairs (H, y), the following holds. Let $p_{y|H}$ be the probability that P^H outputs y, and let x_0,x_1 be the pre-images of y. Then, for all $b \in \{0,1\}$, there exists $i \in [q]$ such that the state of the query register of P^H right before the i-th query has weight $\frac{1}{2}p_{y|H} \cdot (1 - \mathrm{negl}(n))$ on x_b.*

Note that a version of the above theorem that applies to provers who win with probability non-negligibly greater than $\frac{1}{2}$ also holds (but we stated the close-to-ideal version for simplicity). We provide a sketch of how this theorem is used in the proof of $\mathsf{QNC}^{\mathsf{BPP}}_{\mathcal{O}(1)} \not\subseteq \mathsf{BPP}^{\mathsf{QNC}_d}$ in Subsection 2.2.2. We refer to Corollary 113 for a formal statement of the theorem.

1.3 Discussion and open problems

Further questions in the random oracle model. Our separations are with respect to *search* problems. The main question left open by our work is whether the same separations can be shown with respect to *decision* problems. Recall that our approach to proving the separations is to *lift* a problem that separates BPP and BQP in the random oracle model (for example a *proof of quantumness*) to a problem that requires at least a certain amount of quantum depth. However, we note that this approach is unlikely to yield depth separations for decision problems. This is because the Aaronson-Ambainis conjecture [AA09] states that one cannot separate the decision versions of BPP and BQP in the random oracle model. Thus, a different approach is likely to be necessary.

Another interesting related question is the following. When we instantiate our lifting lemma with the proof of quantumness from YZ, the resulting problem inherits the property that solutions can be publicly verified. We thus obtain a proof of quantum depth that is publicly verifiable. Can we further push this quantum soundness to obtain verification of BQP with a BPP verifier relative to a random oracle?

We have also seen that making use of a problem inspired by the Brakerski et al. [BKVV20] proof of quantumness allows us to prove more fine grained separations between hybrid classes. It is then natural to ask, whether these separations also yield *finer grained proofs of quantum depth* (which are sound against $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$ provers and complete for a $\mathsf{BPP}^{\mathsf{QNC}^{\mathsf{BPP}}_{2d+\mathcal{O}(1)}}$ prover). This does not immediately follow from our results, as the problem we construct from BKVV is not efficiently verifiable, and our current techniques do not directly extend to the computationally-bounded setting. We therefore leave this as an open problem.

Separations without the random oracle. Our work gives the first instantiatable quantum depth separation by virtue of being in the random oracle model. It is natural to ask if one can establish this separation in the plain model. Unfortunately, a separation in the random oracle model seems to be the best that one can hope for, given that even for classical depth there are no known separations that rely on standard cryptographic assumptions (other than the random oracle). In some sense this is peculiar, since one would imagine that using more structured problems would allow one to prove stronger separations. The random oracle is the least structured type of oracle, but the fact that it is an oracle helps in establishing provable lower bounds.

Generalizing beyond $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$. We have argued that $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$ is the most natural class capturing the notion of d-depth quantum computation, combined with polynomial-depth classical computation. However, for the purpose of *certifying* quantum depth, as we have mentioned earlier (and as we discuss in more detail in Example 12), the situation becomes more subtle when the certification protocol involves *interaction*. We therefore propose that any protocol which establishes quantum depth d and uses r rounds of interaction should be sound against at least an r level generalization of $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$ (e.g. a 2 level generalization with quantum depth d would be $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}}}$ — here 2 counts the number of times QNC_d appears in the tower of complexity classes, so that an r level generalisation would have r appearances of QNC_d). In our case, since the proof of depth protocols are single-round, we show the necessary soundness against a $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$ prover.

Of course, there are other possible ways to define hybrid d-depth quantum-classical computation. For instance, one can define the class QDepth_d of problems solved by polynomial sized circuits with quantum and classical gates where the key constraint is that *the longest path connecting quantum gates (with quantum wires) is at most d*. We expect that the union over all r level generalizations of $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$ (where r is polynomially bounded) equals QDepth_d. We also expect our separating problems (and d-$\mathsf{Rec}[\mathcal{P}]$ in general, for classical query sound $\mathcal{P}$) to not be in QDepth_d, but we leave the proof to future work.

1.4 Previous work

We compare our results to the previous works [CCL20], [CM20], [AGS22], and [CH22].

Comparison to [CCL20], [CM20] and [AGS22]. Compared to previous work on the topic, our work gives a comprehensive treatment of the complexity of hybrid quantum-classical computation.

As mentioned earlier, the primary difference compared to [CCL20] and [CM20] is that all of our separations are with respect to a random oracle, rather than with respect to highly structured oracles. However, one caveat is that our separations are for search problems. Our contribution is also conceptual. We propose $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$ as the appropriate model to capture "d-depth quantum computation combined with polynomial-time classical computation". While [CCL20] and [CM20] showed that $\mathsf{BPP}^{\mathsf{QNC}_d} \cup \mathsf{QNC}_d^{\mathsf{BPP}} \not\subseteq \mathsf{BQP}$, we show the stronger result that $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}} \not\subseteq \mathsf{BQP}$.

Our work also shows separations between different hybrid models. Such separations were considered in [AGS22], where they are again proven only with respect to highly structured oracles.

In terms of techniques, we take inspiration and ideas from both [CCL20] and [AGS22]. In particular we build on two key ideas—sampling argument and domain hiding. One of the main contribution of our analysis is to abstract and generalise these techniques beyond their original scope which was tailored to specific promise problems. While most of our results build on these techniques, we also point out that to prove the separation between the hybrid models $\mathsf{QNC}_{\mathcal{O}(1)}^{\mathsf{BPP}} \not\subseteq \mathsf{BPP}^{\mathsf{QNC}_d}$ we use entirely different ideas. In particular, as an intermediate step, we establish a theorem that characterizes the structure of strategies that succeed at the proof of quantumness in BKVV, which may be of independent interest.

Comparison to [CH22]. The work of [CH22] was the first to consider proofs of quantum depth. However, the notion of soundness that they propose, and their corresponding protocol (in the single prover setting), suffers from the issues that we discussed after Theorem 2 (and in Example 12 below).

In particular, their protocol can be spoofed by a d level tower of $\mathsf{BPP}^{\mathsf{QNC}_{\mathcal{O}(1)}^{\mathsf{BPP}}}$ (as described in Subsection 1.3). In practical terms, this means that it can be spoofed by running several constant depth quantum computers in parallel, provided the "idle coherence time" of each quantum computer is longer than the time that elapses between messages in the protocol. In contrast, our proof of depth protocol does not suffer from this issue and can be used to certify that the prover is able to perform computations "beyond" $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$.

Acknowledgments

We are thankful to Joseph Slote, Ulysse Chabaud and Thomas Vidick for various insightful discussions. While at ETH, AG was supported by Dr. Max Rössler, the Walter Haefner Foundation and the ETH Zürich Foundation. AC is a Quantum Postdoctoral Fellow at the Simons Institute for the Theory of Computing supported by NSF QLCI Grant No. 2016245, and by DARPA under agreement No. HR00112020023. 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 United States Government or DARPA. US acknowledges the support by Polish National Science Center (NCN) (Grant No. 2019/35/B/ST2/01896). HW is supported by an MC2 postdoctoral fellowship.

2 Technical Overview

Here we give a high level technical overview of the paper.

2.1 Bounds on quantum depth — $\mathsf{BPP}^{\mathsf{QNC}^{\mathsf{BPP}}} \subsetneq \mathsf{BQP}$

In this subsection, we describe the proof of Theorem 1. As mentioned previously, our main technical contribution is a general lifting lemma that takes any problem separating BPP from BQP in the random oracle model, which additionally satisfies a property that we call *classical query soundness*, and constructs a problem separating $\mathsf{BPP}^{\mathsf{QNC}_\mathsf{d}^{\mathsf{BPP}}}$ and BQP. We first explain the key idea behind this construction. To be concrete, after describing the key idea, we restrict to an NP search problem due to Yamakawa and Zhandry [YZ22], which satisfies classical query soundness (this problem is particularly appealing because it is in NP, and thus solutions can be publicly verified, however we emphasize that other known search problems that are not in NP can also be used for the separation). We then build towards a proof that this problem is not in $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$ by considering hardness for the three special cases QNC_d, $\mathsf{QNC}_d^{\mathsf{BPP}}$ and $\mathsf{BPP}^{\mathsf{QNC}_d}$. The desired result is obtained by combining the ideas in these three cases.

Let $\mathcal{P}$ be a (search) problem, defined relative to a random oracle H, that separates BPP from BQP. Suppose that $\mathcal{P}$ is such that it requires *quantum* access to H in order to be solved with polynomially many queries (*classical query soundness* will eventually require a bit more than this). As mentioned in Subsection 1.2.1, the first natural idea to lift this to a separation between low quantum depth and polynomial quantum depth is to *replace the evaluation of H with a sequential evaluation of random oracles*. For example, suppose that originally $H : \Sigma \to \{0,1\}^n$. Then, let $H_0, \ldots, H_{d-1} : \Sigma \to \Sigma$, and $H_d : \Sigma \to \{0,1\}^n$ be random oracles. Define $\tilde{H} = H_d \circ \cdots \circ H_0$. Now, let $\mathcal{P}'$ be the problem that is identical to $\mathcal{P}$ except that it is relative to $\tilde{H}$. Then, it is natural to imagine that $\mathcal{P}'$ requires quantum depth at least $d+1$ to solve. This idea does not quite work right away, since $\tilde{H}$, as defined, is not actually a uniformly random oracle any more. This is because with every H_i that is added, the number of collisions in $\tilde{H}$ increases (on average). To remedy this, one could assume that $H_0, \ldots, H_{d-1}$ are random *permutations* (although note that random permutations cannot be generically constructed from random oracles). A similar idea works in a different setting, for arguing about the post-quantum security of "proofs of sequential work" [BLZ21]. However, in our case, the analysis is complicated by the fact that we consider hybrid models. CCL were the first to consider a variant of sequential hashing (sequential permutations), in the context of hybrid models. However, their analysis only works for certain structured oracles. In this work, we adapt their ideas to the random oracle setting and overcome these difficulties.

Lifting $\mathcal{P} \notin \mathsf{BPP}$ to $\tilde{\mathcal{P}} \notin \mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$. Given a problem $\mathcal{P}$ with respect to H, we define the problem $\tilde{\mathcal{P}} = d\text{-}\mathsf{Rec}[\mathcal{P}]$ to be $\mathcal{P}$ with respect to $\tilde{H} = H_d \circ \cdots \circ H_0$ where $H_0, \ldots, H_d$ are independent random oracles with the following domains and co-domains: $H_0 : \Sigma \to \Sigma^{d'}$, $H_i : \Sigma^{d'} \to \Sigma^{d'}$ for $i \in \{1 \ldots d-1\}$, and $H_d : \Sigma^{d'} \to \{0,1\}^n$ with $d' = 2d + 5$.

Notice that H_0 is not surjective, as its codomain is much larger than its image.[11] In fact, this is also true for $H_i \circ \cdots \circ H_0$, for all $i < d$. This and the fact that the H_i functions are random, have two important consequences. First, it means that with high probability $H_{d-1} \circ \cdots \circ H_0$ is injective and so $\tilde{H}$ behaves like a random oracle. Consequently, $\mathcal{P}'$ inherits the soundness and completeness of $\mathcal{P}$. Second, it means that one can apply a "domain hiding" technique, which, at a high level, works as follows. One way of evaluating $\tilde{H}$ at $x \in \Sigma$ is to sequentially compose $H_0, H_1, \ldots, H_d$ which would require depth $d+1$. Intuitively, it seems unlikely that there is a more depth efficient way of evaluating $\tilde{H}$ because the domain on which the H_i's need to be evaluated (which is $H_{i-1} \circ \cdots \circ H_0(\Sigma)$) is getting shuffled and lost in an exponentially larger domain (which is $\Sigma^{d'}$). Therefore, even though one has access to all $\mathcal{L} = (H_0, H_1, \ldots, H_d)$ oracles at the first layer of depth, one only knows that H_0 needs to be queried at Σ but the algorithm has no information about where the relevant domains of $H_1 \ldots H_d$ are. At the second depth layer, the algorithm can learn $H_0(\Sigma)$ and so learns where to query H_1 but, and this needs to be shown, it still does not know where the relevant domains of $H_2, \ldots H_d$ are. By starting with a sufficiently large expansion, i.e. a sufficiently large $d' > d$, this argument can be repeated until depth d where the relevant domain of H_d still remains hidden. Thus, even though $\mathcal{P}'$ can potentially be solved with $d+1$ depth, it cannot be solved with depth d. This is the basic idea behind why the problem is not in QNC_d. Instead of working with $\mathcal{P}$ and $d\text{-}\mathsf{Rec}[\mathcal{P}]$ abstractly, we consider the following concrete problem.

[11] We sometimes refer to this fact by saying that the function is "expanding".

2.1.1 d-CodeHashing — The problem

We refer to the problem introduced by Yamakawa and Zhandry [YZ22] as CodeHashing in this work. The problem is stated in terms of a family of error-correcting codes called *suitable codes*. For our purposes, it suffices to think of suitable codes as a family of sets $\{C_\lambda\}_\lambda$ where each C_λ is a set of codewords $\{(\mathrm{x}_1,\dots \mathrm{x}_n)\}$ with each coordinate x_i belonging to some alphabet Σ. The size of this alphabet, $|\Sigma| = 2^{\lambda^{\Theta(1)}}$ is exponential in λ, and the number of components $n = \Theta(\lambda)$ essentially equal to λ. CodeHashing is defined as follows.

Definition 9 (CodeHashing; informal)**.** Let $\{C_\lambda\}_\lambda$ be a suitable code and let $H : \{0,1\}^{\log n} \times \Sigma \to \{0,1\}$ be a random oracle. Given a description of the suitable code (e.g. as parity check matrices) and oracle access to H, on input 1^λ, the problem is to find a codeword $\mathrm{x} = (\mathrm{x}_1 \dots \mathrm{x}_n) \in C_\lambda$ such that[12] $H(i\|\mathrm{x}_i) = 1$ for all $i \in \{1\dots n\}$.

Note that CodeHashing is an NP search problem, since from, e.g. the parity check matrix of the code, it is easy to verify that x is indeed a codeword and with a single parallel query (n queries in total) to H, one can check that it hashes correctly.

YZ shows that CodeHashing satisfies the following two properties.

Lemma 10 (Paraphrased from YZ)**.** *The following hold.*

- Completeness: *There is a QPT machine which solves* CodeHashing *with probability* $1 - \mathrm{negl}(\lambda)$ *and makes only one parallel query to* H.
- Soundness: *Every (potentially unbounded time) classical circuit which makes at most* 2^{λ^c} *queries to* H, *with* $c < 1$, *solves* CodeHashing *with probability at most* $2^{-\Omega(\lambda)}$.

The fact that soundness holds against *unbounded time* classical circuits which make only poly-many queries to the random oracle is essential in proving that $\mathsf{BPP}^{\mathsf{QNC}^{\mathsf{BPP}}} \subsetneq \mathsf{BQP}$. Applying our lifting map, d-$\mathsf{Rec}[\mathcal{P}]$ on CodeHashing we obtain the following.[13]

Definition 11 (d-CodeHashing; informal)**.** Let $\{C_\lambda\}_\lambda$ be a suitable code, and $\tilde{H} := H_d \circ \dots \circ H_1 \circ H_0$, where $H_0, \dots, H_d$ are as in Section 2.1. Given a description of the suitable code, access to random oracles $\mathcal{L} = (H_0 \dots H_d)$, on input 1^λ, find a codeword $\mathrm{x} = (\mathrm{x}_1 \dots \mathrm{x}_n) \in C_\lambda$ such that $\mathrm{bit}_i[\tilde{H}(\mathrm{x}_i)] = 1$ for all $i \in \{1\dots n\}$.

To convey the key ideas behind the proof that d-CodeHashing $\notin \mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$, we first consider the QNC_d case in some more detail, and extend the analysis to $\mathsf{QNC}_d^{\mathsf{BPP}}$. We then analyse the $\mathsf{BPP}^{\mathsf{QNC}_d}$ case, which uses a technique called the "sampling argument" due to [CDGS18]. These ideas were first considered in the structured oracle setting by [CCL20] and [AGS22]. We adapt them to show d-CodeHashing $\notin \mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$ relative to a random oracle.

2.1.2 d-CodeHashing $\notin \mathsf{QNC}_d$

Base sets. We started our discussion in Subsection 2.1 by observing that the analysis is simplified by taking $H_0 \dots H_{d-1}$ to be injective functions. However, for a large enough d', it is not hard to see that this is indeed the case on an appropriately restricted domain. The sets which describe this restricted domain are chosen randomly. We call them *base sets* and denote them by $S_{01}, \dots S_{0d}$ (corresponding to $H_1, \dots H_d$ respectively). Observe that H_0 maps Σ to $\Sigma^{d'}$ (which is exponentially larger than Σ; recall that $|\Sigma| = 2^{\lambda^{\Theta(1)}}$) and, since H_0 is a random function, the probability that this mapping is injective is $1 - \mathrm{negl}(\lambda)$. Pick any set $S_{01} \subseteq \Sigma^{d'}$ uniformly at random in the domain of H_1 subject to two constraints: (1) it includes $H_0(\Sigma)$, i.e. the domain of H_1 on which the value of $\tilde{H}$ depends, and (2) its size is $|S_{01}| = |\Sigma|^{d+2}$. The first constraint ensures that the domain we care about is included in the base sets and the second ensures that: (a) $|S_{01}|$ is exponentially smaller than $|\Sigma|^{d'}$ and (b) $|S_{01}|$ is large enough for applying "domain hiding" as mentioned above. Define $S_{0i} := H_{i-1}(\dots H_1(S_{01})\dots)$ to be the image of S_{01} through the first 1 to $(i-1)$'th oracles for $i \in \{2\dots d\}$. Let E denote the event that H_0 is injective and $H_1 \dots H_{d-1}$ are injective on the base sets. We show that E (given our choice for d'), occurs with overwhelming probability. In the subsequent discussion, we assume that base sets have been selected and that E occurs.

[12] We use $a\|b$ to mean concatenation of a and b.

[13] We used $\mathrm{bit_i}[\tilde{\mathrm{H}}(\cdot)] = 1$ instead of $\tilde{H}(i\|\cdot) = 1$ for notational convenience later.

Proof idea. We describe the proof that d-$\mathsf{CodeHashing} \notin \mathsf{QNC}_d$ in some more detail, which implements the previously described "domain hiding" idea and proceeds via a hybrid argument. Denote a QNC_d circuit that makes d parallel calls to the oracle $\mathcal{L} = (H_0, \dots H_d)$ by $U_{d+1} \circ \mathcal{L} \circ U_d \dots U_2 \circ \mathcal{L} \circ U_1 \circ \rho_0$. Here, ρ_0 is some initial state, U_i are single layered unitaries, and the composition is meant to act as conjugation, i.e. $U_1 \circ \rho_0 = U_1 \rho_0 U_1^\dagger$. We show that the behaviour of such a circuit, i.e. its probability of outputting a valid answer, is negligibly close to the behaviour of another circuit $U_{d+1} \circ \mathcal{M}_d \circ U_d \dots U_2 \circ \mathcal{M}_1 \circ U_1 \circ \rho_0$ where $\mathcal{M}_1, \dots \mathcal{M}_d$ are "shadow oracles" corresponding to $\mathcal{L}$ that contain no information about the values taken by $\tilde{H}$ on Σ. Clearly then, this circuit cannot be solving d-$\mathsf{CodeHashing}$ because it never queries $\tilde{H}$. This in turn means that the original circuit also cannot solve d-$\mathsf{CodeHashing}$, which implies d-$\mathsf{CodeHashing} \notin \mathsf{QNC}_d$. It remains to define $\mathcal{M}_1 \dots \mathcal{M}_d$ and to argue that the two circuits have essentially the same behaviour. Using a hybrid argument, one can establish the latter by showing that the following are close in trace distance: (1) $\mathcal{L} \circ U_1 \circ \rho_0$ and $\mathcal{M}_1 \circ U_1 \circ \rho_0$, (2) $\mathcal{L} \circ U_2 \circ \mathcal{M}_1 \circ U_1 \circ \rho_0$ and $\mathcal{M}_2 \circ U_2 \circ \mathcal{M}_1 \circ U_1 \circ \rho_0$, and so on. To convey intuition, we sketch these steps one at a time, and we define $\mathcal{M}_1 \dots \mathcal{M}_d$ as we proceed. We restrict to base sets $S_{01} \dots S_{0d}$ as described above.

Hybrid 1. $\mathcal{L} \circ U_1 \circ \rho_0 \approx \mathcal{M}_1 \circ U_1 \circ \rho_0$.
Let $S_{11} \subseteq S_{01}$ be a random subset of S_{01}, subject to the constraints that (a) it includes $S_1 := H_0(\Sigma)$ and (b) $|S_{11}|/|S_{01}| = 1/|\Sigma| = \mathrm{negl}(\lambda)$. Let $S_{1j} := H_{j-1}(S_{1,j-1})$ be the propagation of S_{11} through H_1 to H_{j-1}. Here, we are trying to define a sequence of sets $(S_{11}, \dots S_{1d})$ on which we require that $\mathcal{M}_1$ outputs $\perp$ and outside of these sets, we require that $\mathcal{M}_1$ behaves just like $\mathcal{L}$, i.e. if one denotes $\mathcal{M}_1 = (H_0, M_{11}, \dots M_{1d})$, then we require that M_{1i} behaves as H_i outside S_{1i} and outputs $\perp$ inside S_{1i}. To be concise, we will say that $\mathcal{M}_1$ is a shadow oracle of $\mathcal{L}$ with respect to $(S_{11} \dots S_{1d})$. Why do we want this behaviour? For $S_i := H_{i-1}(\dots H_0(\Sigma) \dots)$, $\mathcal{M}_1$ clearly contains no information about $\tilde{H}$ on Σ, since $S_j \subseteq S_{1j}$. But why couldn't we just have chosen $(S_1 \dots S_d)$ instead of $(S_{11} \dots S_{1d})$ to define $\mathcal{M}_1$? Briefly, this is because choosing to hide an exponentially larger set (note that $|S_{11}| = |\Sigma|^{d+1}$ while $|S_1| = |\Sigma|$) allows us to easily apply similar arguments in the subsequent hybrids. This will become evident shortly. Recalling our goal, we want to establish that $\mathcal{L} \circ U_1 \circ \rho_0$ and $\mathcal{M}_1 \circ U_1 \circ \rho_0$ are close in trace distance. To do this, we use the so-called one-way to hiding (O2H) lemma [AHU19]. Informally, the lemma, as applied to our situation, says that if (a) the input state ρ_0 contains no information about the set where $\mathcal{L}$ and $\mathcal{M}_1$ behave differently, and (b) the probability of finding any element inside this set is negligible, then the trace distance between the two states of interest is negligible. The lemma clearly applies in our case because (a) initially the algorithm contains no information about $\mathcal{L}$ (it has not yet made any queries) and (b) the probability of finding any element in the set S_{1i} where $\mathcal{L}$ and $\mathcal{M}_1$ behave differently, without knowing anything about $\mathcal{L}$, is at most $|S_{1i}|/|S_{0i}| = \mathrm{negl}(\lambda)$, for each $i \in \{1 \dots d\}$, and thus still negligible by a union bound.

Hybrid 2. $\mathcal{L} \circ U_2 \circ \rho_1 \approx \mathcal{M}_2 \circ U_2 \circ \rho_1$ where $\rho_1 = \mathcal{M}_1 \circ U_1 \circ \rho_0$.
In this step, we will see the advantage of having chosen a sequence of sufficiently large sets $(S_{11}, \dots S_{1d})$ where $\mathcal{M}_1$ outputs $\perp$. Let us begin with examining the information contained in ρ_1 about $\mathcal{L}$. In the previous case, ρ_0 contained no information about $\mathcal{L}$. Since ρ_1 only learns about $\mathcal{L}$ by querying $\mathcal{M}_1$, it suffices to examine the information contained in $\mathcal{M}_1$. Since $\mathcal{M}_1$ does not hide any information about H_0, ρ_1 could have learnt $S_1 = H_0(\Sigma)$. Recall also that $S_1 \subseteq S_{11}$. This means that if one were to take $\mathcal{M}_2$ equal to $\mathcal{M}_1$, then one cannot expect $\mathcal{L} \circ U_2 \circ \rho_1$ to be close to $\mathcal{M}_2 \circ U_2 \circ \rho_1$ in general because U_2 could query the oracle at S_1 and the outputs of the two circuits would be different with probability one—$\mathcal{M}_1$ outputs $\perp$ while $\mathcal{L}$ does not. Consequently, when constructing $\mathcal{M}_2$, we do not hide anything about H_1. As for $H_2 \dots H_d$, note that, $\mathcal{M}_1$ contains no information about the behaviour of $\mathcal{L}$ inside $S_{12}, S_{13} \dots S_{1d}$. We can therefore, treat $S_{12} \dots S_{1d}$ as the new "base sets" and proceed analogously. Let $S_{22} \subseteq S_{12}$ be a random subset of S_{12}, subject to the constraint (as before) that (a) it includes $S_2 = H_1(H_0(\Sigma))$ and (b) $|S_{22}|/|S_{12}| = 1/|\Sigma| = \mathrm{negl}(\lambda)$. Defining $\mathcal{M}_2$ to be the shadow oracle of $\mathcal{L}$ with respect to $(\varnothing, S_{22}, \dots S_{2d})$, one can again apply the O2H lemma to conclude that $\mathcal{L} \circ U_2 \circ \rho_1$ and $\mathcal{M}_2 \circ U_2 \circ \rho_1$ are close in trace distance. Note that it is crucial that $|S_{12}|$ is sufficiently large such that condition (b) above is satisfied.

Generalising the argument above, one sees that the sets S_{ij} constitute a triangular matrix (where the i-th row corresponds to sets on which $\mathcal{M}_i$ outputs $\perp$)

$$\begin{bmatrix} S_{11} & H_1(S_{11}) & H_2(H_1(S_{11}) & \dots & H_d(\dots H_1(S_{11}) \dots) \\ \varnothing & S_{22} & H_2(S_{22}) & \dots & H_d(\dots H_2(S_{22}) \dots) \\ \varnothing & \varnothing & S_{33} & \dots & H_d(\dots H_3(S_{33}) \dots) \\ & & & \ddots & \\ \varnothing & \varnothing & \varnothing & & S_{dd} \end{bmatrix}$$

which clarifies why the argument can only be applied for d steps (as we expect). To see this, note that at the dth step, all oracles except the last have been completely revealed (last row). Crucially, the last oracle is blocked at $S_d \subseteq S_{dd}$ and therefore reveals no information about $\tilde{H}(\Sigma)$. If one proceeds with the $(d+1)$-th step, all oracles are revealed and one can no longer argue that the algorithm does not access $\tilde{H}(\Sigma)$.

Observe that so far, we have not used the fact that CodeHashing is classically hard, only that without access to the oracle, the problem cannot be solved. The classical hardness comes into play once BPP computations are allowed.

2.1.3 d-CodeHashing $\notin$ $\mathsf{QNC}_d^{\mathsf{BPP}}$

We now sketch how one goes from arguing d-CodeHashing $\notin$ QNC_d to arguing d-CodeHashing $\notin$ $\mathsf{QNC}_d^{\mathsf{BPP}}$. Denote circuits corresponding to $\mathsf{QNC}_d^{\mathsf{BPP}}$ by $\mathcal{A}_{d+1} \circ \mathcal{B}_d^{\mathcal{L}} \circ \dots \circ \mathcal{B}_1^{\mathcal{L}} \circ \rho_0$ where $\mathcal{B}_i^{\mathcal{L}} := \Pi_i \circ \mathcal{L} \circ U_i \circ \mathcal{A}_i^{\mathcal{L}}$, $\mathcal{A}_i^{\mathcal{L}}$ denotes a classical algorithm, and Π_i denotes a (possibly partial) measurement. The analogous circuit with shadow oracles is denoted by $\mathcal{A}_{d+1} \circ \mathcal{B}_d^{\mathcal{M}_d} \circ \dots \mathcal{B}_1^{\mathcal{M}_1} \circ \rho_0$ where $\mathcal{B}_i^{\mathcal{M}_i} := \Pi_i \circ \mathcal{M}_i \circ U_i \circ \mathcal{A}_i^{\mathcal{L}}$. The idea, again, is to establish, via a hybrid argument, that the two circuits are close in trace distance. In the QNC_d case, thanks to the depth of the circuit being d, we were able to argue that any QNC_d algorithm behaves equivalently if we take away its access to $\tilde{H}$. When trying to argue that a $\mathsf{QNC}_d^{\mathsf{BPP}}$ algorithm cannot solve the problem, we have to be more careful because the BPP part has sufficient depth to make queries to $\tilde{H}$. In our argument, this will affect how the shadow oracles $\mathcal{M}_i$ are defined.

In some more detail, we allow the classical algorithm to make "path queries"—which intuitively just means that if H_i is queried at x_i, the algorithm also learns $(x_0, x_1 \dots x_d)$ such that[14] $x_{j+1} = H_j(x_j)$ for all j. This of course can only help the algorithm.

The key idea is that we account for the "paths" that have been queried classically until depth i and define $\mathcal{M}_i$ to be consistent with those (i.e. it never outputs $\perp$ on these paths). As before, we can replace queries to $\mathcal{L}$ with queries to $\mathcal{M}_i$ that contain no information about $\tilde{H}$ except for the paths which were classically queried. Appealing to the soundness of CodeHashing, such an algorithm cannot succeed. This is because CodeHashing has the property that even an unbounded classical algorithm cannot succeed if it only makes polynomially many queries to the oracle.

2.1.4 d-CodeHashing $\notin$ $\mathsf{BPP}^{\mathsf{QNC}_d}$

Observe that a poly depth quantum circuit can access $\tilde{H}$ and since a $\mathsf{BPP}^{\mathsf{QNC}_d}$ circuit has poly many QNC_d circuits, it is not a priori clear that $\mathsf{BPP}^{\mathsf{QNC}_d}$ cannot also access $\tilde{H}$. This is why the approach we used to prove that d-CodeHashing $\notin$ QNC_d cannot be applied directly. Crucially, to argue that the problem is not in $\mathsf{BPP}^{\mathsf{QNC}_d}$, one must use the fact that the contents of each QNC_d circuit are measured entirely, and that each QNC_d circuit takes only classical inputs. In order to handle the classical information that each QNC_d circuit receives as input, we use a technique called the "sampling argument". In essence, this says that if $\mathcal{L}$ has high entropy (which is to say that the oracles being queried are sufficiently random), then conditioned on any string s correlated with it, the resulting $\mathcal{L}|s$ behaves as a "convex combination" of high entropy distributions with a small fraction of their values completely fixed. This allows us to reduce the analysis to that of a particular set of paths being exposed, which we can handle by proceeding as in the $\mathsf{QNC}_d^{\mathsf{BPP}}$ case.

A similar argument was used by CCL to establish that a problem is not in $\mathsf{BPP}^{\mathsf{QNC}_d}$ with respect to a (structured) oracle. Their analysis used a sequence of permutation oracles and was simplified by viewing the oracles, equivalently, as distributions over paths (as opposed to a sequence of functions assigning values to individual points). The paths viewpoint was particularly helpful when considering the "sampling argument" (the version we use is derived from [CDGS18]). [AGS22] showed that such a sampling argument can be obtained for almost any oracle which can be viewed as a distribution over paths. In our setting, since the oracles are random, paths can collide. Thus, one needs to define a suitable notion of "paths" in this setting. We provide more details in the next three paragraphs. However, since these are relatively more technical, one may wish to skip directly to Subsection 2.1.5 on a first read.

Sampling argument for Permutations. Suppose t is a permutation over N elements labelled $\{0, \dots, N-1\}$. This permutation t is ordinarily viewed as a function, $t(x)$ specifying how x is mapped. However, one could equivalently view t as a collection of pairs (or tuples later) (x, y) such that $t(x) = y$. We call such a pair a "path".

[14] Two caveats: (1) $H_0 : \Sigma \to \Sigma^{d'}$ therefore some of the paths will not have well defined first components and (2) we only care about queries made inside the base sets where conditioned on E, $H_1 \dots H_{d-1}$ behave as permutations.

Now consider distributions over permutations. Let's begin with a uniform distribution $\mathbb{F}$ over all permutations u. One may characterise $\mathbb{F}$ as follows: for any $u \sim \mathbb{F}$, i.e. any u sampled from $\mathbb{F}$, it holds that $\Pr[u(x) = y] = \Pr[(x, y) \in \text{paths}(u)]$.

We first state a basic version of the sampling argument. To this end, we define a (p, δ) *non-uniform distribution*, $\mathbb{F}^{(p,\delta)}$, which is closely related to the uniform distribution $\mathbb{F}$. At a high level, $\mathbb{F}^{(p,\delta)}$ is "δ close to" $\mathbb{F}$ with at most p many paths fixed. What does "δ closeness" mean? Let $\Pr[S \subseteq \text{paths}(u)]$ denote the probability that a collection S of (non-colliding) paths is in u. Then, for any distribution $\mathbb{G}$ (over permutations), a distribution $\mathbb{G}^{\delta}$ is δ close to it if the following holds: when $t' \sim \mathbb{G}^{\delta}$ and $t \sim \mathbb{G}$, one has $\Pr[S \subseteq \text{paths}(t')] \le 2^{\delta |S|} \Pr[S \subseteq \text{paths}(t)]$ for all S.

We are almost ready to state the basic sampling argument. We need the notion of a "convex combination" of random variables. We say a random variable (such as our permutation) t is a convex combination of random variables t_i, denoted by $t \equiv \sum_i \alpha_i t_i$ (where $\sum_i \alpha_i = 1$ and $\alpha_i \ge 0$), if the following holds for all t': $\Pr[t = t'] = \sum_i \alpha_i \Pr[t_i = t']$.

Informally, the basic sampling argument is a statement about a uniform permutation $u \sim \mathbb{F}$ and how the distribution $\mathbb{F}$ changes if we are given some "advice" about this permutation which is simply a function $g(u)$. Roughly speaking, given that $g(u)$ evaluates to r with probability at least 2^{-m}, the distribution $\mathbb{F}$ conditioned on r is a convex combination[15] of $\mathbb{F}^{(p,\delta)}$ distributions where the number of paths fixed is at most $p = 2m/\delta$. Here δ is a free parameter. We slightly abuse the notation and write this basic sampling argument as

$$\mathbb{F}|r \equiv \text{conv}(\mathbb{F}^{(p,\delta)}).$$

If we view $g(u)$ as the output of the first quantum part of the circuit for $\mathsf{BPP}^{\mathsf{QNC}_d}$, and u as the oracle of interest (details are in the next section), it is suggestive that $u|g(u)$ will be the oracle for the second quantum part of the circuit. We can use the sampling argument above and re-use our analysis because $\mathbb{F}$ and $\mathbb{F}^{(p,\delta)}$ have very similar statistical properties. However, it is unclear how to use the sampling argument thereafter as the basic sampling argument seems to only apply to $\mathbb{F}$ (and not to $\mathbb{F}^{(p,\delta)}$). It turns out that one can extend the sampling argument to obtain

$$\mathbb{F}^{(p',\delta')}|r \equiv \text{conv}(\mathbb{F}^{(p+p',\delta'+\delta)}).$$

Consequently, if the procedure is successively applied $\tilde{n} \le \text{poly}(n)$ times (starting with $\mathbb{F}$), the convex combination would be over distributions of the form $\mathbb{F}^{(\tilde{n}p,\tilde{n}\delta)}$. The parameters can be appropriately chosen to ensure that at most polynomially many paths are exposed but we omit the details in this overview.

Sampling argument for Injective Shufflers. The proofs of the previously mentioned statements do not rely on any special property of the distribution $\mathbb{F}$ nor do they depend on the fact that we were considering permutations. Any object for which we can describe a "reasonable" notion of "paths" admits such a sampling argument. Therefore, as we did for permutations, to describe the sampling argument, we change our viewpoint and consider "paths" in $\mathcal{L} = (H_0, \dots H_d)$ instead of individual values taken by the H_i's. Recall that a "path" was a tuple of the form $(x_0, x_1 \dots)$ such that $x_i = H_{i-1}(x_{i-1})$ for all i.

This viewpoint is inadequate for capturing the probabilistic behaviour of $\mathcal{L}$ due to two reasons (which are not hard to rectify). *First*, since $H_0 : \Sigma \to \Sigma^{d'}$, it is clear that at least $\left|\Sigma^{d'-1}\right|$ many points will never be contained in any "path" as described above. Therefore the behaviour of most points in H_i (for $i \in \{1 \dots d\}$) will not be captured by the "paths" viewpoint. *Second*, even though H_i maps $\Sigma^{d'} \to \Sigma^{d'}$ for $i \in \{1, \dots d-1\}$, H_i may not be injective and therefore the paths might collide, which again would mean the behaviour of many points would not be captured by the "paths" viewpoint.

To rectify the *second* issue, we can select base sets $(S_{01}, \dots S_{0d}) =: \bar{S}_0$ and condition on the event E. Since in our proofs, we only care about the behaviour of $\mathcal{L}$ on $\bar{S}_0$, it suffices to restrict our attention to $\bar{S}_0$. Recall that $\mathcal{L}|E$ behaves as a permutation on $\bar{S}_0$. Therefore no "path" inside $\bar{S}_0$ collides. To rectify the *first* issue, we consider two kinds of paths—Type 0 paths and Type 1 paths.[16] A *Type 0 path* is what we described earlier: a tuple of the form $(x_0, x_1 \dots)$ such that $x_i = H_{i-1}(x_{i-1})$ for all i. A *Type 1 path* is a tuple of the form $(\llcorner\!\lrcorner, x_1, x_2 \dots)$ such that $x_1 \notin H_0(\Sigma)$ (i.e. $\nexists x_0$ st $H_0(x_0) = x_1$) and $x_i = H_{i-1}(x_{i-1})$ for all $i \in \{2, 3 \dots\}$.

Observe that, restricted to $\bar{S}_0$ and conditioned[17] on E, we have the following equivalence: given $\Pr[H_i(x) = x']$ for all i, x and x', one can compute the probability associated with both types of paths and conversely, given probabilities associated with the paths, one can compute $\Pr[H_i(x) = x']$ for all i, x and x'.

[15] In the convex combination, there is a small component, of weight at most 2^{-m}, of some arbitrary distribution.

[16] The 0 and 1 represent where the first non-$\llcorner\!\lrcorner$ component sits.

[17] Recall, E is the event that the oracles H_0 and $H_1 \dots H_d$ are injective on Σ and $\bar{S}_0$ resp.

As is evident, working with $\mathcal{L}$ directly is cumbersome and we therefore define a simpler object, the *injective shuffler*. Fix sets $S_{0i} \subseteq \Sigma^{d'}$ of size $|\Sigma^{d+2}|$ for all $i \in \{1, \dots d\}$. Let $H'_0 : \Sigma \to S_{01}$, $H'_i : S_{0i} \to S_{0,i+1}$ for all $i \in \{1, \dots d-1\}$ be injective functions and let $H'_d : S_{0d} \to \{0,1\}^n \cup \{\perp\}$ (which may not be injective) such that H'_d outputs $\perp$ for all paths originating from Σ (and no other).[18] We define the *injective shuffler*, $\mathcal{K}$ as $(H'_0, \dots H'_d)$.

Think of $\mathcal{K}$ as a simpler way to denote the relevant object associated with $\mathcal{L}|E$. What do we mean by the relevant object—not only is it injective, it also never reveals any information[19] about the values taken by $\tilde{H}$ in Σ. As alluded to at the beginning of this subsection, since the strings s_i arise from quantum parts which only get access to $\mathcal{L}$ via shadow oracles, the sampling argument only needs to be applied to parts of $\mathcal{L}$ outside of paths in $\tilde{H}$.

To state the sampling argument for the injective shuffler, we define (p, δ) non-β-uniform distributions $\mathbb{F}_{\text{inj}}^{(p,\delta)|\beta}$ for the injective shuffler (analogous to the way we defined them for permutations). We begin with the uniform distribution—it is simply a distribution which assigns equal probabilities to all the possible injective shufflers, given the sets $(S_{0i})_i$. As for β-uniform distributions, $\mathbb{F}_{\text{inj}}^{|\beta}$, we first need to define the "paths", β. Here, β will again be a set of "non-colliding paths" but formalising this requires some care (see Subsection 7.6.2). Then a β-uniform distribution is the same as the uniform distribution except that the paths in β are fixed. Omitting further details, one can define $\mathbb{F}^{(p,\delta)|\beta}$ to be a distribution which is "δ close to" the β-uniform distribution with at most p many paths fixed (in addition to β).

The sampling argument for injective shufflers is the following. Suppose we start with $t \sim \mathbb{F}_{\text{inj}}^{\delta'|\beta}$ (i.e. a distribution which is "δ' close to" β-uniform) and are given some advice $h(t)$ which happens to be r with probability at least 2^{-m}. Then the distribution $\mathbb{F}_{\text{inj}}^{\delta'|\beta}$ conditioned on r is, roughly speaking, a convex combination[20] of $\mathbb{F}_{\text{inj}}^{(p,\delta+\delta')|\beta}$ distributions where the number of paths fixed (in addition to β) is at most $p = 2m/\delta$ and δ again is a free parameter. Using the previous shorthand, we have

$$\mathbb{F}_{\text{inj}}^{\delta'|\beta}|r \equiv \text{conv}(\mathbb{F}_{\text{inj}}^{(p,\delta+\delta')|\beta}).$$

Stitching everything together As asserted before we described the sampling argument, one can replace all the oracles $\mathcal{L}$ in the quantum part of the circuit for $\mathsf{BPP}^{\mathsf{QNC}_d}$ with appropriate shadow oracles. Let $\mathcal{M}_{11}, \dots \mathcal{M}_{1d}$ denote the shadow oracles for the first quantum part, $\mathcal{M}_{21} \dots \mathcal{M}_{2d}$ for the second quantum part and so on. Suppose the paths queried by the ith classical part were β_i, the string outputted by the ith quantum part be s_i. Suppose $\mathcal{M}_{11} \dots \mathcal{M}_{1d} \dots \mathcal{M}_{i-1,1} \dots \mathcal{M}_{i-1,d}$ have been specified. Now, conditioned on s_i, the sampling argument says $\mathcal{L}|s_i$ behaves as a convex combination of injective shufflers with certain paths exposed, when restricted to base sets. Let $\beta(s_i)$ be the random variable which specifies these paths and occurs with the weights specified in the convex combination. One can define $\mathcal{M}_{i1} \dots \mathcal{M}_{id}$ as in the QNC_d case, ensuring the paths $\beta_1 \dots \beta_{i-1}$ and $\beta(s_1) \dots \beta(s_{i-1})$ have been exposed. Note crucially that s_i is obtained by a quantum part which only had access to $\mathcal{L}$ via shadow oracles so it does not change the distribution over $\tilde{H}$ (except for polynomially many paths which were already exposed, $\beta_1 \dots \beta_{i-1}$ and $\beta(s_1) \dots \beta(s_{i-1})$). Using a hybrid argument as in the QNC_d case, and using properties of the injective shuffler which is "δ close" to being uniform, one can apply the O2H lemma and conclude that the hybrids (again, defined as in the QNC_d case) are close in trace distance. Eventually, this yields that the initial circuit is close in trace distance to the circuit which only accesses $\mathcal{L}$ via the shadows $\mathcal{M}_{11} \dots \mathcal{M}_{1d} \dots \mathcal{M}_{m1} \dots \mathcal{M}_{md}$ in the quantum part (denote the number of quantum parts by $m \leq \text{poly}(\lambda)$). The latter circuit cannot solve d-$\mathsf{CodeHashing}$ again, because $\tilde{H}$ is only accessed by the classical parts of this circuit. More precisely, $\tilde{H}$ is only queried at at most $|\beta_1 \cup \dots \beta_m \cup \beta(s_1) \cup \dots \beta(s_m)| \leq \text{poly}(\lambda)$ locations and therefore the whole circuit can be simulated while only making polynomially many classical queries to $\tilde{H}$. From the soundness of $\mathsf{CodeHashing}$, this entails d-$\mathsf{CodeHashing}$ cannot be solved.

2.1.5 d-$\mathsf{CodeHashing} \notin \mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$

Just as the analysis of the $\mathsf{BPP}^{\mathsf{QNC}_d}$ case built on the QNC_d case, one can analyze the $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$ case by building on the $\mathsf{QNC}_d^{\mathsf{BPP}}$ case. While the high level idea stays the same, the details are more involved. This is partly because, in the QNC_d case, one could construct the shadow oracles $\mathcal{M}_1 \dots \mathcal{M}_d$ "all at once" since

[18] i.e. $H'_d(x_d) = \perp$ iff $(x_0, x_1, \dots x_d, x_{d+1})$ is a Type 0 path (therefore $x_{d+1} = \perp$).

[19] Except for polynomially possibly many paths exposed by classical queries; we handle these shortly.

[20] Again, neglecting a component with weight at most 2^{-m}.

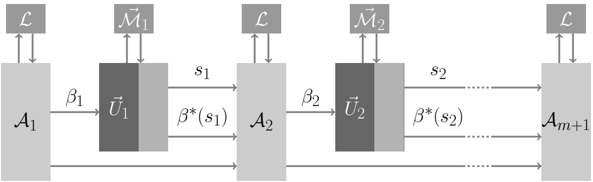


Figure 2: Here $\vec{\mathcal{M}}_i$ denotes the shadow oracles $(\mathcal{M}_{i1}, \dots \mathcal{M}_{id})$.

we were assuming the "worst case", i.e. the quantum algorithm learns everything there is to learn from the shadow oracles. However, in the $\mathsf{QNC}_d^{\mathsf{BPP}}$ case, to define $\mathcal{M}_i$, one had to know the behaviour of the classical algorithms in the hybrid circuits which involved $\mathcal{M}_1 \dots \mathcal{M}_{i-1}$ (in particular one has to know the "paths" that have been exposed). We show how one can account for this, but we leave the details to the main body.

2.1.6 Proof of quantum depth

In this subsection, we discuss how our complexity-theoretic separations also yield protocols for certifying quantum depth, i.e. *proofs of quantum depth*, in a way that is insensitive to classical polynomial depth. First, let us be a bit more precise about what we mean by proof of quantum depth.

Definition (informal)**.** A proof of d quantum depth is a two-message protocol involving two parties, a verifier and a prover. Both parties are assumed to have access to the random oracle H. The verifier is a PPT machine. The protocol satisfies the following, where λ is the security parameter.

- Completeness: There is a prover in BQP which makes the verifier accept with probability $1 - \text{negl}(\lambda)$.
- Soundness: No prover in $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$ makes the verifier accept with probability more than $\text{negl}(\lambda)$.

Let d be at most a fixed polynomial. Since d-$\mathsf{CodeHashing}$ is in NP, it immediately yields a proof of d quantum depth.

We conclude this discussion by illustrating the subtlety of considering proofs of quantum depth with more than two messages. Consider the following protocol.

Example 12. The verifier, Alice, prepares BB84 states $|b_i\rangle_{\theta_i} := H^{\theta_i} |b_i\rangle$ (b_i, θ_i are both chosen uniformly at random) for $i \in \{1, \dots n\}$ where H is the Hadamard operation (not to be confused with the random oracle). She sends them all to the prover, Bob.

Alice and Bob then engage in an n round protocol. In the i-th round, Alice sends θ_i and Bob sends b'_i. Alice accepts if $b_1 = b'_1, \dots b_n = b'_n$.

In this example,[21] it is not hard to see that Bob has to have n layers of unitaries. Could this simple construction already constitute a proof of quantum depth? Consider the following observations.

- *Spoofed by n single quantum depth devices.* It is easy to see that Bob can pass this test using n-many single-qubit quantum devices, each of which need only apply one quantum gate and make one computational basis measurement. The protocol works by simply delaying the application of the quantum gate and subsequent measurement. It is therefore difficult to call this a proof of quantum depth in any meaningful way.
- *Interaction seems superfluous.* The only use of the interaction is to introduce a delay. The same effect could be achieved with a single round protocol where Alice delays sending her message. Therefore, this procedure, at best, certifies "idle coherence" time.

The example shows how defining quantum depth in interactive settings can be quite subtle. We refer the reader back to the discussion in Section 1.3 for our proposal of what this definition should be.

[21] While we used quantum communication in the protocol, one could (using known results) delegate the production of these states to the prover (under computational assumptions) and run a similar protocol using classical communication.

2.1.7 Tighter upper bounds

Ideally, one would like to show the more fine-grained separation $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}} \subsetneq \mathsf{BPP}^{\mathsf{QNC}_{d+1}^{\mathsf{BPP}}}$. Since the best known algorithm for solving YZ's $\mathsf{CodeHashing}$ uses polynomial depth, d-$\mathsf{CodeHashing}$ inherits this limitation. We overcome this limitation and show the following.

Theorem 13. *Relative to a random oracle,* $\mathsf{QNC}_{2d+\mathcal{O}(1)} \not\subseteq \mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$ *which implies* $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}} \subsetneq \mathsf{BPP}^{\mathsf{QNC}_{2d+\mathcal{O}(1)}^{\mathsf{BPP}}}$.

We obtain the above by instantiating our lifting procedure, d-$\mathsf{Rec}[\cdot]$, with a variant of the proof of quantumness from [BKVV20], which we refer to as $\mathsf{CollisionHashing}$ (see Table 3). It is straightforward to show that $\mathsf{CollisionHashing}$ also satisfies classical query soundness by using the main argument in [BKVV20] and the query lower bound for finding collisions proved in [AS04].

Let g be a $2 \to 1$ function for which it is hard to find a collision. Then, the (slightly simplified) problem is to produce a pair (y, r) such that $r \cdot (x_0 \oplus x_1) \oplus H(x_0) \oplus H(x_1) = 0$ where $\{x_0, x_1\} \in g^{-1}(y)$. This problem can be solved in $\mathsf{QNC}_{\mathcal{O}(1)}$ (assuming that calls to g take only depth 1) by preparing the superposition $\sum_x |g(x)\rangle |x\rangle$, measuring the second register in the standard basis, and the first in the Hadamard basis.

We said simplified because in $\mathsf{CollisionHashing}$, g is in fact a uniformly random function g (treated as an oracle) with a domain twice as large as the co-domain. Note that this is not a $2 \to 1$ function in general. However, with overwhelming probability, a constant fraction of the elements in the co-domain has exactly two pre-images. Then, we require a pair (y, r) such that either y has exactly two pre-images and (y, r) satisfies the "equation", or y does not have exactly two pre-images. The limitation of $\mathsf{CollisionHashing}$ is that solutions to the problem are not verifiable, so the problem cannot be used to obtain a fine-grained proof of quantum depth.

2.2 Separations of hybrid quantum depth classes

2.2.1 Establishing $\mathsf{BPP}^{\mathsf{QNC}_{\mathcal{O}(1)}} \not\subseteq \mathsf{QNC}_d^{\mathsf{BPP}}$.

We describe our second lifting procedure, called d-$\mathsf{Ser}[\cdot]$. This takes any problem $\mathcal{P} \notin \mathsf{BPP}$ (relative to a random oracle) that satisfies offline soundness, and produces a new problem d-$\mathsf{Ser}[\mathcal{P}] \notin \mathsf{QNC}_d^{\mathsf{BPP}}$ (see Lemma 7).

Denote by R_H the set of solutions to $\mathcal{P}$ (defined with respect to H). Then, the key idea is simple. The problem d-$\mathsf{Ser}[\mathcal{P}]$ is to return a tuple $(c_0, c_1, \ldots, c_d)$ such that: c_0 is a solution to $\mathcal{P}$, i.e. $c_0 \in R_{H(\cdot)}$; c_1 is a solution to $\mathcal{P}$ but with respect to $H(c_0\|\cdot)$, i.e. $c_1 \in R_{H(c_0\|\cdot)}$, and similarly until c_d, which should be such that $c_d \in R_{H(c_0 \ldots c_{d-1}\|\cdot)}$.

To be a bit more concrete, take $\mathcal{P}$ to be $\mathsf{CollisionHashing}$. We know $\mathsf{CollisionHashing} \in \mathsf{QNC}_{\mathcal{O}(1)}$. Clearly, d-$\mathsf{Ser}[\mathsf{CollisionHashing}] \in \mathsf{BPP}^{\mathsf{QNC}_{\mathcal{O}(1)}}$. This is because $\mathsf{BPP}^{\mathsf{QNC}_{\mathcal{O}(1)}}$ allows one to run polynomially many $\mathsf{QNC}_{\mathcal{O}(1)}$ circuits. Consequently, one can use the first circuit to obtain the classical output c_0, use the second circuit to find c_1 and so on. On the other hand, intuitively, we expect that d-$\mathsf{Ser}[\mathsf{CollisionHashing}] \notin \mathsf{QNC}_d^{\mathsf{BPP}}$. This is because to solve the $(i+1)$-th sub-problem, one seems to require the solution to all of the previous i sub-problems. Since there are $d+1$ sub-problems in total, $\mathsf{QNC}_d^{\mathsf{BPP}}$ does not seem to suffice (here of course we are implicitly using the fact that $\mathcal{P} \notin \mathsf{BPP}$). Formally, the argument proceeds in a similar way as for the lifting map d-Rec in Subsection 2.1.3, except for one subtlety which is handled by requiring that the problem $\mathcal{P}$ satisfies the extra property of offline soundness. We refer the reader to the main text for more details. We remark that offline soundness follows from classical query soundness and therefore both $\mathsf{CollisionHashing}$ and $\mathsf{CodeHashing}$ satisfy it.

Problem	Additional Assumption	Verifiable	Classical Query Soundness	Offline Soundness	Completeness
$\mathsf{CodeHashing}$ [YZ]	None	Yes	Yes	Yes	BQP
$\mathsf{CollisionHashing}$	None	No	Yes	Yes	$\mathsf{QNC}_{\mathcal{O}(1)}$

Table 3: Problems in the random oracle model, which are intractable for BPP and used as building blocks for establishing quantum depth separations.

The immediate consequence of the existence of the lifting map $d\text{-}\mathsf{Ser}[\cdot]$ is that $\mathsf{BPP}^{\mathsf{QNC}_{\mathcal{O}(1)}} \nsubseteq \mathsf{QNC}_d^{\mathsf{BPP}}$ (first part of Theorem 4). However, we can also leverage $d\text{-}\mathsf{Ser}[\cdot]$, together with the separation from the next subsection, to show that $\mathsf{BPP}^{\mathsf{QNC}^{\mathsf{BPP}}_{\mathcal{O}(1)}} \nsubseteq \mathsf{BPP}^{\mathsf{QNC}_d} \cup \mathsf{QNC}_d^{\mathsf{BPP}}$ (Theorem 5). This is done as follows.

In Subsection 2.2.2, we introduce the problem d-$\mathsf{hCollisionHashing}$ (which also satisfies offline soundness), and argue that it is in $\mathsf{QNC}^{\mathsf{BPP}}_{\mathcal{O}(1)}$, but not in $\mathsf{BPP}^{\mathsf{QNC}_d}$. Now, applying the lifting map to it gives $d\text{-}\mathsf{Ser}[d\text{-}\mathsf{hCollisionHashing}] \notin \mathsf{BPP}^{\mathsf{QNC}_d} \cup \mathsf{QNC}_d^{\mathsf{BPP}}$. To obtain the containment, notice that d-Ser yields a problem that can be solved by solving $d+1$ many instances of the original problem. Thus, it follows that $d\text{-}\mathsf{Ser}[d\text{-}\mathsf{hCollisionHashing}] \in \mathsf{BPP}^{\mathsf{QNC}^{\mathsf{BPP}}_{\mathcal{O}(1)}}$.

2.2.2 Establishing $\mathsf{QNC}^{\mathsf{BPP}}_{\mathcal{O}(1)} \nsubseteq \mathsf{BPP}^{\mathsf{QNC}_d}$

This is the more surprising of the two hybrid separations, and its proof is more involved. In this section, we fix $d \le poly(\lambda)$. The problem that yields this separation is the following variation on $\mathsf{CollisionHashing}$: given access to a 2-to-1 function g[22], and to $H_0, \dots H_d$ (which specify h as $h = H_d \circ \cdots \circ H_0$), find a pair (y, r) such that

$$r \cdot (x_0 \oplus x_1) \oplus H(h(y)\|x_0) \oplus H(h(y)\|x_1) = 0,$$

where $\{x_0, x_1\} = g^{-1}(y)$. We refer to the new problem as d-$\mathsf{hCollisionHashing}$.

Without relying on h (that is, requiring that the equation to be satisfied is just $r \cdot (x_0 \oplus x_1) \oplus H(x_0) \oplus H(x_1) = 0$), this problem is the same as $\mathsf{CollisionHashing}$. This can be solved in $\mathsf{QNC}_{\mathcal{O}(1)}$ as follows:

(i) Evaluate g on a uniform superposition of inputs, obtaining $\sum_x |x\rangle |g(x)\rangle$,

(ii) Measure the image register obtaining some outcome y and a state $(|x_0\rangle + |x_1\rangle)|y\rangle$,

(iii) Query a phase oracle for H to obtain $((-1)^{H(x_0)}|x_0\rangle + (-1)^{H(x_1)}|x_1\rangle)|y\rangle$,

(iv) Make a Hadamard basis measurement of the first register, obtaining outcome r.

At a high level, in order to solve the new problem, which includes the evaluation of h as an input to H, one needs the ability to perform a (classical) depth d computation to evaluate $h(y)$ (since this requires the sequential evaluations of $H_0, \dots, H_d$). Note that a $\mathsf{QNC}^{\mathsf{BPP}}$ algorithm can solve this problem: the only modification to the algorithm described above is that, at step (iii), the algorithm first computes $h(y)$ (using polynomial classical computation), and then queries the oracle H on a superposition of $(h(y), x_0)$ and $(h(y), x_1)$. One can easily verify that this leads to a valid y, r for the problem.

Next, we give a sketch of how one can argue that the problem cannot be solved in $\mathsf{BPP}^{\mathsf{QNC}}$. The key technical ingredient is a "structure theorem" that characterizes the structure of efficient quantum strategies that are successful at $\mathsf{CollisionHashing}$. Our structure theorem applies equally to the proof of quantumness protocol from [BKVV20] (recall that the latter is just a version of collision hashing where g is replaced by a 2-to-1 trapdoor claw-free function).

Theorem 14 (informal)**.** *Let P be any* BQP *prover that succeeds with $1 - \text{negl}(n)$ probability at the proof of quantumness protocol from [BKVV20], by making q queries to the oracle H. Then, with $1 - \text{negl}(n)$ probability over pairs (H, y), the following holds. Let $p_{y|H}$ be the probability that P^H outputs y, and let x_0,x_1 be the pre-images of y. Then, for all $b \in \{0,1\}$, there exists $i \in [q]$ such that the state of the query register of P^H right before the i-th query has weight $\frac{1}{2} p_{y|H} \cdot (1 - \text{negl}(n))$ on x_b.*

See Corollary 113 for a formal statement of this result. This is a crucial strengthening of a Theorem from [CGV22], and employs the compressed oracle technique [Zha19]. A slight adaptation of this to our problem asserts that a successful strategy must be querying the random oracle H at a (close to) uniform superposition of $(h(y), x_0)$ and $(h(y), x_1)$.

Now let A be a $\mathsf{BPP}^{\mathsf{QNC}}$ algorithm that succeeds at d-$\mathsf{hCollisionHashing}$ with high probability and let q be the total number of queries to h made by the algorithm.

Then, one can show that, since the QNC part of the algorithm does not have sufficient depth to evaluate h (which is a sequential evaluation of $H_0, \dots, H_d$), we can assume, without loss of generality, the QNC part of A has no access to h. In other words, all of the queries to h are classical.

[22] Since we want our problem to be relative to a uniformly random oracle, in the formal description of the problem in the main text, we will not assume that g is exactly 2-to-1. Rather we will take g to be a uniformly random function with domain twice as large as the co-domain, and simply restrict our attention to y's in the co-domain that have exactly two pre-images (this is a constant fraction of the elements of the co-domain with overwhelming probability).

Now, Theorem 14 says essentially that, for any y, the only way to succeed with high probability (conditioned on that y being the output) is to query (with as much weight as the probability of outputting y) a uniform superposition of $(h(y), x_0)$ and $(h(y), x_1)$. However, observe that, for any y, the only way for A to query H (with a high weight) at a uniform superposition of $(h(y), x_0)$ and $(h(y), x_1)$ is to correctly guess the value of $h(y)$. Since this value is uniformly random for any algorithm that has not queried h at y, it follows that querying H at the uniform superposition of $(h(y), x_0)$ and $(h(y), x_1)$ must necessarily happen *after* the algorithm has already queried h on y.

This implies that there must exist an $i^* \in [q]$ such that, with high probability, A outputs y, r such that y is contained in the list of classical queries made to h up to the i^*-th query. Denote such a list by L_{i^*}. Moreover, with high probability over L_{i^*}, the continuation of A (from that point on) queries H at a uniform superposition of $(h(y), x_0)$ and $(h(y), x_1)$ for some $y \in L_{i^*}$. We show that such an algorithm A can be leveraged to extract a collision for g.

The key observation is that, since A is a $\mathsf{BPP}^{\mathsf{QNC}}$ algorithm, and all of the queries to h happen in the BPP portion of A, the "state" of algorithm A right after the i^*-th query to h is entirely *classical*. Thus, one can take a "snapshot" of the state of A at that point (i.e. copy it), and simply run *two independent executions* of A from that point on (with independent classical randomness). By what we argued earlier, with high probability, there exists $y \in L_{i^*}$, such that the execution of A from that point on, queries H at a uniform superposition of $(h(y), x_0)$ and $(h(y), x_1)$. Since the two executions are identical and independent, it follows that measuring the query registers of H in both executions will yield distinct pre-images of y with significant probability.

Finding collisions of g is of course hard (for any query-bounded quantum algorithm) [AS04]. Hence, this yields a contradiction.

3 Preliminaries

We state the preliminaries which are common to both parts in this section. Each part also has its own set of preliminary results.

3.1 Models of Computation

We first list the standard notation we use. PPT denotes a probabilistic polynomial time algorithm, QPT denotes a quantum polynomial time algorithm. As we primarily focus on search problems, to keep the presentation clean, we slightly abuse the notation and use decision class names to represent the corresponding search classes. For instance, we use BPP and BQP to denote the search classes FBPP and FBQP resp. which in turn are defined as follows.

Definition 15 ($\mathsf{FBPP}, \mathsf{FBQP}$; paraphrased from [Aar10; Aar13])**.** Let FBPP be the set of relations $R \subseteq \{0,1\}^* \times \{0,1\}^*$ such that for each R, there is a PPT algorithm $\mathcal{A}$ satisfying the following: for all input strings x,

$$\Pr[(x,y) \in R : y \leftarrow \mathcal{A}(x)] \geq 1 - o(1)$$

FBQP is defined analogously (PPT is replaced with QPT).[23]

Unlike the decision classes, it is unclear if changing the error from $o(1)$ to some constant (say 2/3rds) preserves the class. For our purposes, $o(1)$ suffices. We now define circuit models and the associated classes, depending on their depth; we drop the "F" prefix entirely.

Notation 16. A *single layer unitary*, is defined by a set of one and two-qubit gates which act on disjoint qubits (so that they can all act in parallel in a single step). The number of single layer unitaries in a circuit defines its *depth.*

Definition 17 (QNC_d circuits and QNC_d relations)**.** Denote by QNC_d the set of d-depth quantum circuits (see Figure 3a).

Define QNC_d to be the set of all relations $R \in \{0,1\}^* \times \{0,1\}^*$ which satisfy the following: for each relation $R \in \mathsf{QNC}_d$, there is a circuit family $\{\mathcal{C}_n : \mathcal{C}_n \in \mathsf{QNC}_d$ and acts on $\text{poly}(n)$ qubits$\}$ and for all strings x,

$$\Pr[(x,y) \in R \,:\, y \leftarrow \mathcal{C}_{|x|}(x)] \geq 1 - o(1).$$

Definition 18 ($\mathsf{QC_d}$ circuits and $\mathsf{QNC}_d^{\mathsf{BPP}}$ relations)**.** Denote by $\mathsf{QC_d}$ the set of all circuits which, for each $n \in \mathbb{N}$, act on $\text{poly}(n)$ qubits and bits and can be specified by

- d single layered unitaries, $U_1, U_2 \dots U_d$,
- $d+1$ $\text{poly}(n)$-sized classical circuits $\mathcal{A}_{c,1} \dots \mathcal{A}_{c,d}, \mathcal{A}_{c,d+1}$, and
- d computational basis measurements

that are connected as in Figure 3b.

Define $\mathsf{QNC}_d^{\mathsf{BPP}}$ analogously to QNC_d relations, replacing QNC_d circuits with $\mathsf{QC_d}$ circuits. When $d(n) = \text{polylog}(n)$, denote the set of relations by $\mathsf{QNC}^{\mathsf{BPP}}$.

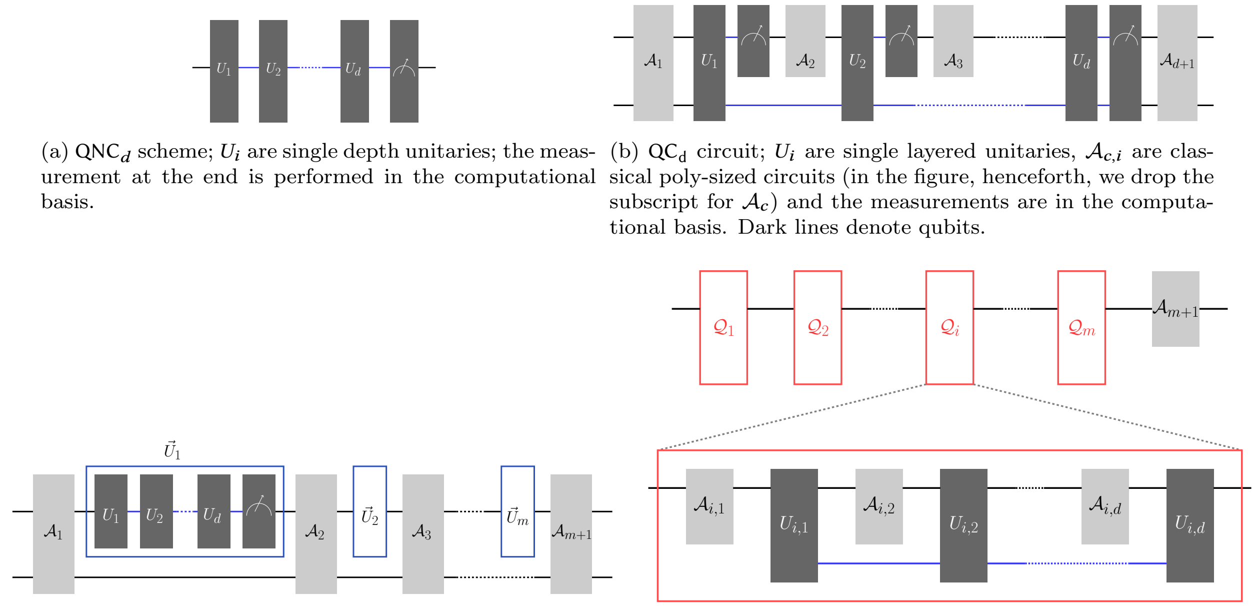
Definition 19 ($\mathsf{CQ_d}$ circuits and $\mathsf{BPP}^{\mathsf{QNC}_d}$ relations)**.** Denote by $\mathsf{CQ_d}$ the set of all circuits which, for each $n \in \mathbb{N}$ and $m = \text{poly}(n)$, act on $\text{poly}(n)$ qubits and bits and can be specified by

- m tuples of d single layered unitaries $(U_{1,i}, U_{2,i} \dots U_{d,i})_{i=1}^m$,
- $m+1$, $\text{poly}(n)$ sized classical circuits $\mathcal{A}_{c,1} \dots \mathcal{A}_{c,m}, \mathcal{A}_{c,m+1}$, and
- m computational basis measurements

that are connected as in Figure 3c.

Define, as above, $\mathsf{BPP}^{\mathsf{QNC}_d}$ analogously to QNC_d relations, replacing QNC_d circuits with $\mathsf{CQ_d}$ circuits. When $d(n) = \text{polylog}(n)$, denote the set of relations by $\mathsf{BPP}^{\mathsf{QNC}}$.

[23] NB: This, in particular, implies there is at least one y for every x, s.t. $(x,y) \in R$.



(a) QNC_d scheme; U_i are single depth unitaries; the measurement at the end is performed in the computational basis.

(b) $\mathsf{QC_d}$ circuit; U_i are single layered unitaries, $\mathcal{A}_{c,i}$ are classical poly-sized circuits (in the figure, henceforth, we drop the subscript for $\mathcal{A}_c$) and the measurements are in the computational basis. Dark lines denote qubits.

(c) $\mathsf{CQ_d}$ circuit; for clarity, we dropped the indices in $\mathcal{A}_c$ and the second indices in $U_{1,i}, U_{2,i} \ldots U_{d,i}$.

(d) CQC_d circuit; $\mathcal{Q}_i$ denotes ith $\mathsf{QC_d}$ circuit and $m = \text{poly}(n)$. The measurements after the single layer unitaries are included in $U_{i,j}$ with $j = 1, \cdots, d$. The final classical part is labelled $\mathcal{A}_{m+1}$ instead of $\mathcal{A}_{c,m+1,1}$ for simplicity.

Figure 3: The four circuit models we consider. We draw single wires to represent potentially polynomially many wires. Black lines and blue lines indicate wires carrying classical and quantum information, respectively. We implicitly follow this convention henceforth.

Definition 20 (CQC_d circuits and $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$ relations)**.** Denote by CQC_d the set of all circuits which, for each $n \in \mathbb{N}$ and $m = \text{poly}(n)$, which can be specified by m $\mathsf{QC_d}$ circuits acting on $\text{poly}(n)$ qubits and bits, that are connected as in Figure 3d.

Define, as above, $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$ analogously to QNC_d relations, replacing QNC_d circuits with CQC_d circuits. With $d(n) = \text{polylog}(n)$, denote the set of relations by $\mathsf{BPP}^{\mathsf{QNC}^{\mathsf{BPP}}}$.

Remark 21. Connection with the more standard notation: QNC_d has depth d and QNC^m has depth $\log^m(n)$, i.e. $\mathsf{QNC}^m = \mathsf{QNC}_{\log^m(n)}$.

Later, it would be useful to symbolically represent these three circuit models but we mention them here for ease of reference.

Notation 22. We use the following notation convention.

- Probability: The probability of an event E occurring, as a result of a process P, is denoted by $\Pr[E : P]$. In our context, the probability of a random variable X taking the value x when process Y takes place is denoted by $\Pr[x \leftarrow X : Y]$. When the process Y is just a sampling of X, we drop the Y and use $\Pr[x \leftarrow X]$.

- QNC_d: We denote a d-depth quantum circuit (see Definition 18 and Figure 3a) by $\mathcal{A} = U_d \circ \cdots \circ U_1$ and (by a slight abuse of notation) the probability that running the algorithm on all zero inputs yields x, by $\Pr[x \leftarrow \mathcal{A}]$ while that on some input state ρ by $\Pr[x \leftarrow \mathcal{A}(\rho)]$.

- $\mathsf{QC_d}$: We denote a $\mathsf{QC_d}$ circuit (see Definition 18 and Figure 3b) by $\mathcal{B} = \mathcal{A}_{c,d+1} \circ \mathcal{B}_d \circ \mathcal{B}_{d-1} \cdots \circ \mathcal{B}_1$ where $\mathcal{B}_i := \Pi_i \circ U_i \circ \mathcal{A}_{c,i}$ and "$\circ$" implicitly denotes the composition as shown in Figure 3b. As above, the probability of running the circuit $\mathcal{A}$ on all zero inputs and obtaining output x is denoted by $\Pr[x \leftarrow \mathcal{B}]$ while that on some input state ρ by $\Pr[x \leftarrow \mathcal{B}(\rho)]$.

- $\mathsf{CQ_d}$: We denote a $\mathsf{CQ_d}$ circuit (see Definition 19 and Figure 3c) by $\mathcal{C} = \mathcal{A}_{c,m+1} \circ \mathcal{C}_m \circ \cdots \circ \mathcal{C}_1$ where $\mathcal{C}_i := \Pi_i \circ U_{d,i} \circ \cdots \circ U_{1,i} \circ \mathcal{A}_{c,i}$ and "$\circ$" implicitly denotes the composition as shown in Figure 3c. Again, the probability of running the circuit $\mathcal{C}$ on all zero inputs and obtaining output x is denoted by $\Pr[x \leftarrow \mathcal{C}]$ while that on some input state ρ by $\Pr[x \leftarrow \mathcal{C}(\rho)]$.

- CQC_d: We denote a CQC_d circuit (see Definition 20 and Figure 3d) by $\mathcal{D} = \mathcal{A}_{c,m+1,1} \circ \mathcal{D}_m \circ \cdots \circ \mathcal{D}_1$ where $\mathcal{D}_i = \mathcal{B}_{i,d} \circ \mathcal{B}_{i,d-1} \circ \cdots \circ \mathcal{B}_{i,1}$ is a[24] QC_d circuit with $\mathcal{B}_{i,j} := \Pi_{i,j} \circ U_{i,j} \circ \mathcal{A}_{c,i,j}$ for $i, j \in \{1, \dots d\}$ and "$\circ$" implicitly denotes the composition as shown in Figure 3d.

3.1.1 The Oracle Versions

We consider the standard Oracle/query model corresponding to functions—the oracle returns the value of the function when invoked classically and its action is extended by linearity when it is accessed quantumly.

Notation 23. An oracle $\mathcal{O}_f$ corresponding to a function f is given by its action on "query" and "response" registers as $\mathcal{O}_f \left|x\right\rangle_Q \left|a\right\rangle_R = \left|x\right\rangle_Q \left|a \oplus f(x)\right\rangle_R$. An oracle $\mathcal{O}_{(f_i)_{i=1}^k}$ corresponding to multiple functions $f_1, f_2 \dots f_k$ is given by $\mathcal{O}_{(f_i)_{i=1}^k} \left|x_1, x_2 \dots x_k\right\rangle_Q \left|a_1, a_2 \dots a_k\right\rangle_R = \left|x_1, x_2 \dots x_k\right\rangle_Q \left|a_1 \oplus f_1(x_1), a_2 \oplus f_2(x_2), \dots a_k \oplus f_k(x_k)\right\rangle_R$.
When $\mathcal{O}_f$ is accessed classically, we use $\mathcal{O}_f(x)$ to mean it returns $f(x)$.

Remark 24 ($\mathsf{QNC}_d^{\mathcal{O}}$, ${\mathsf{QC}_\mathsf{d}}^{\mathcal{O}}$, ${\mathsf{CQ}_\mathsf{d}}^{\mathcal{O}}$). The oracle versions of QNC_d, QC_d and CQ_d circuits are as shown in Figures 4a, 4b, and 4c. We allow (polynomially many) parallel uses of the oracle even though in the figures we represent these using single oracles. We do make minor changes to the circuit models, following [CCL20] when we consider QNC_d circuits and CQ_d circuits—an extra single layered unitary is allowed to process the final oracle call.

We end by explicitly augmenting Notation 22 to include oracles.

Notation 25. When oracles are introduced, we use the following notation.

- $\mathsf{QNC}_d^{\mathcal{O}}$: $\mathcal{A}^{\mathcal{O}} = U_{d+1} \circ \mathcal{O} \circ U_d \circ \dots \mathcal{O} \circ U_1$ (see Figure 4a)

- ${\mathsf{QC}_\mathsf{d}}^{\mathcal{O}}$: $\mathcal{B}^{\mathcal{O}} = \mathcal{A}_{c,d+1}^{\mathcal{O}} \circ \mathcal{B}_d^{\mathcal{O}} \circ \dots \mathcal{B}_1^{\mathcal{O}}$ where $\mathcal{B}_i^{\mathcal{O}} = \Pi_i \circ \mathcal{O} \circ U_i \circ \mathcal{A}_{c,i}^{\mathcal{O}}$ and $\mathcal{A}_{c,i}^{\mathcal{O}}$ can access $\mathcal{O}$ classically (see Figure 4b).

- ${\mathsf{CQ}_\mathsf{d}}^{\mathcal{O}}$: $\mathcal{C}^{\mathcal{O}} = \mathcal{A}_{m+1}^{\mathcal{O}} \circ \mathcal{C}_m^{\mathcal{O}} \circ \dots \mathcal{C}_1^{\mathcal{O}}$ where $\mathcal{C}_i^{\mathcal{O}} := \Pi_i \circ U_{d+1,i} \circ \mathcal{O} \circ U_{d,i} \circ \cdots \circ \mathcal{O} \circ U_{1,i} \circ \mathcal{A}_{c,i}^{\mathcal{O}}$ where $\mathcal{A}_{c,i}^{\mathcal{O}}$ can access $\mathcal{O}$ classically (see Figure 4c).

- $\mathsf{CQC}_d^{\mathcal{O}}$: $\mathcal{C}^{\mathcal{O}} = \mathcal{A}_{c,m+1,1} \circ \mathcal{D}_m^{\mathcal{O}} \circ \dots \mathcal{D}_1^{\mathcal{O}}$ where $\mathcal{D}_i = \mathcal{B}_{i,d} \circ \dots \mathcal{B}_{i,1}$ with $\mathcal{B}_{i,j}^{\mathcal{O}} := \Pi_{i,j} \circ \mathcal{O} \circ U_{i,j} \circ \mathcal{A}_{c,i,j}^{\mathcal{O}}$ and $\mathcal{A}_{c,i,j}$ accesses $\mathcal{O}$ classically (see Figure 4d

The classes $\left(\mathsf{QNC}_d^{\mathsf{BPP}}\right)^{\mathcal{O}}, \left(\mathsf{BPP}^{\mathsf{QNC}_d}\right)^{\mathcal{O}}$ and $\left(\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}\right)^{\mathcal{O}}$ are implicitly defined to be the query analogues of $\mathsf{QNC}_d^{\mathsf{BPP}}, \mathsf{BPP}^{\mathsf{QNC}_d}$ and $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$ (resp.), i.e. class of relations solved by ${\mathsf{QC}_\mathsf{d}}^{\mathcal{O}}, {\mathsf{CQ}_\mathsf{d}}^{\mathcal{O}}$ and $\mathsf{CQC}_d^{\mathcal{O}}$ circuits (resp.).

3.2 The Random Oracle Model

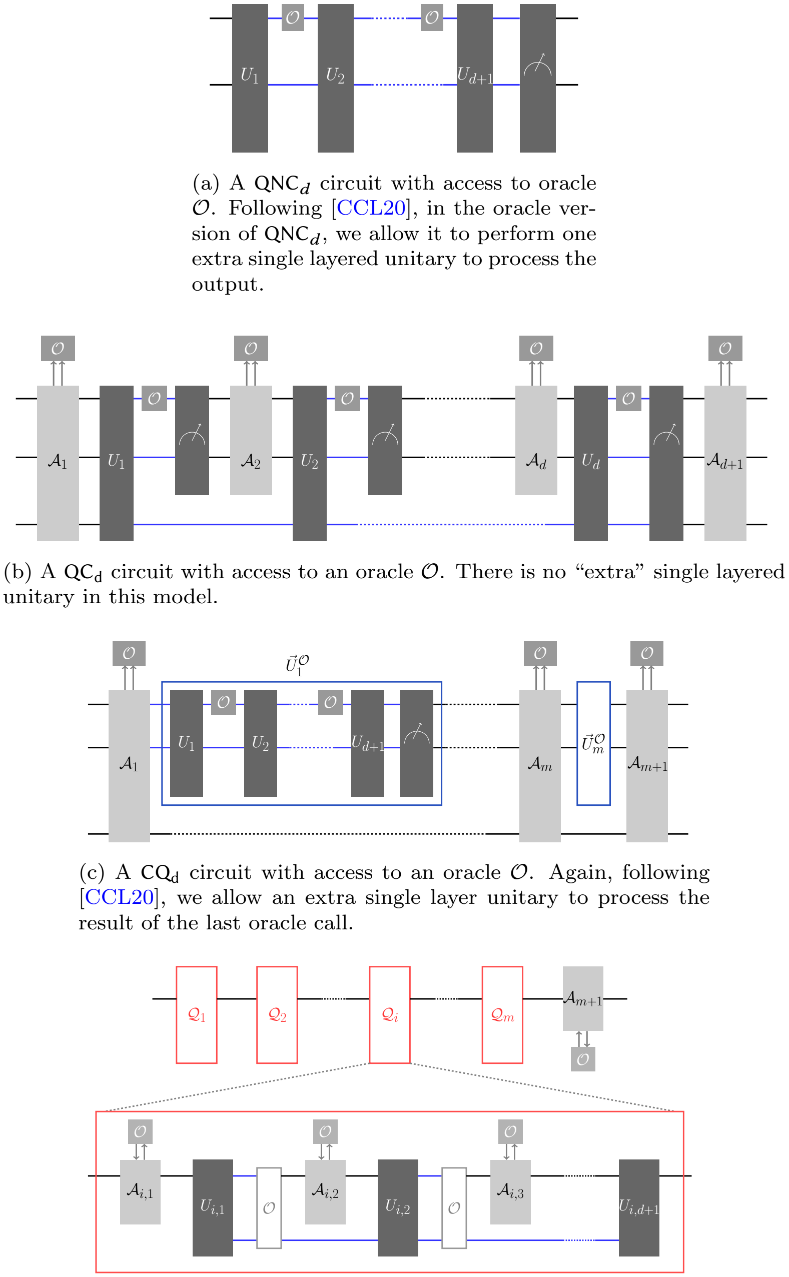
In the random oracle model, all parties are given access to a random function H which is defined from $\{0,1\}^* \to \{0,1\}$ s.t. it assigns 0 or 1 to each input x, independently with probability half. Quantum algorithms can access H in superposition.

3.2.1 Domain Splitting

Using domain splitting, one can efficiently construct expanding (compressing resp.) random functions, i.e. uniformly random functions H' from $\{0,1\}^n \to \{0,1\}^m$ where $m \geq n$ ($m \leq n$ resp.) using H. By efficiently (wrt n) we mean in time $\mathrm{poly}(n)$ which translates to $m \leq \mathrm{poly}(n)$. More precisely, one can define $H'(x) := (H(x\|0), H(x\|1), \dots H(x\|m))$ where $\|$ denotes concatenation and the second part of the string has length at most $\log(m)$. Similarly, one can construct polynomially[25] many distinct random compressing/expanding functions from H. One can therefore use such random functions without loss of generality in the random oracle model.

[24] except we excluded the last classical circuit $\mathcal{A}_{c,i,d+1}$. This is without loss of generality because $\mathcal{A}_{c,i,d+1}$ can be absorbed in the first classical circuit, $\mathcal{A}_{c,i+1,1}$, of $\mathcal{D}_{i+1}$.

[25] In fact, exponentially many as we only need to polynomially many bits to index them.



(a) A QNC_d circuit with access to oracle $\mathcal{O}$. Following [CCL20], in the oracle version of QNC_d, we allow it to perform one extra single layered unitary to process the output.

(b) A $\mathsf{QC_d}$ circuit with access to an oracle $\mathcal{O}$. There is no "extra" single layered unitary in this model.

(c) A $\mathsf{CQ_d}$ circuit with access to an oracle $\mathcal{O}$. Again, following [CCL20], we allow an extra single layer unitary to process the result of the last oracle call.

(d) A CQC_d circuit with access to oracle $\mathcal{O}$.

Figure 4: The same four circuit models, but with oracle access.

3.2.2 Oracle Independent, Uniform and Non-Uniform Adversaries

We consider three kinds of adversaries (circuit families $\{\mathcal{C}_n\}$) and their correlation with the random oracle.

- *Oracle independent.* The circuit family $\{\mathcal{C}_n\}$ and H are uncorrelated. First the circuit family is chosen, then H is sampled.
- *Uniformly oracle dependent.* First H is chosen; then some fixed length string a (possibly correlated with H) is given as advice to the circuit family $\{\mathcal{C}_n\}$.
- *Non-uniformly oracle dependent.* First H is chosen; for each input length n, a potentially different string a_n is chosen which is given to circuit C_n as advice.

In the cryptographic setting, security against the third type of adversary is desired. We will prove our results against oracle independent adversaries and invoke known results to lift the security to non-uniform oracle dependent adversaries for cryptographic applications.

3.3 Basic Quantum information results

We setup some notation for distances and recall some basic results. Here, all density matrices are defined on the same Hilbert space.

Definition 26. Let ρ, ρ' be two mixed states. Then we define

- Fidelity: $\mathrm{F}(\rho, \rho') := \mathrm{tr}(\sqrt{\sqrt{\rho}\rho'\sqrt{\rho}})$
- Trace Distance: $\mathrm{TD}(\rho, \rho') := \frac{1}{2}\mathrm{tr}\,|\rho - \rho'|$ and
- Bures Distance: $\mathrm{B}(\rho, \rho') := \sqrt{2 - 2F(\rho, \rho')}$.

Fact 27. *For any set of strings S, any string s and any two mixed states, ρ and ρ', and any quantum algorithm $\mathcal{A}$ (which outputs a classical string), we have*

$$\left|\Pr[s \in S : s \leftarrow \mathcal{A}(\rho)] - \Pr[s \in S : s \leftarrow \mathcal{A}(\rho')]\right| \leq \mathrm{TD}(\rho, \rho') \leq B(\rho, \rho').$$

To see this, recall that $\mathrm{TD}(\rho, \rho') = \max_{\mathbb{I} \geq P \geq 0} |P(\rho - \rho')|$ and therefore $|\Pr[s \in S : s \leftarrow \mathcal{A}(\rho)] - \Pr[s \in S : s \leftarrow \mathcal{A}(\rho')]| \leq \mathrm{TD}(\rho, \rho')$. Recalling also $\mathrm{TD}(\rho, \rho') \leq \sqrt{1 - F(\rho, \rho')} \leq B(\rho, \rho')$ one obtains the asserted result.

We use the following basic properties repeatedly in our analysis. For any density matrices ρ, ρ', σ, it holds that: (1) $\mathrm{TD}(\rho, \rho') \leq \mathrm{TD}(\rho, \sigma) + \mathrm{TD}(\sigma, \rho')$, and (2) $\mathrm{TD}(\Phi(\rho), \Phi(\sigma)) \leq \mathrm{TD}(\rho, \sigma)$ for any completely positive trace non-increasing map Φ (see, e.g., [PGWPR06]).

Part I
Bounds on Quantum Depth

We first establish that $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}} \subsetneq \mathsf{BQP}$ relative to a random oracle. Based on this result, we describe how to construct a proof of d quantum depth which is insensitive to polynomial classical depth. Subsequently, we tighten the BQP upper bound to obtain a more fine-grained quantum depth separation, $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}} \subsetneq \mathsf{BPP}^{\mathsf{QNC}_{2d+\mathcal{O}(1)}^{\mathsf{BPP}}}$.

More precisely, in Section 4, we introduce a map which can be applied to any problem separating BQP and BPP which additionally specifies what we call *classical query soundness*, to create a new problem which separates $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$ and BQP. For concreteness, in Section 6 we apply this procedure to YZ's $\mathsf{CodeHashing}$ and in Section 7 prove that the resulting problem is not in $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$. We then, in Section 8, formalise the notion of a *proof of quantum depth* and construct a two-message proof of quantum depth protocol based on the previous result. Finally, in Section 9, we apply the map to a different problem and improve the upper bound to obtain the previously mentioned fine-grained quantum depth separation. Since this new problem is not efficiently verifiable, we do not obtain the associated fine-grained proof of quantum depth.

4 d-Recursive$[\mathcal{P}]$

Consider any problem $\mathcal{P}$ defined relative to a random oracle. We describe a map, which acts on $\mathcal{P}$ and creates a new problem d-$\mathsf{Rec}[\mathcal{P}]$. If $\mathcal{P}$ can be solved quantumly but not classically (in the sense explained below), then d-$\mathsf{Rec}[\mathcal{P}]$ can still be solved quantumly but cannot be solved with less than d quantum depth, i.e. d-$\mathsf{Rec}[\mathcal{P}] \in \mathsf{BQP}$ but d-$\mathsf{Rec}[\mathcal{P}] \notin \mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$. In fact, if $\mathcal{P}$ can be solved in depth d' then one can tighten the upper bound on $\mathcal{P}$ from BQP to $\mathsf{BPP}^{\mathsf{QNC}_{d''}^{\mathsf{BPP}}}$ where d'' is a function of d' (which we describe later).

4.1 Definition of $\mathcal{P}$

Any problem $\mathcal{P}$ which has the following two properties can be lifted to a problem which is not in $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$. The first property, *classical query soundness*, requires that no *unbounded* algorithm can solve the problem by making only *polynomially* many *classical* queries to the oracles. The second property, *bounded oracle domain*, is also intuitively quite simple. It requires that the problem only depends on a bounded domain of the random oracle. We formalise this property by requiring that the problem does not change if the random oracle is replaced with an arbitrary function except that it behaves exactly like the random oracle on the bounded oracle domain. We include this property for technical reasons and it is possible that it is not really necessary. However, for the problems we consider, both are easily satisfied. Formally, we have the following.

Definition 28 (*Classical query soundness* and *bounded oracle domain*)**.** Denote by $H : \{0,1\}^* \to \{0,1\}$ a random oracle. Define a problem $\mathcal{P}$ by a tuple $(\mathcal{X}, R_H)$ where $\mathcal{X}$ is a procedure which on input 1^λ generates a problem instance of size $\text{poly}(\lambda)$ and $R_H = \{0,1\}^* \times \{0,1\}^*$ is a relation which depends on H.

- We say $\mathcal{P}$ is *classical query sound* if for any unbounded algorithm $\mathcal{A}^H$ which makes at most $\text{poly}(\lambda)$ classical queries to H, it holds that

$$\Pr_H\left[(x,y) \in R_H : \begin{array}{c} x \leftarrow \mathcal{X}(1^\lambda) \\ y \leftarrow \mathcal{A}^H(x) \end{array}\right] \leq \text{negl}(\lambda)$$

  for all sufficiently large λ.

- Let $R_H(x) := \{y : (x,y) \in R_H\}$. We say $\mathcal{P}$ has a *bounded oracle domain* if there is a set $\{0,1\}^{p(\lambda)}$ where $p(\lambda)$ is an integer valued polynomial such that the following holds for each λ,

$$R_H(x) = R_{H'}(x)$$

  for all $x \in \mathcal{X}(1^\lambda)$ and all choices of functions $H' : \{0,1\}^* \to \{0,1\}$ such that $H'(z) = H(z)$ for all $z \in \{0,1\}^{p(\lambda)}$.

When we define YZ's $\mathsf{CodeHashing}$ problem, it would be evident that it satisfies both properties.

4.2 Definition of d-Recursive$[\mathcal{P}]$

Let $\mathcal{P}$ be any problem satisfying the properties in Definition 28 and which is in $\mathsf{BPP}^{\mathsf{QNC}_{d'}^{\mathsf{BPP}}}$. We can now introduce d-$\mathsf{Rec}[\mathcal{P}]$, a general construction which, for any $0 \le d \le \text{poly}(\lambda)$, lifts $\mathcal{P}$, to a problem which is not in $\mathsf{BPP}^{\mathsf{QNC}_{d}^{\mathsf{BPP}}}$ but is in $\mathsf{BPP}^{\mathsf{QNC}_{\text{poly}(d,d')}^{\mathsf{BPP}}}$. More precisely, the polynomial would be $(2d+1)\cdot d'$ because for each oracle call in $\mathcal{P}$, d-$\mathsf{Rec}[\mathcal{P}]$ would need $2d+1$ oracle calls. To see why, we need to know how d-Rec is defined.

At a high level, instead of asking for $\mathcal{P}$ to be solved relative to the random oracle H, d-$\mathsf{Rec}[\mathcal{P}]$ requires $\mathcal{P}$ to be solved relative to the random oracle H composed with itself d times. Clearly, H cannot be composed with itself in general because the domain and co-domain may not match. Suppose $H : \Sigma \to \{0,1\}^n$. Then one natural choice to consider is $\tilde{H} := H_d \circ \cdots \circ H_0$ where $H_d : \Sigma \to \{0,1\}^n$, $H_i : \Sigma \to \Sigma$ for $i \in \{1 \ldots d\}$. This has some issues, for instance the number of collisions in $\tilde{H}$ on an average would be larger than those in H. It turns out that for our analysis, enlarging the domain (as a function of $|\Sigma|$) is the appropriate choice, as explained below.

Definition 29 (d-$\mathsf{Rec}[\mathcal{P}]$)**.** Let $\mathcal{P} = (\mathcal{X}, R)$ be a problem satisfying Definition 28. We define d-$\mathsf{Rec}[\mathcal{P}]$ as follows. On input 1^λ, proceed as follows:

- Sample an instance of $\mathcal{P}$ as $x \leftarrow \mathcal{X}(1^\lambda)$, and
- denote its bounded oracle domain by $\Sigma := \{0,1\}^{p(\lambda)}$.
- Define $\tilde{H} := H_d \circ \cdots \circ H_1 \circ H_0$ where $H_0 : \Sigma \to \Sigma^{d'}$, for $\ell \in \{1, \ldots d-1\}$, $H_\ell : \Sigma^{d'} \to \Sigma^{d'}$ and $H_d : \Sigma^{d'} \to \{0,1\}$ are independent random oracles with $d' = 2d+5$.

The (d-$\mathsf{Rec}[\mathcal{P}]$) problem then is, given x, and oracle access to $(H_0, \ldots H_d)$, find a y s.t. $(x,y) \in R_{\tilde{H}}$.

4.2.1 Upper Bound

It may be the case that the algorithm which solves $\mathcal{P}$ makes only, say 1, query to the random oracle while it still has depth d' which is some large constant. Clearly, in this case, one can obtain a bound tighter than $(2d+1)\cdot d'$ on the depth of the circuit which solves d-$\mathsf{Rec}[\mathcal{P}]$. Indeed, we later consider a problem ($\mathsf{CollisionHashing}$) which has this property and therefore we formally state this upper bound as follows.

Lemma 30. *Suppose $\mathcal{P}$ is solved by a family of circuits in $\mathsf{QNC}_{d'}$ with probability $1-\epsilon(\lambda)$ and by making at most $t(\lambda)$ parallel queries. Then there is a family of circuits in $\mathsf{QNC}_{d''}$ which solves d-$\mathsf{Rec}[\mathcal{P}]$ with probability $1-\epsilon(\lambda)$ where $d'' \le \min[d' + (2d+1)\cdot t, (2d+1)\cdot d']$. The analagous statement holds for QC and CQ as well.*

Proof sketch. Fix a λ, let $\mathcal{C}_1 \in \mathsf{QNC}_{d'}$ be the circuit that solves $\mathcal{P}$ and let $\mathcal{C}_2 \in \mathsf{QNC}_{d''}$be a circuit which we construct and assert that it solves d-$\mathsf{Rec}[\mathcal{P}]$, with the same $1-\epsilon(\lambda)$ probability.

To obtain $d'' \le d' + (2d+1)\cdot t$, suppose the circuits are identical, except that for each of the t set of parallel oracle calls that $\mathcal{C}_1$ makes, $\mathcal{C}_2$ makes $(2d+1)\cdot t$ set of parallel oracle calls. This allows $\mathcal{C}_2$ to compute $\tilde{H}$ and therefore proceed exactly like $\mathcal{C}_1$. A simple upper bound on the quantum depth d'' of $\mathcal{C}_2$ then is $d' + (2d+1)\cdot t$.

To obtain $d'' \le (2d+1)\cdot d'$, suppose that the oracle is parallel invoked (worst case) at each layer. Then, $\mathcal{C}_2$ is identical to $\mathcal{C}_1$, except that for each of the d' layers of $\mathcal{C}_1$, $\mathcal{C}_2$ gets $(2d+1)$ layers and can therefore evaluate $\tilde{H}$ exactly like $\mathcal{C}_1$. An upper bound on the depth of $\mathcal{C}_2$ is then $d'' \le (2d+1)\cdot d'$. $\square$

4.2.2 Lower bound

The main property of d-$\mathsf{Rec}[\cdot]$ is the following which we establish in Section 7.

Lemma 31 (d-$\mathsf{Rec}[\mathcal{P}] \notin \mathsf{BPP}^{\mathsf{QNC}_{d}^{\mathsf{BPP}}}$)**.** *Every CQC_d circuit succeeds at solving d-$\mathsf{Rec}[\mathcal{P}]$ (see Definition 29) with probability at most* $\text{negl}(\lambda)$ *on input 1^λ for $d \le \text{poly}(n)$.*

5 Preliminaries

Instead of working with d-$\mathsf{Rec}[\cdot]$ abstractly, we apply this map to the $\mathsf{CodeHashing}$ problem introduced by [YZ22]. To use and describe the seminal work of YZ, we need the following notions.

Problem	$\in$		$\notin$	Verification	Interpretation	Remarks
CodeHashing	BQP	$\nsubseteq$	BPP	Public	Proof of Quantumness	[YZ22]; Definition 33
d-CodeHashing	BQP	$\nsubseteq$	$\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$	Public	$(d, \text{poly}(n))$-Proof of Quantum Depth Refutes Jozsa's conjecture	See Definition 35; Equivalent to d-Rec[CodeHashing]

Table 4: In Part I we started with CodeHashing due to [YZ22] and created a new problem we termed d-CodeHashing which showed $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}} \subsetneq \mathsf{BQP}$, for any fixed $d \le \text{poly}(n)$. This refutes Jozsa's conjecture relative to the random oracle model. This problem also immediately serves as a Proof of d Quantum Depth.

Error Correcting Codes

Codes

A code of length $n \in \mathbb{N}$ over an alphabet Σ is a subset $C \subseteq \Sigma^n$.

Linear codes [YZ22]. Let $\mathbb{F}_q$ be a finite field of order q for some prime power q and $\Sigma = \mathbb{F}_q$. A linear code C of length $n \in \mathbb{N}$ over the alphabet Σ is defined as a subset $C \subseteq \mathbb{F}_q^n$, which is also a linear subspace of $\mathbb{F}_q^n$. Further, we define the rank of a linear code C as the dimension of the linear subspace $C \subseteq \mathbb{F}_q^n$.

Folded linear codes [Kra03; GR08; YZ22]. Let $\mathbb{F}_q$ be a finite field of order q for some prime power q and m be a positive integer. A code C is said to be an m-folded linear code of length n if its alphabet is $\Sigma = \mathbb{F}_q^m$ and $C \subseteq \Sigma^n$ is a linear subspace of $C \subseteq \Sigma^n$, where C is embedded into $\mathbb{F}_q^{mn}$ in the canonical way.

It is clear that 1-folded linear codes are just linear codes. In fact, for a positive integer m that divides n and a linear code $C \subseteq \mathbb{F}_q^n$, we can define its m-folded version $C^{(m)}$ as

$$C^{(m)} := \left\{ \left(\overbrace{(x_1, \cdots, x_m)}^{m}, \cdots, \overbrace{(x_{n-m+1}, \cdots, x_n)}^{m} \right) : (x_1, \cdots, x_n) \in C \right\}.$$

Conversely, any folded linear code can be written as $C^{(m)}$ for some linear code C and a positive integer m.

Dual codes [YZ22]. A dual code $C^\perp$ of a linear code C of length n and rank k over the alphabet $\mathbb{F}_q$ is defined as the orthogonal complement of C as a linear subspace over $\mathbb{F}_q^n$. That is,

$$C^\perp := \left\{ \boldsymbol{x} \in \mathbb{F}_q^n : \boldsymbol{x} \cdot \boldsymbol{x}' = 0 \ \forall \ \boldsymbol{x}' \in C \right\}.$$

$C^\perp$ is itself a linear code of length n and rank $n-k$ over $\mathbb{F}_q$. Similarly, for an m-folded linear code $C \in \mathbb{F}_q^{mn}$ of length n over the alphabet $\mathbb{F}_q^m$, its dual $C^\perp$ is defined as

$$C^\perp := \left\{ \boldsymbol{x} \in \mathbb{F}_q^{mn} : \boldsymbol{x} \cdot \boldsymbol{x}' = 0 \ \forall \ \boldsymbol{x}' \in C \right\}.$$

Note that for any linear code C of length n and an integer m that divides n, we have $(C^\perp)^{(m)} = \left(C^{(m)}\right)^\perp$.

List recovery [YZ22]. Let $C \subseteq \Sigma^n$ be a code and $\boldsymbol{x} := (x_1, \cdots, x_n) \in C$ be a codeword. For subsets $S_i \subseteq \Sigma$ such that $|S_i| \le l$ for $i \in [n]$, define the index set $I_{\boldsymbol{x}, \{S_i\}, l} := \{i \in [n] : x_i \in S_i\}$. Then, we say that $C \subseteq \Sigma^n$ is (ζ, l, L)-list recoverable if for any subsets $S_i \subseteq \Sigma$ such that $|S_i| \le l$ for $i \in [n]$, we have [26]

$$\left| \left\{ \boldsymbol{x} \in C : \left| I_{\boldsymbol{x}, \{S_i\}, l} \right| \ge (1 - \zeta) n \right\} \right| \le L.$$

Suitable Codes

YZ use folded codes which satisfy certain properties. They call these codes *suitable codes*. They show that folded Reed-Solomon codes with appropriate parameters are suitable. We would not need these details for our result—the following suffices.

[26] List recovery usually requires efficient computation of all codewords $(x_1, \cdots, x_n) \in C$ that satisfy $\left| I_{\boldsymbol{x}, \{S_i\}, l} \right| \ge (1 - \zeta) n$ using $\{S_i\}$, however, it is not relevant for our purposes, so we don't demand it here.

Lemma 32 (Suitable Codes [YZ22])**.** *For any constants $0 < c < c' < 1$, there exists an explicit family $\{C_\lambda\}_{\lambda \in \mathbb{N}}$ of folded linear codes over the alphabet $\Sigma = \mathbb{F}_q^m$ of length n where $|\Sigma| = 2^{\lambda^{\Theta(1)}}$, $n = \Theta(\lambda)$ and $|C_\lambda| \geq 2^{n+\lambda}$ that satisfies the following.*[27]

1. *C_λ is (ζ, ℓ, L)-list recoverable where $\zeta = \Omega(1), l = 2^{\lambda^c}$ and $L = 2^{\tilde{O}(\lambda^{c'})}$.*
2. *There is an efficient deterministic decoding algorithm $\mathsf{Decode}_{C^\perp}$ for $C^\perp$ that satisfies the following. Let $\mathcal{D}$ be a distribution over Σ that outputs 0 with probability $1/2$ and otherwise outputs an element of $\Sigma \backslash \{0\}$ uniformly at random. Then, it holds that*
$$\Pr_{\mathrm{e} \leftarrow \mathcal{D}^n}[\forall \mathrm{x} \in C^\perp, \mathsf{Decode}_{C^\perp}(\mathrm{x} + \mathrm{e}) = \mathrm{x}] = 1 - 2^{-\Omega(\lambda)}.$$
3. *For all $j \in [n-1]$, $\Pr_{\mathrm{x} \leftarrow C_\lambda}[\mathsf{hw}(\mathrm{x}) = n - j] \leq \left(\frac{n}{|\Sigma|}\right)^j$.*

6 The d-CodeHashing Problem

This section introduces the problem we use for proving our main result.

6.1 Background — CodeHashing Problem [YZ22]

To describe our problem, we first recall that YZ's proof of quantumness is based on the following problem.

Definition 33 (CodeHashing Problem; Paraphrased from [YZ22])**.** Let

- $\{C_\lambda\}_\lambda$ be a family of codes over an alphabet $\Sigma = \mathbb{F}_q^m$ that satisfies the requirements of Lemma 32 with arbitrary $1 < c < c' < 1$,
- $H : \Sigma \to \{0,1\}^n$ be a random oracle.

Given the code family, and access to H, on input 1^λ, the CodeHashing problem is to find an $\mathrm{x} = (\mathrm{x}_1, \mathrm{x}_2, \ldots \mathrm{x}_n) \in C_\lambda$ such that for all $i \in \{1 \ldots n\}$, the ith bit of $H(\mathrm{x}_i)$ equals 1.

Note that for suitable codes, $\lambda = \Theta(n)$. Also note that the oracle H as described in the problem can be implemented using the standard random oracle from $\{0,1\}^*$ to $\{0,1\}$, as discussed in Section 3.2. YZ showed that this problem is contained in NP and BQP but not in BPP.

Theorem 34 (Paraphrased from [YZ22])**.** *The following hold in the random oracle model (for oracle-independent circuits).*

CodeHashing $\in$ BQP*: A QPT machine can solve the code hashing problem with overwhelming probability, i.e.* $1 - \mathrm{negl}(\lambda)$.

CodeHashing $\notin$ BPP*: Every classical circuit which makes at most 2^{λ^c} queries to the oracle solves the code hashing problem with probability at most $2^{-\Omega(\lambda)}$.*

YZ not only show that CodeHashing $\notin$ BPP, they show that even an unbounded machine would not succeed at solving CodeHashing with noticeable probability if it makes at most polynomially many (classical) queries to the random oracle. It is this *classical query soundness* property that we use later in our proof. Also observe that a BPP machine can easily check if a solution to CodeHashing is valid.

6.2 The Problem — d-CodeHashing Problem

We call our problem d-CodeHashing which is basically[28] To be conceret, we explicitly state it. d-$\mathsf{Rec}[CH]$.

Definition 35 (d-CodeHashing Problem)**.** Let

- $\{C_\lambda\}_\lambda$ be a family of codes over an alphabet $\Sigma = \mathbb{F}_q^m$ that satisfies the requirements of Lemma 32 with arbitrary $1 < c < c' < 1$,

[27] YZ point out that item 3 is not needed for proof of quantumness. It is used for showing one-way-functions. We inherit these in our construction of proof of depth and one-way functions resp.

[28] The only difference is in the range of H but this is without loss of generality due to domain splitting.

- $\tilde{H} := H_d \circ \cdots \circ H_1 \circ H_0$ where $H_0 : \Sigma \to \Sigma^{d'}$, for $\ell \in \{1, \dots, d-1\}$, $H_\ell : \Sigma^{d'} \to \Sigma^{d'}$ and $H_d : \Sigma^{d'} \to \{0,1\}^n$ are independent random oracles with $d' := 2d + 5$,
- $\mathrm{Bit}_i[\tilde{H}]$ denote the ith bit of $\tilde{H}$,

Given the code family $\{C_\lambda\}_\lambda$, access to random oracles $H_0, \dots H_d$, on input 1^λ, the d-$\mathsf{CodeHashing}$ problem is to find an $\mathrm{x} = (\mathrm{x}_1, \mathrm{x}_2, \dots \mathrm{x}_n) \in C_\lambda$ such that for all i, the ith bit of $\tilde{H}(\mathrm{x}_i)$ is 1, i.e. $\mathrm{Bit}_i[\tilde{H}(\mathrm{x}_i)] = 1$.

7 Lower Bounds

In this section, we establish the following key property of the d-$\mathsf{CodeHashing}$ problem. The proof of Lemma 31 is also immediate from the proof of the following.

Lemma 36 (d-$\mathsf{CodeHashing} \notin \mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$)**.** *Every* CQC_d *circuit*[29] *(which subsumes* $\mathsf{QC_d}$ *and* $\mathsf{CQ_d}$ *circuits) with oracle access to* $H_0, \dots H_d$*, succeeds at solving* d-$\mathsf{CodeHashing}$ *with probability at most* $\mathrm{negl}(\lambda)$ *on input* 1^λ*.*

We prove Lemma 36 in three main steps. First, we establish $\mathsf{QNC_d}$ hardness. We use this as a warm-up for introducing notations and concepts (in particular "base sets") which we build on for establishing $\mathsf{QC_d}$ hardness. The basic tools we need are discussed next, in Subsection 7.2. $\mathsf{CQ_d}$ hardness requires more work (and more technical tools) and we defer that discussion to Subsection 7.5. We then combine the ideas used in these three main steps to establish CQC_d hardness. Before delving into the proof of Lemma 36, we look at one of its main consequences.

7.1 Consequence: Jozsa's conjecture/Aaronson's challenge

Jozsa had conjectured that $\mathsf{BPP}^{\mathsf{QNC}} = \mathsf{BQP}$. Lemma 36 and Theorem 34, however, immediately yield the following theorem. Note that the classes stated below are the corresponding search variants (see Section 3.1).

Theorem 37 ($\mathsf{BPP}^{\mathsf{QNC}^{\mathsf{BPP}}} \subsetneq \mathsf{BQP}$)**.** *The following hold (unconditionally) in the random oracle model, for* $d = \lambda$ *where* λ *is the input size.*

d-$\mathsf{CodeHashing} \in \mathsf{BQP}$*: A QPT machine can solve the code hashing problem with overwhelming probability, i.e.* $1 - \mathrm{negl}(\lambda)$*, by making* $\mathcal{O}(\lambda)$ *queries to the random oracle.*

d-$\mathsf{CodeHashing} \notin \mathsf{BPP}^{\mathsf{QNC}^{\mathsf{BPP}}}$*: Every* $\mathsf{CQC}_{\log(\lambda)}$ *circuit succeeds at solving* d-$\mathsf{CodeHashing}$ *with probability at most* $\mathrm{negl}(\lambda)$ *on input* 1^λ*.*

We emphasise that $\mathsf{QNC}^{\mathsf{BPP}} \cup \mathsf{BPP}^{\mathsf{QNC}} \subseteq \mathsf{BPP}^{\mathsf{QNC}^{\mathsf{BPP}}}$ and so Theorem 37 shows that even a more liberal interpretation of Jozsa's conjecture, in the random oracle model, is false. One might wonder if $\mathsf{BPP}^{\mathsf{QNC}^{\mathsf{BPP}}}$ is strictly larger than $\mathsf{QNC}^{\mathsf{BPP}} \cup \mathsf{BPP}^{\mathsf{QNC}}$. Indeed, this is the case and we show it in Part II.

7.2 Known Results

We first state a simplified version of the so called "one-way to hiding", or briefly, the O2H lemma (see Subsection 7.2.1) due originally to Ambainis, Hamburg and Unruh [AHU19]. Our presentation, however, is inspired by [CCL20] and [AGS22]. We then state a version tailored to our setup (see Subsection 7.2.2) and end with some elementary results (see Subsection 7.2.3).

7.2.1 The O2H lemma

Informally, the O2H lemma says the following: suppose there are two oracles $\mathcal{O}$ and $\mathcal{Q}$ which behave identically on all inputs except some subset S of their input domain. Let $\mathcal{A}^{\mathcal{O}}$ and $\mathcal{A}^{\mathcal{Q}}$ be identical quantum algorithms, except for their oracle access, which is to $\mathcal{O}$ and $\mathcal{Q}$ respectively. Then, the probability that the result of $\mathcal{A}^{\mathcal{O}}$ and $\mathcal{A}^{\mathcal{Q}}$ will be distinct, is bounded by the probability of finding the set S. We suppress the details of the general finding procedure and only focus on the case of interest for us here.

[29] We assume the circuits are "oracle independent" as described in Subsection 3.2.2.

We begin by setting up some notation for this section (adapted from [AGS22]). We use the symbol $\mathcal{L}$ for the oracle.[30] The workspace register is denoted by W which is left untouched by the oracle. The query register is denoted by Q and the response register by R. Suppose we make $m = \mathrm{poly}(n)$ parallel queries to $\mathcal{L}$. We use boldface to represent the associated quantities. In particular, the parallel queries $(q_1, q_2 \dots q_m)$ are denoted by the tuple $\boldsymbol{q}$, the query registers $(Q_1, Q_2 \dots Q_m)$ which would hold these queries are denoted by $\boldsymbol{Q}$ and the corresponding response registers $(R_1, R_2 \dots R_m)$ are denoted by $\boldsymbol{R}$.

Definition 38 ($U^{\mathcal{L}\backslash S}$)**.** Suppose U acts on $\boldsymbol{QR}W$, $\mathcal{L}$ is an oracle that acts on $\boldsymbol{QR}$ and S is a subset of the query domain of $\mathcal{L}$. We define

$$U^{\mathcal{L}\backslash S} |\psi\rangle_{\boldsymbol{QR}W} |0\rangle_B := \mathcal{L} U_S U |\psi\rangle_{\boldsymbol{QR}W} |0\rangle_B$$

where B is a qubit register, and U_S flips qubit B if any query is made inside the set S, i.e.

$$U_S |\boldsymbol{q}\rangle_{\boldsymbol{Q}} |b\rangle_B := \begin{cases} U_S |\boldsymbol{q}\rangle_{\boldsymbol{Q}} |b\rangle_B & \text{if } \boldsymbol{q} \cap S = \varnothing \\ U_S |\boldsymbol{q}\rangle_{\boldsymbol{Q}} |b \oplus 1\rangle_B & \text{otherwise.} \end{cases}$$

Here[31] we treat $\boldsymbol{q}$ as a set when we write $\boldsymbol{q} \cap S$.

For notational simplicity, in the following, we drop the boldface for the query and response registers as they do not play an active role in the discussion.

Definition 39 ($\Pr[\mathrm{find} : U^{\mathcal{L}\backslash S}, \rho]$; adapted from [AGS22])**.** Let $U^{\mathcal{L}\backslash S}$ be as above and suppose $\rho \in \mathrm{D}(QRWB)$. We define

$$\Pr[\mathrm{find} : U^{\mathcal{L}\backslash S}, \rho] := \mathrm{tr}[\mathbb{I}_{QRW} \otimes |1\rangle\langle 1|_B U^{\mathcal{L}\backslash S} \circ \rho].$$

This will depend on $\mathcal{L}$ and S. When $\mathcal{L}$ and S are random variables, we additionally take expectation over them.[32]

Remark 40 (adapted from [AGS22])*.* Let $U^{\mathcal{L}\backslash S}$ be as in Definition 38 and let $|\psi\rangle \in QRW$. Note that we can always write

$$\mathcal{L} U |\psi\rangle_{QRW} = |\phi_0\rangle_{QRW} + |\phi_1\rangle_{QRW}$$

where $|\phi_0\rangle$ and $|\phi_1\rangle$ contains queries outside S and inside S respectively, i.e. $\langle \phi_0 | \phi_1 \rangle = 0$. Further, we can write

$$U^{\mathcal{L}\backslash S} |\psi\rangle_{QRW} |0\rangle_B = |\phi_0\rangle_{QRW} |0\rangle_B + |\phi_1\rangle_{QRW} |1\rangle_B .$$

The following is a special case of the O2H lemma introduced in [AHU19].

Lemma 41 (O2H lemma; as stated in [AGS22])**.** *Let*

- *$\mathcal{L}$ be an oracle which acts on QR and S be a subset of the query domain of $\mathcal{L}$,*
- *$\mathcal{G}$ be a shadow of $\mathcal{L}$ with respect to S, i.e. $\mathcal{G}$ and $\mathcal{L}$ behave identically for all queries outside S,*
- *further, suppose that within S, $\mathcal{G}$ responds with $\perp$ while (again within S), $\mathcal{L}$ does not respond with $\perp$. Finally, let Π_t be a measurement in the computational basis, corresponding to the string t.*

Then

$$\begin{aligned} |\mathrm{tr}[\Pi_t \mathcal{L} \circ U \circ \rho] - \mathrm{tr}[\Pi_t \mathcal{G} \circ U \circ \rho]| &\le B(\mathcal{L} \circ U \circ \rho, \mathcal{G} \circ U \circ \rho) \\ &\le \sqrt{2 \Pr[\mathit{find}\ : U^{\mathcal{L}\backslash S}, \rho]}. \end{aligned}$$

If $\mathcal{L}$ and S are random variables with a joint distribution, we take the expectation over them in the RHS (see Definition 39).

The right hand side in Lemma 41 may be bounded using Lemma 42 below. Lemma 42 applies when the locations queried are independent of the set being hidden.

[30] Instead of $\mathcal{Q}$ as above to avoid confusion.

[31] i.e. the condition $\boldsymbol{q} \cap S = \varnothing$ reads there is no $\boldsymbol{i}$ for which $q_i \notin S$.

[32] i.e. $\Pr[\mathrm{find} : U^{\mathcal{L}\backslash S}, \rho] := \mathbb{E}_{\mathcal{L},S} \mathrm{tr}[\mathbb{I}_{QRW} \otimes |1\rangle\langle 1|_B U^{\mathcal{L}\backslash S} \circ \rho]$.

Lemma 42 ([CCL20; AHU19] Bounding $\Pr[\text{find} : U^{\mathcal{L}\backslash S}, \rho]$)**.** *Suppose S is a random variable and $\Pr[x \in S] \le p$ for some p. Further, assume that U and ρ are uncorrelated*[33] *to S. Then, (see Definition 38)*

$$\Pr[\text{find} : U^{\mathcal{L}\backslash S}, \rho] \le \bar{q} \cdot p$$

where $\bar{q}$ is the total number of queries U makes to $\mathcal{L}$.

For completeness, we include the proofs in Section A of the Appendix.

7.2.2 O2H adapted to our analysis

Recall that d-CodeHashing (see Definition 35) is defined using $d+1$ oracles, $\{H_i\}_{0,1,\dots d}$. Therefore, instead of considering a set S where the oracles ($\mathcal{L}$ and $\mathcal{G}$) behave differently, we consider a sequence of sets. Let S^{out} denote a sequence of d sets and similarly let S^{in} denote a sequence of d sets contained in S^{out} (element-wise). Why we take d and not $d+1$ should become evident later—briefly, it is because the domain of H_0 is known by construction but the domain of H_1 which is of interest, i.e. $H_0(\Sigma)$, is what we are trying to hide (and similarly for $H_2, \dots H_d$). Observe that in Lemma 41, the state ρ was uncorrelated to the set S. However, in our application, the quantum state can potentially contain information about $\mathcal{L}$ restricted to values outside S^{out}. However, within S^{out}, the values of S^{in} stay uncorrelated and we can apply Lemma 41. The following notation allows us to state this formally.

Notation 43. Consider the following (see Figure 5).

- Let $\mathcal{L}' := (H'_0, H'_1, \dots H'_d)$ where the domain and range of H'_i is the same as that of H_i (as defined in Definition 35).
  - These functions themselves may be sampled from an arbitrary distribution (unlike H_i).
- Let $S^{\text{out}} := (S^{\text{out}}_1, \dots S^{\text{out}}_d)$ and $S^{\text{in}} := (S^{\text{in}}_1, \dots S^{\text{in}}_d)$ be a sequence of (random) subsets such that $S^{\text{in}}_i \subseteq S^{\text{out}}_i \subseteq \text{dom}(H'_i)$.
  - Note that S^{out} and S^{in} are random variables which may be arbitrarily correlated with $\mathcal{L}'$.
- Let $\check{\mathcal{L}}'$ refer to $\mathcal{L}'$ outside of S^{out}, i.e. $(\check{H}'_0, \dots \check{H}'_d)$ where $\check{H}'_i : \text{dom}(H'_i)\backslash S^{\text{out}} \to H'_i(\text{dom}(H'_i)\backslash S^{\text{out}})$ and $\check{H}'_i(x) := H'_i(x)$ for all $x \in \text{dom}(\check{H}'_i)$.
- Let $\hat{\mathcal{L}}'$ refer to $\mathcal{L}'$ inside S^{out}, i.e. $(\hat{H}'_0, \dots \hat{H}'_d)$ where $\hat{H}'_i : S^{\text{out}}_i \to H'_i(S^{\text{out}}_i)$.

We used $\mathcal{L}'$ instead of $\mathcal{L}$ because, in our proofs, $\mathcal{L}$ will be conditioned on various random variables and it is this conditioned $\mathcal{L}$ we work with.

Corollary 44. *Let $\mathcal{L}', S^{\text{out}}, S^{\text{in}}, \hat{\mathcal{L}}', \check{\mathcal{L}}'$ be as in Notation 43 above. Suppose a quantum state ρ and a unitary U are drawn from a distribution which may be correlated with $\mathcal{L}'$. Suppose, $\sigma := \rho|\check{\mathcal{L}}'$ and $V := U|\check{\mathcal{L}}'$, are uncorrelated to $R := S^{\text{in}}|\check{\mathcal{L}}'$. Let $\mathcal{N} := \mathcal{L}'|\check{\mathcal{L}}'$. Given that $\Pr[x \in S^{\text{in}}_i|\check{\mathcal{L}}'] \le p$ for all $x \in \text{dom}(H'_i)$ and $i \in \{1 \dots d\}$, it holds that*

$$\Pr[\text{find} : V^{\mathcal{N}\backslash R}, \sigma] \le d \cdot \bar{q} \cdot p$$

where $\bar{q}$ is the total number of queries V makes to the oracles $\mathcal{L}'$.

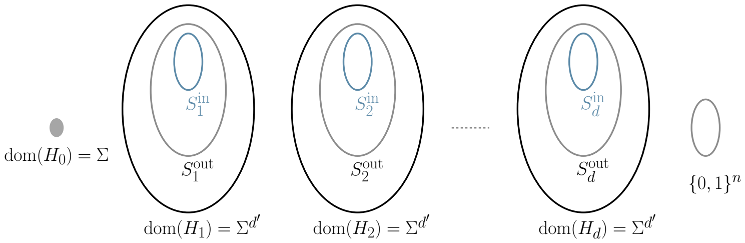


Figure 5

[33] i.e. the distribution from which S is sampled is uncorrelated to the distribution from which U and ρ are sampled,

Proof sketch. We assume that the ρ contains information about $\check{\mathcal{L}}'$ and therefore contains information about S^{out}. At best, U can query $\mathcal{L}$ at x such that $x \in S_i^{\text{out}}$ for some i. However, given $\hat{\mathcal{L}}'$ (and therefore S^{out}), $\Pr[x \in S_i^{\text{in}} | \hat{\mathcal{L}}]$ is bounded by p so, by argument used in the proof for Lemma 42, together with a union bound, one obtains the asserted bound. □

When we apply the O2H lemma via the corrollary above, it would be helpful to consider shadows for a sequence of oracles—the analogue of $\mathcal{G}$ in Lemma 41. Defining it formally helps the presentation.

Definition 45 (Shadow oracle wrt $\bar{S}'$)**.** Let $\mathcal{L}' := (H'_0, \dots H'_d)$ and Σ be as in Notation 43. Let $\bar{S}' = (S'_1, S'_2 \dots S'_d)$ be a tuple of d sets where each set $S'_i \subseteq \Sigma^{d'}$ for all $i \in \{1, \dots d\}$. The shadow oracle $\mathcal{M}'$ of $\mathcal{L}'$ wrt $\bar{S}'$ is defined as $\mathcal{M}' := (M'_0, \dots M'_d)$ where

$$M'_i(\mathrm{l}) := \begin{cases} H'_i(\mathrm{l}) & \mathrm{l} \in \Sigma \backslash S'_i \\ \perp & \mathrm{l} \in S'_i. \end{cases}$$

7.2.3 Elementary results

The following elementary observations will be useful in computing probabilities which arise in our analysis. We use the following convention: ${}_aP_b := a!/(a-b)!$ and ${}_aC_b := a!/(b! \cdot (a-b)!)$ for $a \geq b$.

Fact 46. *One has*

$$\frac{{}_aP_b}{{}_{a+1}P_{b+1}} = \frac{1}{a+1} \quad \text{and} \quad \frac{{}_aC_b}{{}_{a+1}C_{b+1}} = \frac{b+1}{a+1}.$$

Remark 47. Let $M \geq N$ be an integer and fix some element $x \in \{1, 2 \dots M\}$. Suppose t is a *tuple* of size N, sampled uniformly from the collection of all size N *tuples* containing distinct elements from $\{1, 2 \dots M\}$. Then

$$\Pr(x \in t) = \frac{{}_{M-1}P_{N-1} \cdot N}{{}_MP_N} = \frac{N}{M}.$$

Similarly, suppose X is a *set* of size N, sampled uniformly from the collection of all size N *subsets* of $\{1 \dots M\}$. Then, again,

$$\Pr(x \in X) = \frac{{}_{M-1}C_{N-1}}{{}_MC_N} = \frac{N}{M}.$$

The following elementary calculation was alluded to in the discussion following Definition 35. It allows us to reduce our problem to permutations, without loss of generality.

Claim 48. Let $f : A \to A$ be a random function, i.e. for all $a \in A$**,** $f(a)$ is mapped to $a' \in A$ with probability $1/|A|$. Let $B \subsetneq A$ be an arbitrary set. Then the probability that $|f(B)| = |B|$ is at least $1 - |B|^2/|A|$. Equivalently, the probability that $|f(B)| < |B|$ is at most $|B|^2/|A|$.

Proof. It suffices to show that f is injective on B with the same probability. We have

$$\begin{aligned} \Pr[|f(B)| = |B|] &= \Pr(f \text{ has no collisions in } B) \\ &= 1 - \Pr(f \text{ has at least one collision in } B) \\ &\geq 1 - \epsilon \end{aligned}$$

if $\Pr(f$ has at least one collision in $B) \leq \epsilon$. Since f is random, the probability that a given b collides with some b' is simply the probability that $f(b')$ is assigned the value $f(b)$ by f which is at most $|B|/|A|$, i.e. $\Pr(\exists \ \ b' \neq b \ \text{ s.t. } \ f(b) = f(b')) \leq |B|/|A|$. Therefore,

$$\begin{aligned} \Pr(f \text{ has at least one collision in } B) &= \Pr(\vee_{b \in B} \quad b \text{ collides under } f) \\ &\leq \sum_{b \in B} \Pr(b \text{ collides under } f) \\ &= \sum_{b \in B} \Pr(\exists \ \ b' \neq b \ \text{ s.t. } \ f(b) = f(b')) \\ &\leq |B| \cdot |B|/|A|. \end{aligned}$$

□

7.3 Warm-up — QNC_d exclusion

We have now stated all the preliminaries we need to show our first lower bound. We do this in three stages. First, we define two algorithms which help us reduce to the case of permutations and allow us to perform "domain hiding" for each set of parallel calls. The latter is essentially the same as the "russian nesting doll" technique, as applied by [CCL20], adapted to the random oracle setup. In the second stage, we prove that the first algorithm does indeed produce permutations with high probability and that the second algorithm satisfies the properties needed to apply Corollary 44. In the third (final) stage, we combine these into a proof of $\mathsf{QNC_d}$ hardness of d-$\mathsf{CodeHashing}$. The primary purpose here is to setup the basic notation which is used to show $\mathsf{QC_d}$ and later $\mathsf{CQ_d}$ hardness.

7.3.1 Shadow oracles for QNC_d hardness

We begin with constructing "base sets" (see Figure 6a). We simply generate a random set $S_{01} \subseteq \mathsf{dom}(H_1)$ and propagate it through $\mathcal{L}$. Ensuring this set is sufficiently small compared to $\Sigma^{d'}$, one can later show that $\mathcal{L}$ restricted to the sequence of sets $(S_{01}, H_1(S_{01}), H_2(H_1(S_{01})), \dots H_d(\dots H_1(S_{01})\dots))$ behaves as a permutation with high probability.

Algorithm 49 (Base sets)**.** *Let $\mathcal{L} := (H_0, \dots H_d)$, d' and Σ be as in Definition 35. Let $S_i := H_{i-1}(\dots H_0(\Sigma)\dots) \subseteq \Sigma^{d'}$ for $i \in \{1, \dots d\}$.*

1. *Base Sets*
   (a) *Sample $S_{01} \subseteq \Sigma^{d'}$ uniformly at random,s.t. $S_1 \subseteq S_{01}$ and $|S_{01}|^2/|\Sigma^{d'}| = 1/|\Sigma|$ (i.e. $|S_{01}| = |\Sigma^{d+2}|$).*
   (b) *Define $S_{0,i+1} := H_i(S_{0,i})$ for $i \in \{1, \dots d-1\}$.*
2. *Abort if any of the following conditions are not met.*
   (a) *$|S_{0i}| = |\Sigma|^{d+2}$ for all $i \in \{1, \dots d\}$ (the $i = 1$ condition holds by construction).*
   (b) *$|S_1| = |\Sigma|$ (which together with (a) implies $|S_i| = |\Sigma|$ for all $i \in \{1, \dots d\}$).*

Conditions in item 2 are important because the random function may introduce collisions. The conditions ensure there are no collisions in the domains of interest.

We now introduce the construction of the sets S_{ij} (see Figure 6b). These are perhaps best viewed as a matrix whose elements are subsets of $\Sigma^{d'}$,

$$S_{ij} \doteq \begin{bmatrix} S_{11} & H_1(S_{11}) & H_2(H_1(S_{11})) & \dots & H_d(\dots H_1(S_{11})\dots) \\ \varnothing & S_{22} & H_2(S_{22}) & \dots & H_d(\dots H_2(S_{22})\dots) \\ \varnothing & \varnothing & S_{33} & \dots & H_d(\dots H_3(S_{33})\dots) \\ & & & \ddots & \\ \varnothing & \varnothing & \varnothing & & S_{dd} \end{bmatrix}.$$

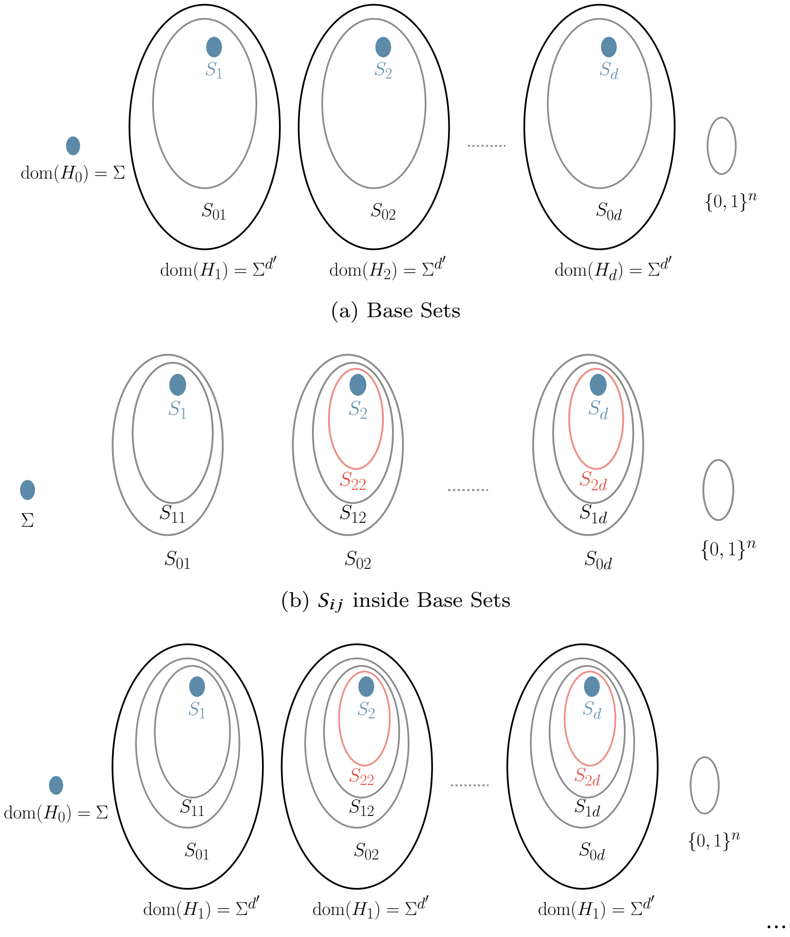
The first row is, element-wise, a subset of $(S_{01}, S_{02}, \dots S_{0d})$. Similarly, each row is an element-wise subset of the previous row. With each row, the size of the set drops exponentially (in n, relative to the previous row). The diagonal sets are chosen uniformly at random, ensuring S_i are contained within (just as we required for the "base sets"). Formally, the procedure is defined as follows.

Algorithm 50 (Procedure for constructing S_{ij})**.** *Let $\mathcal{L} := (H_0, \dots H_d)$, Σ and S_i be as in Algorithm 49. Suppose Algorithm 49 was executed. If Algorithm 49 aborts, define $S_{ij} = \varnothing$ for all $i, j \in \{1, \dots d\}$. If Algorithm 49 does not abort then, for each $i \in \{1, \dots d\}$*

1. *Define $S_{ik} = \varnothing$ for $1 \le k < i$.*
2. *Sample, uniformly at random, $S_{ii} \subseteq S_{i-1,i}$ such that $S_i \subseteq S_{ii}$ and $|S_{ii}|/|S_{i-1,i}| = 1/|\Sigma|$.*
3. *Define $S_{ik} = H_{k-1}(\dots H_i(S_{ii})\dots)$ for $i < k \le d$.*

In both cases, return $\bar{S}_i := (S_{i1}, S_{i2}, \dots S_{id})$ for each $i \in \{1, \dots d\}$.

Two short remarks—first, when Algorithm 49 fails, we simply abort and output $\varnothing$ as we don't care what happens in that case. This is because it fails with vanishing probability as we prove next. Second, it may help to note that $\bar{S}_i$, in the matrix representation above, is just the ith row of S_{ij}.


S_1
S_2
S_d
$\mathrm{dom}(H_0) = \Sigma$
S_{01}
S_{02}
S_{0d}
$\{0,1\}^n$
$\mathrm{dom}(H_1) = \Sigma^{d'}$
$\mathrm{dom}(H_2) = \Sigma^{d'}$
$\mathrm{dom}(H_d) = \Sigma^{d'}$
S_1
S_2
S_d
S_{22}
S_{2d}
Σ
S_{11}
S_{12}
S_{1d}
S_{01}
S_{02}
S_{0d}
$\{0,1\}^n$
S_1
S_2
S_d
S_{22}
S_{2d}
$\mathrm{dom}(H_0) = \Sigma$
S_{11}
S_{12}
S_{1d}
S_{01}
S_{02}
S_{0d}
$\{0,1\}^n$
$\mathrm{dom}(H_1) = \Sigma^{d'}$
$\mathrm{dom}(H_1) = \Sigma^{d'}$
$\mathrm{dom}(H_1) = \Sigma^{d'}$
...

(a) Base Sets

(b) S_{ij} inside Base Sets

(c) Unified view: Base Sets and S_{ij}

Figure 6: Illustrating the sets produced by Algorithm 49 and Algorithm 50

7.3.2 Properties of the shadow oracles

Like we said, Algorithm 49 fails with vanishing probability.

Claim 51. Algorithm 49 outputs abort with at most $(d+1)\cdot\mathrm{negl}(\lambda)$ probability where λ is as in Definition 35.

Proof. We use Claim 48 and a union bound. For each $i \in [d]$, condition 2 (a) fails with probability at most $1/|\Sigma|$. To see this, in Claim 48, set $f \leftarrow H_1$, $A \leftarrow \Sigma^{d'}$, $B \leftarrow S_{01}$ to conclude that the probability that $S_{02} = H_1(S_{01})$ has size strictly less than $|S_{01}|$ is at most $|B|^2/|A| = 1/|\Sigma|$. Proceeding similarly, set $f \leftarrow H_i$, $A \leftarrow \Sigma^{d'}$, $B \leftarrow S_{0i}$ to conclude that the probability that $S_{0,i+1} = H_i(S_{0i})$ has size strictly less than $|S_{0i}|$, is at most $1/|\Sigma|$. By a union bound, condition 2 (a) fails with probability at most $d \cdot 1/|\Sigma|$.

Similarly, condition 2 (b) fails with probability at most $1/|\Sigma|^{d'-2} < 1/|\Sigma|$ by Claim 48 with $f \leftarrow H_0$, $A \leftarrow \Sigma^{d'}$ and $B \leftarrow \Sigma$ (note that the claim is true even when $f : B \to A$). Therefore the probability of abort is at most $(d+1)\cdot 1/|\Sigma|$ where $|\Sigma| = 2^{\Theta(\lambda)}$, yielding the asserted bound. □

To apply Corollary 44, we would need a bound on $\Pr[x \in S_{ik}|S_{i,k-1}]$ conditioned on not aborting.

Claim 52. Let $\mathcal{L}$ be as in Definition 35, run Algorithm 49 and let E be the event that it does not abort. Obtain S_{ij} by running Algorithm 50. Then, it holds that

$$\Pr[x \in S_{ik}|(S_{i-1,k}, E)] \leq 1/\,|\Sigma|$$

and

$$\Pr[x \in S_{ik}|(\check{\mathcal{L}}, E)] \leq 1/|\Sigma|$$

where $\check{\mathcal{L}}$ is $\mathcal{L}$ outside $(S_{i-1,1}, \dots S_{i-1,d})$ (see Notation 43 with $S^{\mathrm{out}} \leftarrow (S_{i-1,j})_j$ and $\mathcal{L}' \leftarrow \mathcal{L}$) for all $1 \leq i \leq k \leq d$ where the probability is over $\mathcal{L}$, the randomness in Algorithm 49 and in Algorithm 50.

Proof sketch. Consider the $k = i$ case. Once $S_{i-1,i}$ is fixed, S_{ii} is a (uniform) random subset of $S_{i-1,i}$ and therefore the probability that any $x \in S_{i,i}$ (assume $x \in S_{i-1,i}$ to get an upper bound), is at most $|S_{i,i}|\,/\,|S_{i-1,i}| = 1/|\Sigma|$ (see Remark 47, first observation). The result continues to hold if $\mathcal{L}$ (or in particular $\check{\mathcal{L}}$) is specified because S_{ii} is sampled uniformly at random by Algorithm 50. For $k > i$, note that conditioned on E, each $H_1, H_2 \dots H_{d-1}$ behaves as a random permutation on $S_{0,1}, S_{0,2} \dots S_{0,d-1}$. In particular, conditioned on E, each $H_1, \dots H_{d-1}$ behaves as a random permutation on $S_{i-1,1}, S_{i-1,2} \dots S_{i-1,d-1}$ (even if $\check{\mathcal{L}}$ is given since it does not determine the values within $(S_{i-1,j})_j$). From the first observation in Remark 47, it follows that $x \in S_{ik}$ conditioned on E and $S_{i-1,k}$ for $k > i$, is also at most $1/|\Sigma|$. This is because H_{i-1} maps every subset of $S_{i-1,k-1}$ of size $|S_{i-1,k-1}|/|\Sigma|$ to another set of the same size in $S_{i-1,k}$ (i.e. H_{i-1} essentially behaves as a permutation) and Remark 47 shows the probability that $x \in S_{i,k}|\,(S_{i-1,k-1}, E)$ and $x \in S_{i,k}|(\check{\mathcal{L}}, E)$, are both bounded by $1/|\Sigma|$. □

7.3.3 d-CodeHashing is hard for QNC$_d$

With all the intermediate results proven, we can stitch them together to establish the QNC$_\mathrm{d}$ hardness of d-CodeHashing.

Lemma 53 (d-CodeHashing $\notin$ QNC$_d$)**.** *Every* QNC$_d$ *circuit succeeds at solving* d-CodeHashing *(see Definition 35) with probability at most* $\mathrm{negl}(\lambda)$ *on input* 1^λ *for* $d \leq \mathrm{poly}(n)$.

Proof. For clarity of presentation, we omit the input 1^λ when convenient. Let $\mathcal{L} := (H_0, \dots H_d)$ and Σ be as in Definition 35. Denote an arbitrary QNC$_d$ circuit, $\mathcal{A}^{\mathcal{L}}$ by

$$\mathcal{A}^{\mathcal{L}} := \Pi_{\mathrm{x}} \circ U_{d+1} \circ \mathcal{L} \circ U_d \dots \mathcal{L} \circ U_2 \circ \mathcal{L} \circ U_1$$

where Π_{x} is a projector corresponding to output x. Let Π_{valid} be a projector on the set $X_{\mathrm{valid}} = \{\mathrm{x}\}$ of all correct solutions to Definition 35. Π_{valid} implicitly depends on H and λ. We use Π_{valid} later. For now, run Algorithm 49 and let E be the event that it does not abort. Note that[34]

$$\left|\sum_{\mathrm{x}\in X_{\mathrm{valid}}} \Pr[\mathrm{x} \leftarrow \mathcal{A}^{\mathcal{L}}] - \sum_{\mathrm{x}\in X_{\mathrm{valid}}} \Pr[\mathrm{x} \leftarrow \mathcal{A}^{\mathcal{L}}|E]\right| \leq \mathrm{negl}(n). \tag{1}$$

[34] Using $\Pr[A] = \Pr[A|E]\Pr[E] + \Pr[A|\neg E]\Pr[\neg E]$, which yields $\Pr[A] - \Pr[A|E] \leq \Pr[A|\neg E]\Pr[\neg E]$, and that $\Pr[\neg E] = \mathrm{negl}(\lambda)$.

Let $(\bar{S}_i)_{i\in\{1,\dots d\}}$ be the output of Algorithm 50. Define

$$\mathcal{A}^{\mathcal{M}} := \Pi_{\mathrm{x}} \circ U_{d+1} \circ \mathcal{M}_d \circ U_d \dots \mathcal{M}_2 \circ U_2 \circ \mathcal{M}_1 \circ U_1$$

where $\mathcal{M}_i$ is the shadow oracle of $\mathcal{L}$ wrt $\bar{S}_i$ (see Definition 45).

$\mathcal{A}^{\mathcal{M}}|E$ cannot succeed with non-negligible probability: In this paragraph, we condition on E implicitly. Recall $\tilde{H} = H_d \circ \dots \circ H_0 : \Sigma \to \{0,1\}^n$ and $\tilde{H}_i$ is the ith bit of $\tilde{H}$ (see Definition 35). Observe that if $\mathrm{x} = (\mathrm{x}_1, \dots \mathrm{x}_d) \in C_\lambda$ is such that $\tilde{H}_i(\mathrm{x}_i) = 1$ for all i, then $\Pr[\mathrm{x} \leftarrow \mathcal{A}^{\mathcal{M}}] \le 1/2^n$. This is because the oracles $\mathcal{M}_1, \dots \mathcal{M}_d$ contain no information about $\tilde{H}_i(\mathrm{x}_i)$ therefore x cannot be correlated to the values the random oracle assigns to $\tilde{H}$. The probability that for any given x, all $\tilde{H}_i(\mathrm{x}_i)$ output 1 is at most $1/2^n$.

$\mathcal{A}^{\mathcal{M}}|E$ and $\mathcal{A}^{\mathcal{L}}|E$ have practically the same behaviour: Using a hybrid argument and the O2H lemma (see Lemma 41), one finds that the output distributions of $\mathcal{A}^{\mathcal{M}}|E$ and $\mathcal{A}^{\mathcal{L}}|E$ cannot be noticeably different. We have (we dropped the $\circ$ symbol, the conditioning on E and the subscript valid from Π_{valid} for brevity/clarity)

$$\begin{aligned}
&\left|\sum_{\mathrm{x}\in X_{\mathrm{valid}}} \Pr[\mathrm{x} \leftarrow \mathcal{A}^{\mathcal{L}}] - \sum_{\mathrm{x}\in X_{\mathrm{valid}}} \Pr[\mathrm{x} \leftarrow \mathcal{A}^{\mathcal{M}}]\right| \\
&= \left|\mathrm{tr}[\Pi_{\mathrm{valid}} U_{d+1} \mathcal{L} U_d \dots \mathcal{L} U_2 \mathcal{L} U_1 \rho_0 - \Pi_{\mathrm{valid}} U_{d+1} \mathcal{M}_d U_d \dots \mathcal{M}_2 U_2 \mathcal{M}_1 U_1 \rho_0]\right| && \text{monotonicity of TD} \\
&\le \left|\mathrm{tr}[\Pi U_{d+1} \mathcal{L} U_d \dots \mathcal{L} U_2 \underbrace{\mathcal{L} U_1 \rho_0}_{} - \Pi U_{d+1} \mathcal{L} U_d \dots \mathcal{L} U_2 \underbrace{\mathcal{M}_1 U_1 \rho_0}_{}]\right| + && \text{triangle inequality} \\
&\left|\mathrm{tr}[\Pi U_{d+1} \mathcal{L} U_d \dots U_3 \underbrace{\mathcal{L} U_2 \mathcal{M}_1 U_1 \rho_0}_{} - \Pi U_{d+1} \mathcal{L} U_d \dots \mathcal{L} U_3 \underbrace{\mathcal{M}_2 U_2 \mathcal{M}_1 U_1 \rho_0}_{}]\right| + \\
&\vdots \\
&\left|\mathrm{tr}[\Pi U_{d+1} \underbrace{\mathcal{L} U_d \mathcal{M}_{d-1} U_{d-1} \dots U_3 \mathcal{M}_2 U_2 \mathcal{M}_1 U_1 \rho_0}_{} - \Pi U_{d+1} \underbrace{\mathcal{M}_d U_d \mathcal{M}_{d-1} \dots U_3 \mathcal{M}_2 U_2 \mathcal{M}_1 U_1 \rho_0}_{}]\right| \\
&\le \mathrm{B}(\mathcal{L} \circ U_1(\rho_0), \mathcal{M}_1 \circ U_1(\rho_0)) + && \text{relation b/w TD and B} \\
&\mathrm{B}(\mathcal{L} \circ U_2(\rho_1), \mathcal{M}_2 \circ U_2(\rho_1)) + \\
&\vdots \\
&\mathrm{B}(\mathcal{L} \circ U_d(\rho_{d-1}), \mathcal{M}_d \circ U_d(\rho_{d-1})) \\
&\le \sum_{i=1}^{d} \sqrt{2 \Pr[\mathrm{find} : U_i^{\mathcal{L}\setminus\bar{S}_i}, \rho_{i-1}]} && \text{Lemma 41}
\end{aligned} \tag{2}$$

where $\rho_0 = |1^\lambda, 0\dots0\rangle\langle 1^\lambda, 0\dots0|$ and $\rho_i = \mathcal{M}_i \circ U_i \circ \dots \mathcal{M}_1 \circ U_1(\rho_0)$ for $i > 0$. To bound the last expression, one can use Lemma 42 via Corollary 44 (recall that everything is conditioned on E). Let $\check{\mathcal{L}}_i$ be $\mathcal{L}$ outside $(S_{i1}, \dots S_{id})$ (see Notation 43 with $\mathcal{L}' \leftarrow \mathcal{L}$, $S^{\mathrm{out}} \leftarrow (S_{ij})_j$ and define $\check{\mathcal{L}}_i := \check{\mathcal{L}}'$) for each $i \in \{0, 1 \dots d\}$ (we include 0 to include the base sets specified by Algorithm 49). Similarly, let $\hat{\mathcal{L}}_i$ be $\mathcal{L}$ inside $(S_{i1}, \dots S_{id})$ (see Notation 43 with $\mathcal{L}' \leftarrow \mathcal{L}$, $S^{\mathrm{out}} \leftarrow (S_{ij})_j$ and define $\hat{\mathcal{L}}_i := \hat{\mathcal{L}}'$). Note that the only information about $\mathcal{L}$ contained in $\mathcal{M}_i$, is $\check{\mathcal{L}}_i$, for each $i \in \{1, \dots d\}$. Consider $\Pr[\mathrm{find} : U_i^{\mathcal{L}\setminus\bar{S}_i}, \rho_{i-1}]$ and note that ρ_{i-1} at most specifies $\check{\mathcal{L}}_{i-1}$ (about $\mathcal{L}$). Let $\sigma_i := \rho_{i-1}|\check{\mathcal{L}}_{i-1}$, $\bar{R}_i := \bar{S}_i|\check{\mathcal{L}}_{i-1}$ and $\mathcal{N}_i := \mathcal{L}|\check{\mathcal{L}}_{i-1}$. Observe that $\bar{R}_i$ is uncorrelated to σ_i (because once $\check{\mathcal{L}}_{i-1}$ is fixed, σ_i contains no information about how $\mathcal{L}$ behaves in $\bar{S}_{i-1} = (S_{i-1,1} \dots S_{i-1,d})$ and $\bar{R}_i$ depends only on the randomness in Algorithm 50 and on $\hat{\mathcal{L}}_{i-1}$). One can thus apply Corollary 44 with Claim 52 to obtain

$$\Pr[\mathrm{find} : V_i^{\mathcal{N}_i\setminus\bar{R}_i}, \sigma_{i-1}] \le d \cdot \bar{q} \cdot \frac{1}{|\Sigma|}$$

which entails

$$\Pr[\mathrm{find} : U_i^{\mathcal{L}\setminus\bar{S}_i}, \rho_{i-1}] \le \mathrm{negl}(\lambda)$$

by using $\Pr[A] = \sum_{B=b} \Pr[A|B=b] \Pr[B=b]$ and the parameters $d, q \le \mathrm{poly}(\lambda)$, and $|\Sigma| = 2^{\Theta(\lambda)}$.

Plugging these into the last expression above (Equation (2)), yields $\left|\Pr[\mathrm{x} \in X_{\mathrm{valid}}|E : \mathrm{x} \leftarrow \mathcal{A}^{\mathcal{L}}] - \Pr[\mathrm{x} \in X_{\mathrm{valid}}|E : \mathrm{x} \leftarrow \mathcal{A}^{\mathcal{M}}|E]\right| \le \mathrm{negl}(\lambda)$ where we now state E explicitly. Using Equation (1) and the triangle inequality, we obtain the asserted result.

□

7.4 $\mathsf{QNC}_d^{\mathsf{BPP}}$ exclusion

Once the analysis of $\mathsf{QNC_d}$ is clear, extending it to $\mathsf{QC_d}$ is not too difficult. One needs to account for the actions of the intermediate classical circuits. The basic approach stays the same. We replace $\mathcal{L}$ with shadow oracles successively. The difference is that after each set of parallel queries, we account for the polynomially many queries made by the corresponding intermediate classical algorithm by exposing those queries in the subsequent shadow oracles.

7.4.1 Shadow oracles for $\mathsf{QC_d}$ hardness

The procedure for constructing base sets stays unchanged. We need the analogue of Algorithm 50. However, unlike Algorithm 50, this time the procedure cannot directly produce S_{ij} for all i, j, given the base sets. This is because the sets S_{ij} now must also depend on the queries made by the classical algorithm at intermediate steps.

Before we present the algorithm, we make the following assumption (which only makes the impossibility result stronger): the classical algorithm makes "path queries", i.e. suppose when it queries H_i at t_i, it learns all tuples $(t_0, t_1, t_2 \dots, t_i, \dots t_d)$ such that $H_{j-1}(t_{j-1}) = t_j$ for all $j \in \{1, \dots d\}$. Since H_0 cannot span the domain of H_1, t_0 may not always exist, corresponding to $(t_1, t_2 \dots t_d)$. More formally, we have the following.

Definition 54 (Path Queries)**.** Let $\mathcal{L}' := (H_0', \dots H_d')$ be as in Notation 43 and let $\bar{T}_i := (T_{i0}, T_{i1}, \dots T_{id})$ be a tuple of sets where for each $0 \le j \le d$, $T_{ij} \subseteq \Sigma^{d'}$. We say $\bar{T}_i$ are *path queries* if $T_{i1} \supseteq H_0(T_{i0})$, and $T_{ij} = H_{j-1}(T_{i,j-1})$ for all $j \in \{2, \dots, d\}$.

We can now define the algorithm. For context, it may help to recall that (see Notation 25) an arbitrary $\mathsf{QC_d}$ circuit with oracle access to $\mathcal{L}$ can be represented as

$$\mathcal{B}^{\mathcal{L}} := \Pi \circ \mathcal{A}_{c,d+1}^{\mathcal{L}} \circ \mathcal{B}_d^{\mathcal{L}} \circ \dots \mathcal{B}_1^{\mathcal{L}} \circ \rho_0$$

where $\mathcal{B}_i^{\mathcal{L}} := \Pi_i \circ \mathcal{L} \circ U_i \circ \mathcal{A}_{c,i}^{\mathcal{L}}$, ρ_0 is the initial state (in our case, encoding 1^λ) and Π is a measurement. Below, informally,[35] $\bar{T}_i$ corresponds to the set of queries made by the classical algorithm $\mathcal{A}_{c,i}$ to $\mathcal{L}$.

Algorithm 55 (Procedure for constructing S_{ij}, given $\bar{T}_i$s)**.** *Let $\mathcal{L} := (H_0, \dots H_d)$, Σ and S_i be as in Algorithm 49 and suppose the Algorithm 49 was executed.*

Input:

1. *The previous sequence of sets for creating the shadow oracle: $\bar{S}_{i-1} := (S_{i-1,j})_{j \in \{1 \dots d\}}$ where $S_{i-1,j} \subseteq S_{0,j}$ for all $j \in \{1, \dots d\}$.*
2. *The path queries made by the classical algorithm at step i: $\bar{T}_i := (T_{i0}, T_{i1}, T_{i2} \dots T_{id})$*

If Algorithm 49 aborts, define $S_{ij} = \varnothing$ for all $i, j \in \{1, \dots d\}$. If Algorithm 49 does not abort then, for each $i \in \{1, \dots d\}$ do the following.

1. *Define $S_{ik} = \varnothing$ for $1 \le k < i$.*
2. *Sample, uniformly at random, $S_{ii} \subseteq S_{i-1,i} \backslash T_{ii}$ such that $(S_i \cap S_{i-1,i}) \backslash T_{ii} \subseteq S_{ii}$ and $|S_{ii}|/|S_{i-1,i}| = 1/|\Sigma|$.*
3. *Define $S_{ik} = H_{k-1}(\dots H_i(S_{ii}) \dots)$ for $i < k \le d$.*

In both cases, return $\bar{S}_i := (S_{i1}, S_{i2} \dots S_{id})$.

7.4.2 Properties of the shadow oracles

Points

- The following could potentially be more generally stated.
- We take the set $\bar{S}_{i-1}$ to be given (we only impose the bare requirements), and have $\bar{T}_i$ be arbitrary poly sized sets

[35] We say informally because the queries $\mathcal{A}_{c,i}$ makes depends on the hybrid we are considering; these details appear later in the proof of Lemma 57.

- We show that given $\bar{S}_{i-1}$ and the sets $\bar{T}_i$, finding x in $\bar{S}_i$ would happen with probability $\mathrm{poly}(\lambda)/|\Sigma|$ at most.

Claim 56. Let $\mathcal{L}$ be as in Definition 35, run Algorithm 49 and let E be the event that it does not abort. Let $1 \le i \le d$. Obtain $\bar{S}_i$ by running Algorithm 55 with the following input:

1. If $i = 1$, use $\bar{S}_0$ generated by Algorithm 49.
   Else, if $i > 1$, let $\bar{S}_{i-1} := (S_{i-1,1}, S_{i-1,2} \dots S_{i-1,d})$ be arbitrary sets such that
   - for $j < i-1$, $S_{i-1,j} = \varnothing$,
   - for $j = i-1$, $S_{i-1,i-1} \subseteq S_{0,i-1}$ and $|S_{i-1,i-1}| = |\Sigma|^{d+2-(i-1)}| = |\Sigma^{d+1-i}|$ and finally
   - for $j > i-1$, $S_{i-1,j} \subseteq H_j(S_{i-1,j-1}) = H_j(\dots H_{i-1}(S_{i-1,i-1})\dots)$.
2. $\bar{T}_i := (T_{i0}, \dots T_{id})$ be arbitrary path queries (see Definition 54) such that $|T_{ij}| \le \mathrm{poly}(\lambda)$ for all $j \in \{0, \dots d\}$.

Then, it holds (for a large enough λ) that

$$\Pr[x \in S_{ik} | (S_{i-1,k}, T_i, E)] \le \mathrm{poly}(\lambda)/|\Sigma|$$

and

$$\Pr[x \in S_{ik} | (\check{\mathcal{L}}, E)] \le \mathrm{poly}(\lambda)/|\Sigma|$$

where $\check{\mathcal{L}}$ is $\mathcal{L}$ outside $(S_{i-1,1} \backslash T_{i1}, \dots S_{i-1,d} \backslash T_{id})$ (see Notation 43 with $S^{\mathrm{out}} \leftarrow (S_{i-1,j} \backslash T_{ij})_j$ and $\mathcal{L}' \leftarrow \mathcal{L}$) for all $1 \le i \le k \le d$ where the probability is over $\mathcal{L}$, the randomness in algorithm 55.

Before looking at the proof, we briefly comment on the claim. Item 1 is meant to enforce the form of the set $\bar{S}_{i-1}$ which would be produced by repeated applications of Algorithm 55. Therefore the first bullet ensures all sets before $i-1$ are empty, the second ensures the diagonal one has the right size (we start with $|\Sigma|^{d+2}$ for base sets and at each iteration, the size drops by $|\Sigma|$) and the last bullet ensures that the sets are no larger than if they were propogated through $\mathcal{L}$. Item 2 allows one to specify the classical queries made at the ith step. The statement says that if these inputs are used in Algorithm 55 to obtain the next sequence of sets, $\bar{S}_i$, then one can obtain a bound analogous to that of Claim 52. The difference is that this time, both the previous sequence of sets $\bar{S}_{i-1}$ and the classical queries $\bar{T}_i$ are revealed.

Proof sketch. The idea is the same as that we used in the proof of Claim 52. The only difference is that instead of considering the sets $S_{i-1,j}$, one considers $S'_{i-1,j} := S_{i-1,j} \backslash T_{i,j}$. Let $f(\lambda)$ be such that $|T_{ij}| \le f(\lambda)$ and suppose λ is large enough so that $|\Sigma| > f(\lambda)$. For the $k = i$ case, we get $x \in S_{i,i}$ is at most[36] $|S_{i,i}|/|S'_{i-1,i}| = \mathrm{poly}(\lambda)/|\Sigma|$. Similarly, for $k > i$, using Remark 47 (first observation) with $N \leftarrow |S_{i,k}| = |\Sigma^{d+2-i}|$ and $M \leftarrow |S'_{i-1,k}| = \left(|\Sigma^{d+1-i}| - \mathrm{poly}(\lambda)\right)$, one obtains that $x \in S_{ik}$ (conditioned on knowing T_{ik} and $S_{i-1,k}$ and E) with probability at most $N/M \le \mathrm{poly}(\lambda)/|\Sigma|$. □

7.4.3 d-CodeHashing is hard for $\mathsf{QC_d}$

We can now establish $\mathsf{QC_d}$ hardness of d-CodeHashing.

Lemma 57 (d-CodeHashing $\notin \mathsf{QNC}_d^{\mathsf{BPP}}$)**.** *Every $\mathsf{QC_d}$ circuit succeeds at solving d-CodeHashing (see Definition 35) with probability at most $\mathrm{negl}(\lambda)$ on input 1^λ for $d \le \mathrm{poly}(n)$.*

[36] We have

$$\begin{aligned} |S_{ii}|/|S'_{i-1,i}| &= |S_{ii}|/(|S'_{i-1,i}| - f) \\ &= |\Sigma|^{d'-i-1}/(|\Sigma|^{d'-1} - f) \\ &= \frac{1}{|\Sigma|(1 - f/|\Sigma|^{d'-i-1})} \\ &\le \frac{\mathrm{poly}(\lambda)}{|\Sigma|}. \end{aligned}$$

using, $(1-x)^{-1} \le 1 + x + \epsilon$ for small enough x, where $\epsilon > 0$ is some constant.

Proof. The proof is similar to that of Lemma 53. Again, we omit the input 1^λ when convenient. Let $\mathcal{L} := (H_0, \dots H_d)$ and Σ be as in Definition 35. Denote an arbitrary QC_d circuit, $\mathcal{B}^{\mathcal{L}}$ by

$$\mathcal{B}^{\mathcal{L}} := \Pi_\text{x} \circ \mathcal{A}^{\mathcal{L}}_{c,d+1} \circ \mathcal{B}^{\mathcal{L}}_d \circ \dots \mathcal{B}^{\mathcal{L}}_1$$

where $\mathcal{B}^{\mathcal{L}}_i := \Pi_i \circ \mathcal{L} \circ U_i \circ \mathcal{A}^{\mathcal{L}}_{c,i}$ and Π_x is a projector corresponding to output x. Let Π_valid be a projector on the set $X_\text{valid} = \{\text{x}\}$ of all correct solutions to Definition 35. Run Algorithm 49 and let E be the event that it does not abort. Note that

$$\left| \sum_{\text{x} \in X_\text{valid}} \Pr[\text{x} \leftarrow \mathcal{B}^{\mathcal{L}}] - \sum_{\text{x} \in X_\text{valid}} \Pr[\text{x} \leftarrow \mathcal{B}^{\mathcal{L}} | E] \right| \le \text{negl}(n). \tag{3}$$

Define

$$\mathcal{B}^{\mathcal{M}} := \Pi_\text{x} \circ \mathcal{A}^{\mathcal{L}}_{c,d+1} \circ \mathcal{B}^{\mathcal{M}}_d \circ \dots \circ \mathcal{B}^{\mathcal{M}}_1$$

where $\mathcal{B}^{\mathcal{M}}_i := \Pi_i \circ \mathcal{M}_i \circ U_i \circ \mathcal{A}^{\mathcal{L}}_{c,i}$ and $\mathcal{M}_i$ is the shadow oracle of $\mathcal{L}$ wrt $\bar{S}_i$ (see Definition 45). We are yet to define $\bar{S}_i$. Do the following for each $i \in (1, 2 \dots d)$. Suppose $\bar{S}_1, \dots \bar{S}_{i-1}$ (and therefore $\mathcal{M}_1, \dots \mathcal{M}_{i-1}$) have been defined and suppose $\mathcal{A}^{\mathcal{L}}_{c,i}$ makes path queries $\bar{T}_i = (T_{i0}, T_{i1}, \dots T_{id})$ to $\mathcal{L}$. Then, let $\bar{S}_i$ be the output of Algorithm 55 with $\bar{S}_{i-1}$ and $\bar{T}_i$ as input.

$\mathcal{B}^{\mathcal{M}}|E$ *cannot succeed with non-negligible probability:* We focus on the intermediate classical algorithms, $\{\mathcal{A}^{\mathcal{L}}_{c,i}\}_{i \in \{1,\dots,d+1\}}$ because the quantum parts have no access to $\tilde{H}$ (other than that already exposed by classical queries). Consider the labelling in Figure 7 and suppose that the input to $\mathcal{A}^{\mathcal{L}}_{c,i}$ is c'_{i-1} and its output is c_i. Similarly, suppose the input to $\Pi_i \mathcal{M}_i U_i$ is c_i (classical) and q_{i-1} (quantum) and its output is c'_i (classical) and q_i (quantum). Observe[37] that, c_i, c'_i, q_i at most reveal $\tilde{H}$ at $T_{i0} \cup T_{i-1,0} \dots \cup T_{1,0}$. Since $|T_{i0} \cup T_{i-1,0} \dots \cup T_{i,0}|$ is at most polynomial, from Theorem 34 (second part), we conclude that $\mathcal{A}^{\mathcal{L}}_{c,d+1}$ succeeds at solving CodeHashing with probability at most negligible. Note in particular, that since the quantum part, $\Pi_i \mathcal{M}_i U_i$ does not access $\tilde{H}$ outside T_{i0}, it can be classically simulated without making any calls to $\tilde{H}$. Consequently, one can treat the entire algorithm as a classical algorithm for applying Theorem 34 (second part) because the theorem statement only depends on the number of classical queries to $\tilde{H}$ (and not on the computational complexity of the circuit).

$\mathcal{B}^{\mathcal{M}}|E$ *and* $\mathcal{B}^{\mathcal{L}}|E$ *have practically the same behaviour:* We use a hybrid argument and the O2H lemma (see Lemma 41) to obtain the following (we dropped the $\circ$ symbol, the conditioning on E)

$$\begin{aligned} &\left| \sum_{\text{x} \in X_\text{valid}} \Pr[\text{x} \leftarrow \mathcal{B}^{\mathcal{L}}] - \sum_{\text{x} \in X_\text{valid}} \Pr[\text{x} \leftarrow \mathcal{B}^{\mathcal{M}}] \right| \\ &= \left| \Pi_\text{valid} \circ \mathcal{A}^{\mathcal{L}}_{c,d+1} \circ \mathcal{B}^{\mathcal{L}}_d \circ \dots \circ \mathcal{B}^{\mathcal{L}}_1 \circ \rho_0 - \Pi_\text{valid} \circ \mathcal{A}^{\mathcal{L}}_{c,d+1} \circ \mathcal{B}^{\mathcal{M}}_d \circ \dots \circ \mathcal{B}^{\mathcal{M}}_1 \circ \rho_0 \right| \\ &\le \sum_{i=1}^{d} B(\mathcal{B}^{\mathcal{L}}_i(\rho_{i-1}), \mathcal{B}^{\mathcal{M}}_i(\rho_{i-1})) \le \sum_{i=1}^{d} \sqrt{2 \Pr[\text{find} : U_i^{\mathcal{L} \backslash \bar{S}_i}, \mathcal{A}^{\mathcal{L}}_{c,i} \circ \rho_{i-1}]} \end{aligned} \tag{4}$$

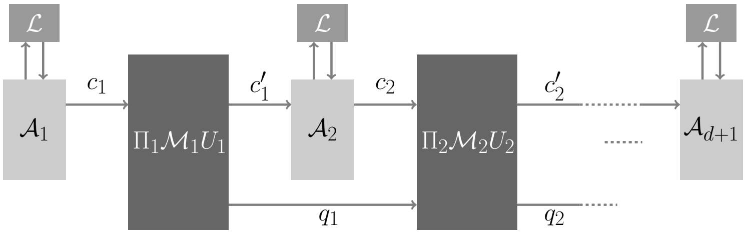


Figure 7: Illustration of the QC_d circuit, $\mathcal{B}^{\mathcal{M}}$, where all oracles have been replaced by shadow oracles. Note that we dropped $\circ$ between the operators for brevity.

[37] To see this, observe that

- c_1 at most reveals T_{10}
- both c'_1 and q_1 reveal at most T_{10}
- c_2 at most reveals $T_{20} \cup T_{10}$
- both c'_2 and q_2 reveal at most $T_{20} \cup T_{10}$
- and so on...

where for $i \in \{1, 2 \dots d-1\}$, $\rho_i := \mathcal{B}_i^{\mathcal{M}} \circ \dots \mathcal{B}_1^{\mathcal{M}} \circ \rho_0$. To bound the last expression, one can use Lemma 42 via Corollary 44 (recall that everything is conditioned on E). Let $\check{\mathcal{L}}_i$ be $\mathcal{L}$ outside $(S_{i1} \backslash T_{i+1,1}, \dots S_{id} \backslash T_{i+1,d})$ (see Notation 43 with $\mathcal{L}' \leftarrow \mathcal{L}$, $S^{\text{out}} \leftarrow (S_{ij} \backslash T_{i+1,j})_{j \in \{1 \dots d\}}$ and define $\check{\mathcal{L}}_i := \check{\mathcal{L}}'$) for each $i \in \{0, 1 \dots d\}$ (we include 0 to include the base sets specified by Algorithm 49). Similarly, let $\hat{\mathcal{L}}_i$ be $\mathcal{L}$ inside $(S_{i1} \backslash T_{i+1,1}, \dots S_{id} \backslash T_{i+1,d})$ (see Notation 43 with $\mathcal{L}' \leftarrow \mathcal{L}$, $S^{\text{out}} \leftarrow (S_{ij} \backslash T_{i+1,j})_j$ and $\hat{\mathcal{L}}_i := \hat{\mathcal{L}}'$). Note that the only information about $\mathcal{L}$ contained in $\mathcal{M}_i$, is at most $\check{\mathcal{L}}_i$, for each $i \in \{1, \dots d\}$ (at most because $\check{\mathcal{L}}$ also contains information queried by $\mathcal{A}_{c,i+1}^{\mathcal{L}}$). Consider $\Pr[\text{find} : U_i^{\mathcal{L} \backslash \bar{S}_i}, \mathcal{A}_{c,i}^{\mathcal{L}} \circ \rho_{i-1}]$ and note that $\mathcal{A}_{c,i}^{\mathcal{L}} \circ \rho_{i-1}$ at most specifies[38] $\check{\mathcal{L}}_{i-1}$ (about $\mathcal{L}$). Note also that the queries, $\bar{T}_i$, made by $\mathcal{A}_{c,i}^{\mathcal{L}}$ have been exposed in $\check{\mathcal{L}}_{i-1}$ and, furthermore, by construction (of Algorithm 55) are excluded from $\bar{S}_i$. Let $\sigma_i := \mathcal{A}_{c,i}^{\mathcal{L}} \circ \rho_{i-1} | \check{\mathcal{L}}_{i-1}$, $\bar{R}_i := \bar{S}_i | \check{\mathcal{L}}_{i-1}$ and $\mathcal{N}_i := \mathcal{L} | \check{\mathcal{L}}_{i-1}$. After conditioning, σ_i is uncorrelated to $\bar{R}_i$ (because once $\check{\mathcal{L}}_{i-1}$ is fixed (which also fixes $\bar{T}_i$), σ_i contains no information about how $\mathcal{L}$ behaves in $\bar{S}_{i-1} \backslash \bar{T}_i$ and $\bar{R}_i$ depends only on the randomness in Algorithm 55 and on $\hat{\mathcal{L}}_{i-1}$). One can thus apply Corollary 44 with Claim 56 to obtain

$$\Pr[\text{find} : V_i^{\mathcal{N}_i \backslash \bar{R}_i}, \sigma_{i-1}] \le d \cdot \bar{q} \cdot \frac{\text{poly}(\lambda)}{|\Sigma|}$$

which entails

$$\Pr[\text{find} : U_i^{\mathcal{L} \backslash \bar{S}_i}, \mathcal{A}_{c,i}^{\mathcal{L}} \circ \rho_{i-1}] \le \text{negl}(\lambda)$$

by using $\Pr[A] = \sum_{B=b} \Pr[A|B=b] \Pr[B=b]$ and the parameters $d, q \le \text{poly}(\lambda)$ and $|\Sigma| = 2^{\lambda^{\Theta(1)}}$.

Plugging these into Equation (4), yields $\left| \Pr[\mathrm{x} \in X_{\text{valid}} | E : \mathrm{x} \leftarrow \mathcal{B}^{\mathcal{L}}] - \Pr[\mathrm{x} \in X_{\text{valid}} | E : \mathrm{x} \leftarrow \mathcal{B}^{\mathcal{M}}] \right| \le \text{negl}(\lambda)$ where we now state conditioning on E explicitly. Using Equation (3) and the triangle inequality, we obtain the asserted result.

□

7.5 $\mathsf{BPP}^{\mathsf{QNC}_d}$ exclusion — Warm up

Establishing $\mathsf{CQ_d}$ hardness takes more work. We briefly outline the approach first and formalise it in the following sections. We take inspiration from [CCL20] and adapt the implementation/formalism introduced in [AGS22]. Let $\mathcal{L} := (H_0, \dots H_d)$ be as defined in Definition 35.

Consider a $\mathsf{CQ_d}$ circuit. To show that it cannot solve d-$\mathsf{CodeHashing}$, the first quantum part, can be analysed as we did the QNC_d part (using domain hiding). Let the output of this quantum part be a string s_1 and suppose the "paths" queried by the subsequent classical part be Y_1. To analyse the subsequent quantum part, one could expose (in the shadow oracles) the paths uncovered by Y_1 (as we did in the analysis of $\mathsf{QC_d}$ circuits, albeit there we had to do it after every unitary layer). However, this is not enough because the string s_1 is correlated to the oracle $\mathcal{L}$ and it is unclear how our techniques would work with $\mathcal{L}|s_1$ instead of $\mathcal{L}$. It turns out that if the string s_1 appears with non-negligible probability, then $\mathcal{L}|s_1$ can be viewed as a "convex combination" of $\mathcal{L}$ with a polynomial number of "paths" fixed. One can then proceed (almost) as in the QNC_d case for the next second quantum part. This procedure can be iterated polynomially many times to yield the desired hardness.

Before we can make any of this precise, we need to introduce the sampling argument. While the following overlaps with the informal discussion presented in the Technical Overview, there are more details and precise statements.

7.6 Technical Results II — The sampling argument

We first describe the sampling argument in its simplest form and subsequently show how to lift the result to our setting of interest.

7.6.1 Warm up — Sampling argument for Permutations

We informally describe the prerequisites to state the sampling argument for permutations, deferring formal definitions and proofs to Section C in the Appendix. We are being slightly redundant below to aid readibility (we overlap slightly with Section 2).

[38] For $i = 1$, $\check{\mathcal{L}}_{i-1} = \check{\mathcal{L}}_0$ is $\mathcal{L}$ outside $\bar{S}_0$ (which is rather lenient because ρ_0 contains no information about $\mathcal{L}$; to be precise, one could have used $(\Sigma^{d'}, \dots \Sigma^{d'})$ instead of $\bar{S}_0$).

Suppose t is a permutation over N elements labelled $\{0,\dots,N-1\}$. This permutation t is ordinarily viewed as a function, $t(x)$ which specifies how x is mapped. However, one could equivalently view t as a collection of tuples (x,y) such that $t(x)=y$. We call such a tuple a "path" and any set of such "paths" a "part".

Now consider distributions over permutations. Let's begin with a uniform distribution $\mathbb{F}$ over all permutations u. One may characterise $\mathbb{F}$ as follows: for any $u\sim\mathbb{F}$, i.e. any u sampled from $\mathbb{F}$, it holds that $\Pr[u(x)=y]=\Pr[(x,y)\in \mathrm{paths}(u)]=(N-1)!/N!$. In fact, it also holds that $\Pr[S\subseteq \mathrm{paths}(u)]=(N-|S|)!/N!$ where S is a collection of (non-colliding) paths. It turns out that this way of viewing the uniform distribution helps us below.

We first state a basic version of the sampling argument. To this end, we define a (p,δ) *non-uniform distribution*, $\mathbb{F}^{(p,\delta)}$, which is closely related to the uniform distribution $\mathbb{F}$. At a high level, $\mathbb{F}^{(p,\delta)}$ is "δ close to" $\mathbb{F}$ with at most p many paths fixed. What does "δ closeness" mean? For any distribution $\mathbb{G}$ (over permutations), a distribution $\mathbb{G}^{\delta}$ is δ close to it if the following holds: when $t'\sim\mathbb{G}^{\delta}$ and $t\sim\mathbb{G}$, one has $\Pr[S\subseteq \mathrm{paths}(t')]\le 2^{\delta|S|}\Pr[S\subseteq\mathrm{paths}(t)]$ for all parts S.

We are almost ready to state the basic sampling argument. We need the notion of a "convex combination" of random variables. We say a random variable (such as our permutation) t is a convex combination of random variables t_i, denoted by $t\equiv\sum_i\alpha_i t_i$ (where $\sum_i\alpha_i=1$ and $\alpha_i\ge 0$), if the following holds for all t': $\Pr[t=t']=\sum_i\alpha_i\Pr[t_i=t']$.

Informally, the basic sampling argument is a statement about a uniform permutation $u\sim\mathbb{F}$ and how the distribution $\mathbb{F}$ changes if we are given some "advice" about this permutation which is simply a function $g(u)$. Roughly speaking, given that $g(u)$ evaluates to r with probability at least 2^{-m}, the distribution $\mathbb{F}$ conditioned on r is a convex combination[39] of $\mathbb{F}^{(p,\delta)}$ distributions where the number of paths fixed is at most $p=2m/\delta$. Here δ is a free parameter. We slightly abuse the notation and write this basic sampling argument as

$$\mathbb{F}|r\equiv\mathrm{conv}(\mathbb{F}^{(p,\delta)}).$$

The formal statement is as follows.

Proposition 58 ($\mathbb{F}|r\equiv\mathrm{conv}(\mathbb{F}^{(p,\delta)})$)**.** *Let $u\sim\mathbb{F}(N)$ be a uniformly random permutation over $N=2^n$ elements and $g(u)$ be an arbitrary function. Fix any $\delta>0$, $\gamma=2^{-m}>0$ where $m=m(n)$ and suppose* $\Pr[g(u)=r]\ge\gamma$*. Then*

$$t\equiv\sum_i\alpha_i t_i+\gamma' t'$$

where $t=u|(g(u)=r)$, $t_i\sim\mathbb{F}_i^{(p,\delta)}$ and $\mathbb{F}_i^{(p,\delta)}$ is (p,δ) non-uniform with $p=\frac{2m}{\delta}$. The coefficients sum to 1, i.e. $\sum_i\alpha_i+\gamma'=1$ and the number of coefficients is finite. The permutation t' is sampled from an arbitrary (but normalised) distribution over permutations and $\gamma'\le\gamma$.

If we view $g(u)$ as the output of the first quantum part of our $\mathsf{CQ_d}$ circuit, and u as the oracle of interest (details are in the next section), it is suggestive that $u|g(u)$ will be the oracle for the second quantum part of $\mathsf{CQ_d}$. We can use the sampling argument above and re-use our analysis because $\mathbb{F}$ and $\mathbb{F}^{(p,\delta)}$ have very similar statistical properties. However, it is unclear how to use the sampling argument thereafter as the basic sampling argument seems to only apply to $\mathbb{F}$ (and not to $\mathbb{F}^{(p,\delta)}$).

To state the more general version of the sampling argument, we need to define a (p,δ) *non-β-uniform distribution* $\mathbb{F}^{(p,\delta)|\beta}$. Just as we defined $\mathbb{F}^{(p,\delta)}$ using $\mathbb{F}$, we can define $\mathbb{F}^{(p,\delta)|\beta}$ using $\mathbb{F}^{|\beta}$, i.e. $\mathbb{F}^{(p,\delta)|\beta}$ is a distribution which is "δ close to" the β-uniform distribution $\mathbb{F}^{|\beta}$, with at most p many paths fixed. It remains to define $\mathbb{F}^{|\beta}$. In this case, $\beta:=\{(x_i,y_i)\}_i$ simply specifies an explicit set of paths contained in the uniform distribution $\mathbb{F}$. Note that these paths are distinct from those associated with p. Why do we introduce β when p was already present? The parameter p simply says there *exist* at most p paths which are fixed while β *explicitly* fixes certain paths. This becomes useful in stating the (general) sampling argument.

Suppose we start with $t\sim\mathbb{F}^{\delta'|\beta}$ (i.e. a distribution which is "δ' close to" β-uniform) and are given some advice $h(t)$ which happens to be r with probability at least 2^{-m}. Then the distribution $\mathbb{F}^{\delta'|\beta}$ conditioned on r is, roughly speaking, a convex combination[40] of $\mathbb{F}^{(p,\delta+\delta')|\beta}$ distributions where the number of paths fixed is at most $p=2m/\delta$ and δ again is a free parameter. Using the previous shorthand, we have

$$\mathbb{F}^{\delta'|\beta}|r\equiv\mathrm{conv}(\mathbb{F}^{(p,\delta+\delta')|\beta}).$$

The formal statement is as follows.

[39] In the convex combination, there is a small component, of weight at most 2^{-m}, of some arbitrary distribution.

[40] Again, neglecting a component with weight at most 2^{-m}.

Proposition 59 ($\mathbb{F}^{\delta'|\beta}|r' = \text{conv}(\mathbb{F}^{(p,\delta+\delta')|\beta})$)**.** *Let $t \sim \mathbb{F}^{\delta'|\beta}(N)$ be sampled from a δ' non-β-uniform distribution with $N = 2^n$. Fix any $\delta > 0$ and let $\gamma = 2^{-m}$ be some function of n. Let $s \sim \mathbb{F}^{\delta'|\beta}|r$, i.e. $s = t|(h(t) = r)$ and suppose $\Pr[h(t) = r] \geq \gamma$ where h is an arbitrary function and r some string in its range. Then s is "γ-close" to a convex combination of finitely many $(p, \delta + \delta')$ non-β-uniform distributions, i.e.*

$$s \equiv \sum_i \alpha_i s_i + \gamma' s'$$

where $s_i \sim \mathbb{F}_i^{p,\delta+\delta'|\beta}$ with $p = 2m/\delta$. The permutation s' may have an arbitrary distribution (over $\Omega(2^n)$) but $\gamma' \leq \gamma$.

How does this solve the limitation of the basic sampling method—which was, how do we apply the sampling argument to $\mathbb{F}^{(p',\delta')}$? Using the observation that $\mathbb{F}^{(p',\delta')} = \mathbb{F}^{\delta'|\beta}$ for some β which fixes at most p' paths, it is not hard to see that the sampling argument yields

$$\mathbb{F}^{(p',\delta')}|r \equiv \text{conv}(\mathbb{F}^{(p+p',\delta'+\delta)}),$$

and in particular, if the procedure is successively applied $\tilde{n} \leq \text{poly}(n)$ times (starting with $\mathbb{F}$), the convex combination would be over distributions of the form $\mathbb{F}^{(\tilde{n}p,\tilde{n}\delta)}$. How the parameters are chosen is discussed later.

The proofs of these statements do not rely on any special property of the distribution $\mathbb{F}$ nor do they depend on the fact that we were considering permutations. Any object for which we can describe a "reasonable" notion of "parts" admits such a sampling argument. We don't attempt to formalise what we mean by "reasonable"—we simply construct such a notation for our oracle and inspect that the properties required in the proof are satisfied.

7.6.2 Definitions and Notation — Sampling argument for Injective Shufflers

As we did for permutations, to describe the sampling argument, we change our viewpoint and look at probabilities associated with "paths" in $\mathcal{L} = (H_0, \dots H_d)$ instead of looking at probabilities associated with the individual outcomes of H_is. By a "path", we mean tuples of the form $(x_0, x_1 \dots)$ such that $x_i = H_{i-1}(x_{i-1})$ for all i.

This viewpoint is inadequate for capturing the probabilistic behaviour of $\mathcal{L}$ due to two reasons (which are not hard to rectify). *First*, since $H_0 : \Sigma \to \Sigma^{d'}$, it is clear that at least $\left|\Sigma^{d'-1}\right|$ many points will never be contained in any "path" as described above. Therefore the behaviour of most points in H_i (for $i \in \{1 \dots d\}$) will not be captured by the "paths" viewpoint. *Second*, even though H_i maps $\Sigma^{d'} \to \Sigma^{d'}$ for $i \in \{1, \dots d-1\}$, H_i may not be injective and therefore the paths might collide, which again would mean the behaviour of many points would not be captured by the "paths" viewpoint.

To rectify the *second* issue, we can run Algorithm 49 and condition on the event E, i.e. that the algorithm does not abort. Since in our proofs, we only care about the behaviour of $\mathcal{L}$ on $\bar{S}_0 = (S_{01}, \dots S_{0d})$, it suffices to restrict our attention to $\bar{S}_0$. By construction (of Algorithm 49), $\mathcal{L}|E$ behaves as a permutation on $\bar{S}_0$. Therefore no "path" inside $\bar{S}_0$ collides. To rectify the *first* issue, we consider two kinds of paths—Type 0 paths and Type 1 paths.[41] A *Type 0 path* is what we described earlier: a tuple of the form $(x_0, x_1 \dots)$ such that $x_i = H_{i-1}(x_{i-1})$ for all i. A *Type 1 path* is a tuple of the form $(\textvisiblespace, x_1, x_2 \dots)$ such that $x_1 \notin H_0(\Sigma)$ (i.e. $\nexists x_0$ st $H_0(x_0) = x_1$) and $x_i = H_{i-1}(x_{i-1})$ for all $i \in \{2, 3 \dots\}$.

Observe that, restricted to $\bar{S}_0$ and conditioned on E, we have the following equivalence: given $\Pr[H_i(x) = x']$ for all i, x and x', one can compute the probability associated with both types of paths and conversely, given probabilities associated with the paths, one can compute $\Pr[H_i(x) = x']$ for all i, x and x'.

To simplify the notation, we define the *injective shuffler*. Fix sets $S_{0i} \subseteq \Sigma^{d'}$ of size $|\Sigma^{d+2}|$ for all $i \in \{1, \dots d\}$. Let $H'_0 : \Sigma \to S_{01}$, $H'_i : S_{0i} \to S_{0,i+1}$ for all $i \in \{1, \dots d-1\}$ be injective functions and let $H'_d : S_{0d} \to \{0,1\}^n \cup \{\perp\}$ (which may not be injective) such that H'_d outputs $\perp$ for all paths originating from Σ (and no other).[42] We define the *injective shuffler*, $\mathcal{K}$ as $(H'_0, \dots H'_d)$. Think of $\mathcal{K}$ as a simpler way to denote the relevant object associated $\mathcal{L}|E$ (with $\bar{S}_0$ being the output of Algorithm 49). What do we mean by the relevant object—as we saw in the QNC_d and QC_d analysis, it helps to use shadow oracles in the analysis which never reveal any

[41] The 0 and 1 represent where the first non-$\textvisiblespace$ component sits.

[42] i.e. $H'_d(x_d) = \perp$ iff $(x_0, x_1, \dots x_d, x_{d+1})$ is a Type 0 path (therefore $x_{d+1} = \perp$)

information[43] about the values taken by $H_d(\dots(H_0(\ell))\dots)$ for any $\ell \in \Sigma$. We capture this limitation in $\mathcal{K}$ by ensuring H'_d outputs $\perp$ for these queries.

To state the sampling argument for the injective shuffler, we define (p,δ) non-β-uniform distributions for the injective shuffler (analogous to the way we defined them for permutations). However, this time we also give formal definitions (it may help to look at the analogous formal definitions for permutations first, as detailed in Section C of the Appendix). We begin with the uniform distribution—it is simply a distribution which assigns equal probabilities to all the possible injective shufflers, given the sets $(S_{0i})_i$. As for β-uniform distributions, we first need to define the "paths", β. Here, β will again be a set of "non-colliding paths" but formalising this requires some care (discussed later). Then a β-uniform distribution is the same as the uniform distribution except that the paths in β are fixed.

We first define "base sets" for convenience as they are repeatedly used in this section. Using these, we define (valid) injective shuffler wrt base sets. Then, one can trivially define $\mathbb{F}_{\text{shuff}}$, as the uniform distribution over injective shufflers.

Definition 60 (Base sets)**.** Let Σ and d' be as in Definition 35. For each $i \in \{1,\dots d\}$, suppose $S_{0i} \subseteq \Sigma^{d'}$ are subsets of size $|\Sigma^{d+2}|$ then we call $\bar{S}_0 := (S_{01}, \dots S_{0d})$ *base sets.*

Definition 61 ((valid) Injective Shuffler wrt base sets $\bar{S}_0$.)**.** Let Σ, d' be as in Definition 35 and let $\bar{S}_0 = (S_{01}\dots S_{0d})$ be a base set (see Definition 60). Then a *(valid) Injective Shuffler* wrt $\bar{S}_0$ is a sequence of functions $(H'_0 \dots H'_d)$ where $H'_0 : \Sigma \to S_{01}$, $H'_i : S_{0i} \to S_{0,i+1}$ for all $i \in \{1\dots d-1\}$ are injective functions and $H'_d : S_{0d} \to \{0,1\}^n \cup \{\perp\}$ is an arbitrary function satisfying the following constraint:

$$H'_d(x) \in \begin{cases} \{\perp\} & x \in H'_{d-1}(\dots H'_0(\Sigma)\dots) \\ \{0,1\}^n & x \in S_{0d} \backslash H'_{d-1}(\dots H'_0(\Sigma)\dots). \end{cases}$$

The conditions on $H'_0, \dots H'_{d-1}$ are straightforward. The conditions on H'_d ensures that all paths originating from Σ (i.e. Type 0 paths) output $\perp$ which, as we remarked earlier, ensures our definition can be used with shadow oracles.

Definition 62 ($\mathbb{F}_{\text{inj}}$—Uniform Distribution over Injective Shufflers)**.** Let $\bar{S}_0$ be base sets (see Definition 60) for d-$\mathsf{CodeHashing}$ (see Definition 35). Then $\mathbb{F}_{\text{inj}}$ is the uniform distribution over all injective shufflers wrt $\bar{S}_0$ (see Definition 61).

So far everything was intuitive. To proceed, we would need to condition these injective shufflers. The conditioning will be in terms of existence of certain non-colliding paths, β, in the injective shuffler. There are two subtleties when we do this, as we alluded to. The *first* is that there are Type 0 and Type 1 paths and thus one must be careful in how collisions are defined. The *second* is that an injective shuffler is defined to yield $\perp$ on paths originating from Σ (i.e. on Type 0 paths) and yet, (as we shall see) we would like to be able to condition on polynomially many paths originating from Σ which yield non-$\perp$ responses. This corresponds to (excluding from the shadow oracles) the paths queried by the classical algorithm because the classical algorithm will have access to $\mathcal{L}$ (and not its shadow). These concerns are addressed in the following definition.

Definition 63 ((valid) paths, β, wrt $\bar{S}_0$. $X_i(\beta)$)**.** Let $\bar{S}_0$ be base sets (see Definition 60) for d-$\mathsf{CodeHashing}$ (see Definition 35) and let $\beta = \{(x_{j,0}, \dots x_{j,d+1})\}_{j\in\{1\dots|\beta|\}}$ be a set of tuples with $d+2$ elements. We say β specifies (valid) paths wrt $\bar{S}_0$ if it satisfies the following:

1. (domain validation) For each $j \in \{1\dots|\beta|\}$, it holds that (a) $x_{j,0} \in \Sigma \cup \{\text{\textvisiblespace}\}$; (b) for all $i \in \{1\dots d\}$, $x_{j,i} \in S_{0i}$ and (c) $x_{j,d+1} \in \{0,1\}^n$ (but cannot output $\perp$),

2. (no collisions) for each distinct pair $j, j' \in \{1\dots|\beta|\}$, $x_{j,i} \neq x_{j',i}$, for all $i \in \{1\dots d\}$ and

3. (handling Type 0 paths) for any distinct pair $j, j' \in \{1\dots|\beta|\}$, $x_{j,0} = x_{j',0} \iff x_{j,0} = x_{j',0} = \text{\textvisiblespace}$.

Notation: For (valid) paths β, define (for $i \in \{0, \dots d+1\}$

- $X_i(\beta) := \{x_{j,i}\}_{j\in\{1\dots|\beta|\}}$, using this, define $X_{i:i'}(\beta) = (X_i(\beta), \dots X_j(\beta))$ for $i \leq i'$ and let $X(\beta) = X_{1:d}(\beta)$,

- $X_i^{(0)}(\beta) := \{x_{j,i}\}_{j:x_{j0}\neq\text{\textvisiblespace}}$, and

[43] Except for polynomially possibly many paths exposed by classical queries; we handle these shortly.

- $X_i^{(1)}(\beta) := \{x_{j,i}\}_{j:x_{j0}=\llcorner\lrcorner}$.

The first condition simply requires that the paths are inside $\bar{S}_0$. The second condition ensures that the paths don't collide but excluding the first component. The third condition ensures that the only way the first component can "collide" is if the path is Type 1; Type 0 paths cannot have the same first component. With (valid) paths β defined, we can define a (valid) injective shuffler conditioned on β and the associated uniform distribution.

Definition 64 ((valid) Injective Shuffler conditioned on β wrt base sets $\bar{S}_0$)**.** Let Σ, d' be as in Definition 35, let $\bar{S}_0 = (S_{01} \dots S_{0d})$ be base sets (see Definition 60) and let $\beta =: \{(x_{j,0}, \dots x_{j,d+1})\}_{j\in\{1\dots|\beta|\}}$ denote (valid) paths wrt $\bar{S}_0$ (see Definition 63). Then, a (valid) Injective Shuffler conditioned on β wrt $\bar{S}_0$ is a sequence of functions $(H'_0, \dots H'_d)$ where $H'_0 : \Sigma \to S_{01}$, $H'_i : S_{0i} \to S_{0,i+1}$ for all $i \in \{1, \dots d-1\}$ are injective functions and $H'_d : S_{0d} \to \{0,1\}^n \cup \{\perp\}$ is an arbitrary function which satisfy the following constraints:

- H'_0: it holds that $H'_0(x_{j0}) = x_{j1}$ for all $j \in \{1 \dots |\beta|\}$ such that $x_{j0} \neq \llcorner\lrcorner$ and $H'_0(\Sigma) \cap X_1^{(1)}(\beta) = \varnothing$ (see Definition 63)
- H'_i: it holds that $H'_i(x_{j,i}) = x_{j,i+1}$ for all $i \in \{1 \dots d-1\}$ and $j \in \{1 \dots |\beta|\}$
- H'_d: it holds that
  1. $H'_d(x_{j,d}) = x_{j,d+1}$ for all $j \in \{1 \dots |\beta|\}$
  2. $H'_d(x) = \perp$ for all $x \in H'_{d-1}(\dots H'_0(\Sigma)\dots)\backslash X_d(\beta)$
  3. $H'_d(x) \in \{0,1\}^n$ otherwise, i.e. for all $x \in S_{0d} \backslash \left(H'_{d-1}(\dots H'_0(\Sigma)\dots) \cup X_d(\beta)\right)$.

The requirements on $H'_1, \dots H'_{d-1}$ are quite clear. On H'_0, the first condition is enforcing consistency with Type 0 paths and the second one is enforcing that none of the Type 1 paths could possibly have originated from[44] Σ. For H'_d, we enforce that it is consistent with the

paths in β and that it outputs $\perp$ for all remaining paths originating in Σ (Type 1 paths) while for all other paths, it outputs non-$\perp$. We can finally define the uniform distribution over injective shufflers conditioned on β.

Definition 65 ($\mathbb{F}_{\text{inj}}^{|\beta}$—$\beta$-uniform distribution over injective shufflers)**.** Let $\bar{S}_0$ be base sets (see Definition 60) for d-$\mathsf{CodeHashing}$ (see Definition 35), and let β denote a (valid) set of paths wrt $\bar{S}_0$ (see Definition 63). Then, $\mathbb{F}_{\text{inj}}^{|\beta}$ is the uniform distribution over all (valid) injective shufflers conditioned on β wrt $\bar{S}_0$ (see Definition 64).

We can now introduce some notation for describing paths of injective shufflers. These paths are slightly different from (valid) paths β wrt $\bar{S}_0$ (see Definition 63)—these paths must assign $\perp$ to paths originating from Σ (Type 0 paths) to any injective shuffler.[45] This is required to stay consistent with the definition of injective shufflers.

We use these paths to define the parts notation explicitly. These in turn, would allow us to easily obtain the analogue of Proposition 59 for injective shufflers.[46]

Notation 66. Let Ξ be an injective shuffler (possibly conditioned on paths β) wrt base sets $\bar{S}_0$ (see Definition 64). Denote by

- $\mathsf{func}_{i,\Xi}$ the function H'_i where $(H'_0, \dots H'_d) := \Xi$
- $\mathsf{cfunc}_{i:j,\Xi}$ the function $H'_j(\dots H'_i(\cdot)\dots)$ where H'_i is as above for $i, j \in \{0 \dots d\}$ satisfying $i \leq j$.
- $\text{paths}(\Xi)$ the set of all tuples $(x_0, x_1 \dots x_d, x_{d+1})$ where $x_0 \in \Sigma \cup \{\llcorner\lrcorner\}$, $x_1 \in S_{01}, \dots, x_d \in S_{0d}$ and $x_{d+1} \in \{0,1\}^n \cup \{\perp\}$ satisfy the following
  - for $i \in \{1, \dots d\}$, it holds that $x_{i+1} = \mathsf{func}_{i,\Xi}(x_i)$
  - if $x_0 = \llcorner\lrcorner$, it holds that $x_1 \notin \mathsf{func}_{0,\Xi}(\Sigma)$
  - if $x_0 \in \Sigma$, it holds that $x_1 = \mathsf{func}_{0,\Xi}(x_0)$

[44] The reason is that that this avoids double counting; otherwise a Type 1 path could be treated as a partially specified Type 0 path and our sampling argument is not a priori robust to these.

[45] If the injective shuffler is conditioned on β, then the statement holds excluding the Type 1 paths specified by β.

[46] Notation: We are using both $\mathcal{K}$ and Ξ to refer to injective shufflers.

As stated, we now describe the parts notation for injective shufflers.

Notation 67. Suppose β is a (valid) path wrt base sets $\bar{S}_0$. Let Ξ be an arbitrary injective shuffler conditioned on β wrt base sets $\bar{S}_0$ (see Definition 64).

- *Parts:* Any set S is a *part* if it holds that $S \subseteq \text{paths}(\Xi)$ for some Ξ.
  - Denote by $\Omega^{\beta}_{\text{parts}}$ the set of all such "parts".
  - Call two parts $S, S' \in \Omega^{\beta}_{\text{parts}}$ *distinct* if $S \cap S' = \varnothing$ and $S \cup S' \subseteq \text{paths}(\Xi)$ for some Ξ.
  - Denote by $\Omega^{\beta}_{\text{parts}}(S)$ the set of all parts $S' \in \Omega^{\beta}_{\text{parts}}$ distinct from S.
- Suppose Ξ is a random variable.
  - *Probability of a part S:* The probability that Ξ maps paths as described in S is denoted by $\Pr[S \subseteq \text{paths}(\Xi)]$.
  - *Conditioning Ξ on a part:* We use the notation Ξ_S to denote the random variable Ξ conditioned on the event $S \subseteq \text{paths}(\Xi)$.

Before we use these definitions for stating and proving the sampling argument for injective shufflers, we use them to define (p, δ) non-β-uniform distributions for injective shufflers.

Definition 68 ($\mathbb{G}^{(p,\delta)|\beta}$—a (p,δ) non-$\mathbb{G}^{|\beta}$ distribution)**.** Suppose β is a valid path wrt base sets $\bar{S}_0$ (see Definition 63). Let $\Xi \sim \mathbb{G}^{|\beta}$ be a an injective shuffler conditioned on β wrt $\bar{S}_0$ (see Definition 64), sampled from some arbitrary distribution $\mathbb{G}^{|\beta}$. Let $p, \delta \geq 0$. Then, we say $\Xi' \sim \mathbb{G}^{(p,\delta)|\beta}$ is sampled from a (p,δ) non-$\mathbb{G}^{|\beta}$ distribution[47] if for all parts $S \in \Omega^{\beta}_{\text{parts}}(S')$ it holds that

$$\Pr[S \subseteq \text{paths}(\Xi') | S' \subseteq \text{paths}(\Xi')] \leq 2^{|S|\delta} \Pr[S \subseteq \text{paths}(\Xi) | S' \subseteq \text{paths}(\Xi)]$$

for some part $S' \in \Omega^{\beta}_{\text{parts}}$ of size $|S'| \leq p$.

Using different distributions in place of $\mathbb{G}^{|\beta}$, one can obtain the following which will be relevant to the sampling argument.

Notation 69. The distribution specified in Definition 68

- with $\mathbb{F}^{|\beta}_{\text{inj}} \leftarrow \mathbb{G}^{|\beta}$, $0 \leftarrow p$ is termed $\mathbb{F}^{\delta|\beta}_{\text{inj}}$,
- with $\mathbb{F}^{|\beta}_{\text{inj}} \leftarrow \mathbb{G}^{|\beta}$ is termed $\mathbb{F}^{(p,\delta)|\beta}_{\text{inj}}$, and
- with $\mathbb{F}^{\delta'|\beta}_{\text{inj}} \leftarrow \mathbb{G}^{|\beta}$ is termed $\mathbb{F}^{(p,\delta+\delta')|\beta}_{\text{inj}}$.

We call $\mathbb{F}^{(p,\delta)|\beta}_{\text{inj}}$ a (p,δ) non-β-uniform distribution.

7.6.3 Statement — Sampling argument for Injective Shufflers

We now state the sampling argument and prove its basic variant to convey the idea, deferring the general proof to the appendix.

Proposition 70 ($\mathbb{F}^{\delta'|\beta}_{\text{inj}}|r' \equiv \text{conv}(\mathbb{F}^{(p,\delta+\delta')|\beta}_{\text{inj}})$)**.** *Suppose β is a valid path wrt base sets $\bar{S}_0$ (see Definition 63). Let $\Xi^t \sim \mathbb{F}^{\delta'|\beta}_{\text{inj}}$ be sampled from a δ' non-β-uniform distribution. Fix any $\delta > 0$ and let $\gamma = 2^{-m}$ be some function of n (where n is as in Definition 35). Let $\Xi^s \sim \mathbb{F}^{\delta'|\beta}_{\text{inj}}|r$, i.e. $\Xi^s := \Xi^t|(h(\Xi^t) = r)$ and suppose that $\Pr[h(\Xi^t) = r] \geq \gamma$ where h is an arbitrary function and r some string in its range. Then Ξ^s is "γ-close" to a convex combination of finitely many injective shufflers sampled from $(p, \delta + \delta')$ non-β-uniform distributions, i.e.*

$$\Xi^s \equiv \sum_i \alpha_i \Xi^s_i + \gamma' \Xi^{s\prime}$$

where there are finitely many α_i, $\sum_i \alpha_i + \gamma' = 1$, $\Xi^s_i \sim \mathbb{F}^{(p,\delta+\delta')|\beta}_{\text{inj},i}$ with[48] *$p = 2m/\delta$. The injective shuffler (conditioned on β), $\Xi^{s\prime}$, may have arisen from an arbitrary distribution, however, $\gamma' \leq \gamma$.*

[47] (over injective shuffler conditioned on β)

[48] (the i in $\mathbb{F}^{(p,\delta+\delta')|\beta}_{\text{inj},i}$, indicates that each Ξ^s_i can come from a different distribution which is still $(p, \delta + \delta')$ non-β-uniform; e.g. they may be fixing different paths but there are at most p such paths)

7.6.4 Properties of the δ non-β-uniform injective shuffler

Suppose $\mathcal{L}' \sim \mathbb{F}_{\text{inj}}$ and $\Xi \sim \mathbb{F}^{\delta}_{\text{inj}}$. It would be useful to go back from the paths perspective to functions and see how their behaviour is related, i.e. we relate the behaviour of $\mathsf{func}_{i,\mathcal{L}'}$ to that of $\mathsf{func}_{i,\Xi}$ (or more generally, $\mathsf{cfunc}_{i:j,\mathcal{L}'}$ to that of $\mathsf{cfunc}_{i:j,\Xi}$).

Claim 71. Suppose $\Xi \sim \mathbb{F}^{|\beta}_{\text{inj}}$ and $\mathcal{K} \sim \mathbb{F}^{\delta|\beta}_{\text{inj}}$ be injective shufflers wrt base sets $\bar{S}_0$ (see Definition 61). Then, for all $x_i \in S_{0,i}$ and $x_{i+1} \in S_{0,i+1}$, it holds that

$$\Pr[\mathsf{cfunc}_{i:j,\mathcal{K}}(x_i) = x_{j+1}] \leq 2^{\delta} \Pr[\mathsf{cfunc}_{i:j,\Xi}(x_i) = x_{j+1}]$$

which in particular entails

$$\Pr[\mathsf{func}_{i,\mathcal{K}}(x_i) = x_{i+1}] \leq 2^{\delta} \Pr[\mathsf{func}_{i,\Xi}(x_i) = x_{i+1}].$$

Proof. First consider $\beta = \varnothing$ and $i = j$. In the paths notation $\mathsf{cfunc}_{i:i}$ corresponds to func_i, so we have

$$\begin{aligned}
\Pr[\mathsf{func}_{i,\mathcal{K}}(x'_i) = x'_{i+1}] &= \sum_{\substack{x_j \in S_{0,j},\ j \in \{1 \dots d\} \setminus \{i,i+1\} \\ x_i = x'_i \\ x_{i+1} = x'_{i+1} \\ x_0 \in \Sigma \cup \{\sqcup\}}} \Pr[(x_0, x_1, \dots x_d) \in \text{paths}(\mathcal{K})] \\
&\leq \sum_{\dots} 2^{\delta} \Pr[(x_0, \dots x_d) \in \text{paths}(\Xi)] && \text{69 and 68} \\
&= 2^{\delta} \Pr[\mathsf{func}_{i,\Xi}(x'_i) = x'_{i+1}]
\end{aligned}$$

where the second sum is over the same variables as the first sum. For $\beta \neq \varnothing$, the same calculation goes through—some of the paths might be assigned zero probability (e.g. if they conflict with the values assigned by paths in β). Similarly for $j > i$. □

7.7 $\mathsf{BPP}^{\mathsf{QNC}_d}$ exclusion

The analysis would be very similar to the QNC_d case, once we use the sampling argument is invoked. We would construct shadows for $\mathcal{L}$ directly as before, except that certain paths β would be fixed The injective shuffler will show up at two places.

- We will state the probability of finding in terms of a distribution over injective shufflers.
- When we apply the sampling argument, we would only focus on how the distribution restricted to $\bar{S}_0$ changes, i.e. over injective shufflers.

How many times are the algorithms for generating $\bar{S}$ called?

- The base algorithm, for generating $\bar{S}_0$, is called once, at the very beginning of the analysis.
- The other algorithm, for generating $(\bar{S}_j)_{j \in \{1 \dots d\}}$, is called after each $\mathcal{C}_i$ is applied

7.7.1 Shadow oracles for $\mathrm{CQ_d}$ hardness

Here, we can state everything in terms of $\mathcal{L}$ and we simply need to add a condition for the event E happening which is meant to denote that Algorithm 49 succeeding.

Algorithm 72 (Procedure for generating S_{ij}, given β)**.** *Let $\mathcal{L}' = (H'_0 \dots H'_d)$ and Σ be as in Notation 43 and Definition 35. Let $S_i = H_{i-1}(\dots H_0(\Sigma)\dots)$ (as defined in Algorithm 49).*
Input:

1. *Base sets $\bar{S}_0$ (see Definition 60)*
2. *(valid) paths β wrt $\bar{S}_0$ (see Definition 63)*
3. *Whether or not event E happened.*

Output:
If E did not happen, set $S_{ik} = \varnothing$ for all $i, k \in \{1 \dots d\}$.
Otherwise, for each $i \in \{1 \dots d\}$, do the following:

1. *Define $S_{ik} := \varnothing$ for $1 \le k < i$.*
2. *Sample, uniformly at random, $S_{ii} \subseteq S_{i-1,i} \backslash X_i(\beta)$ such that $S_i \backslash X_i(\beta) \subseteq S_{ii}$ and $|S_{ii}|/|S_{i-1,i}| = 1/|\Sigma|$.*
3. *Define $S_{ik} := H'_{k-1}(\dots H'_i(S_{ii}) \dots)$ for $i < k \le d$.*

In both cases, return $\bar{S}_i := (S_{i1}, \dots S_{id})$ for each $i \in \{1 \dots d\}$.

7.7.2 Properties of the shadow oracles

We would need the analogue of Claim 52 and Claim 56 which in this case turns out to be the following. Note that the probability of interest can be computed by looking at the injective shuffler associated with the oracle.

Claim 73. Let $\mathcal{K} \sim \mathbb{F}_{\text{inj}}^{\delta|\beta}$ be an injective shuffler conditioned on β wrt base sets $\bar{S}_0$, sampled from a δ non-β-uniform distribution (see Definition 64 and Notation 69) where $|\beta| \le \text{poly}(n)$. Suppose $\mathcal{L}'$ is $\mathcal{L}$ conditioned on some variable τ such that $\mathcal{L}'$ restricted to $\bar{S}_0$ is exactly $\mathcal{K}$. Suppose Algorithm 72 is run with inputs $\bar{S}_0$, β and the assertion that E happened and let its output be S_{ij} for $i, j \in \{1 \dots d\}$. Then,

$$\Pr\left[x \in S_{ij} | \check{\mathcal{L}}'\right] \le (2^{\delta} + c) \cdot \text{poly}(n) \cdot \text{negl}(n)$$

where c is some constant (independent of δ, d etc.) $\check{\mathcal{L}}'$ is $\mathcal{L}'$ outside $\bar{S}_{i-1}$ (see Notation 43 with $S^{\text{out}} \leftarrow \bar{S}_{i-1}$ and $\mathcal{L}' \leftarrow \mathcal{L}'$) for all $1 \le i \le j \le d$ where the probability is over the randomness in $\mathcal{K}$ (i.e. from $\mathbb{F}_{\text{inj}}^{\delta|\beta}$) and the randomness in Algorithm 72.

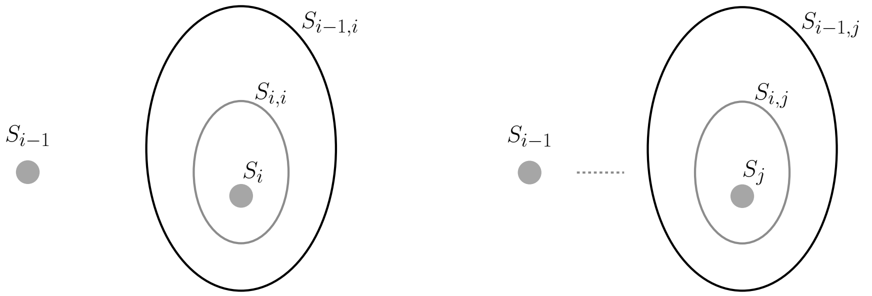
Proof sketch. Our strategy is to reduce the analysis to the case where the injective shuffler is uniformly distributed. We show this for $\beta = \varnothing$ (the $\beta \ne \varnothing$ case follows by reasoning as we did for the proof of Claim 56).

Consider the $k = i$ **case** (see Figure 8 left). Let S_i be as in Algorithm 72, i.e. $S_{i+1} = \mathsf{cfunc}_{0:i,\mathcal{K}}(\Sigma)$ using Notation 66 for $i \in \{0, \dots d\}$. We have

$$\Pr[x \in S_{ii} | \underbrace{S_{i-1}; \bar{S}_{i-1}}_{\mathcal{I}}] = \Pr[x \in S_{ii} | x \in S_i, \mathcal{I}] \Pr[x \in S_i | \mathcal{I}] + \Pr[x \in S_{ii} | x \notin S_i, \mathcal{I}] \Pr[x \notin S_i | \mathcal{I}]$$

$$\le \underbrace{\Pr[x \in S_i | \mathcal{I}]}_{\mathsf{I}} + \underbrace{\Pr[x \in S_{ii} | x \notin S_i, \mathcal{I}]}_{\mathsf{II}}$$

where observe that $\Pr[x \in S_{ii} | x \in S_i, \mathcal{I}] = 1$ because by construction, $S_i \subseteq S_{ii}$ and where we use the trivial bound $\Pr[x \notin S_i | \mathcal{I}] \le 1$ (which as we shall see is almost saturated).



(a) For analysing $\Pr(x \in S_{ii})$.

(b) For analysing $\Pr(x \in S_{ij})$ for $j > i$

Figure 8: Visual aid for analysing $\Pr[x \in S_{ij}]$.

Bounding Term I. The first term may be bounded as

$$\begin{aligned} \Pr[x \in S_i|\mathcal{I}] &\le \sum_{x' \in S_{i-1}} \Pr[\mathsf{func}_{i-1,\mathcal{K}}(x') = x] \\ &\le \sum_{x' \in S_{i-1}} 2^{\delta} \Pr[\mathsf{func}_{i-1,\Xi}(x') = x] \\ &= 2^{\delta} \frac{|S_{i-1}|}{|S_{i-1,i}|} \end{aligned} \tag{5}$$

where the first inequality is just a union bound, the second follows from Definition 61 where $\Xi \sim \mathbb{F}_{\mathrm{inj}}$, and the third is computed by proceeding as follows.

For $i > 1$, $\mathsf{func}_{i-1,\Xi}$ is just a uniformly random permutation from $S_{i-1,i-1} \to S_{i-1,i}$ (which have the same size) and we are asking for the probability that one of the elements is mapped as we like. This is readily computed to be

$$\Pr[\mathsf{func}_{i-1,\Xi}(x') = x] = \frac{(Z-1)!}{Z!} = \frac{1}{Z}$$

where $Z = |S_{i-1,i}|$. One can therefore bound Equation (5) as $2^{\delta} \sum_{x' \in S_{i-1}} \frac{1}{Z} = 2^{\delta}|S_{i-1}|/|S_{i-1,i}|$ as asserted.

For $i = 1$, the function $\mathsf{func}_{i-1,\Xi}$ is a uniformly random injective function from $\Sigma \to S_{i-1,0}$. The probability that one element maps as we like is given by

$$\Pr[\mathsf{func}_{i-1,\Xi}(x') = x] = \frac{{}_{Z-1}P_{|\Sigma|-1}}{{}_{Z}P_{|\Sigma|}} = \frac{1}{Z}$$

as we observed in Fact 46. Proceeding as before, one can again bound Equation (5) as asserted.

Bounding Term II. The second term may be bounded as

$$\Pr[x \in S_{ii}|x \notin S_i, \mathcal{I}] \le \frac{|S_{ii}| - |S_i|}{|S_{i-1,i}| - |S_i|}$$

where we use Remark 47 (second observation) with $M = |S_{i-1,i} \backslash S_i|$ and $N = |S_{ii} \backslash S_i|$. Note that the randomness used in this bound comes from that of Algorithm 72 while in the previous step (for "Bounding Term 1"), we had to use the fact that $\mathcal{K}$ is sampled from $\mathbb{F}^{\delta}_{\mathrm{inj}}$ and Definition 61.

We therefore get

$$\begin{aligned} \Pr[x \in S_{ii}|S_{i-1}; \bar{S}_{i-1}] &\le 2^{\delta} \frac{|S_{i-1}|}{|S_{i-1,i}|} + \frac{|S_{ii}| - |S_i|}{|S_{i-1,i}| - |S_i|} \\ &\le (2^{\delta} + c)\mathrm{negl}(n) \end{aligned} \tag{6}$$

where c is some constant (independent of δ etc; see Subsection B.1 for details).

We now consider the $j > i$ **case** (see Figure 8 right). We proceed analogously to the $i = j$ case and see that almost nothing changes. In particular, one has

$$\begin{aligned} \Pr[x \in S_{ij}|\underbrace{S_{i-1}; \bar{S}_{i-1}}_{\mathcal{I}}] &= \Pr[x \in S_{ij}|x \in S_j; \mathcal{I}] \Pr[x \in S_j|\mathcal{I}] + \Pr[x \in S_{ij}|x \notin S_j; \mathcal{I}] \Pr[x \notin S_j; \mathcal{I}] \\ &\le \underbrace{\Pr[x \in S_j|\mathcal{I}]}_{\mathsf{I}} + \underbrace{\Pr[x \in S_{ij}|x \in S_j, \mathcal{I}]}_{\mathsf{II}}. \end{aligned}$$

Bounding Term I. One can write

$$\begin{aligned} \Pr[x \in S_j|\mathcal{I}] &\le \sum_{x' \in S_{i-1}} \Pr[\mathsf{cfunc}_{i-1:j-1,\mathcal{K}}(x') = x] \\ &\le \sum_{x' \in S_{i-1}} 2^{\delta} \Pr[\mathsf{cfunc}_{i-1:j-1,\Xi}(x') = x] \\ &= 2^{\delta} \frac{|S_{i-1}|}{|S_{i-1,j}|} \end{aligned}$$

where the second inequality follows from Definition 61 where $\Xi \sim \mathbb{F}_{\mathrm{inj}}$, and the third is computed as in the $j = i$ case. More precisely, for the $i = 1$ sub-case (within this $j > i$ case), $\mathsf{cfunc}_{i-1:j-1}$ is a concatenation of

uniformly random injective functions, where the first goes from $\Sigma \to S_{i-1,1}$ and the subsequent ones from $S_{i-1,l} \to S_{i-1,l+1}$. This concatenation may be treated as a uniformly random injective function from $\Sigma \to S_{i-1,j}$ and one can then proceed as in the $i = 1$ sub-case (within the $j = i$ case). As for the $i > 1$ sub-case (within this $j > i$ case), $\mathsf{cfunc}_{i-1:j-1}$ is a concatenation of uniform permutations from $S_{i-1,l} \to S_{i-1,l+1}$ which may be viewed as a single uniform permutation from $S_{i-1,i-1} \to S_{i-1,j-1}$. Therefore, again, one can proceed as in the $i > 0$ sub-case (within the $j = i$ case).
Bounding Term II. One can write

$$\Pr[x \in S_{ij} | x \notin S_j; \mathcal{I}] \le \frac{|S_{ij}| - |S_j|}{|S_{i-1,j}| - |S_j|}$$

where we can use Remark 47 (second observation) with $M = |S_{i-1,j} \backslash S_j|$ and $N = |S_{ij} \backslash S_j|$. This is because the set S_{ii} was chosen uniformly at random (excluding the S_i part which we have anyway accounted for) and therefore $S_{ij} = \mathsf{cfunc}_{i,j-1,\mathcal{K}}(S_{ii})$ may also be viewed as set which is chosen uniformly at random (excluding the S_j part). This is because $\mathsf{cfunc}_{i,j-1,\mathcal{K}}$ is just a permutation (its distribution does not matter as long as S_{ii} is chosen uniformly at random[49]). Thereafter, one can proceed as in the $j = i$ case.

This yields the analogue of Equation (6), i.e.

$$\Pr[x \in S_{ij} | S_{i-1}; \bar{S}_{i-1}] \le (2^{\delta} + c) \cdot \text{negl}(n).$$

The result directly generalises for the $\beta \neq \varnothing$ case which at most adds a poly(n) factor in the final calculations with uniform distributions by changing the sizes of the exponential sized sets by a polynomial factor. □

7.7.3 d-CodeHashing is hard for CQ$_\mathrm{d}$

We now state the main lemma of this subsection.

Lemma 74 (d-$\mathsf{CodeHashing} \notin \mathsf{BPP}^{\mathsf{QNC}_d}$)**.** *Every* CQ$_\mathrm{d}$ *circuit succeeds at solving* d-$\mathsf{CodeHashing}$ *(see Definition 35) with probability at most* negl(λ) *on input* 1^λ *for* $d \le \text{poly}(n)$.

We begin with setting up the notation we use in the proof. It helps to recall that[50] $n = \Theta(\lambda)$.

- Denote by σ_0 the initial state (containing the input 1^λ and ancillae initialised to zero)
- From Notation 22, recall that CQ$_\mathrm{d}$ circuits can be represented as $\mathcal{C} = \mathcal{C}_{\tilde{n}} \circ \dots \mathcal{C}_2 \circ \mathcal{C}_1$ where $\tilde{n} \le \text{poly}(n)$. We write $\mathcal{C}_i := \vec{U}_i \circ \mathcal{A}_{c,i}$ where $\vec{U}_i$ denotes d layers of unitaries, followed by a measurement. For brevity, we drop the subscript "c" from $\mathcal{A}_{c,i}$ and even "$\circ$" to aid readibility.
- Let $\mathcal{L} = (H_0, \dots H_{d+1})$, d' and Σ be as in Definition 35.
- Denote by $\mathcal{C}^{\mathcal{L}} := \mathcal{A}^{\mathcal{L}}_{\tilde{n}+1} \vec{U}^{\mathcal{L}}_{\tilde{n}} \mathcal{A}^{\mathcal{L}}_{\tilde{n}} \dots \vec{U}^{\mathcal{L}}_1 \mathcal{A}^{\mathcal{L}}_1(\sigma_0)$, i.e. a CQ$_\mathrm{d}$ circuit with oracle access to $\mathcal{L}$.
  We make the following assumptions which only makes the result stronger (compare Figure 9a with Figure 9b; also see Example 75 below)
  - $\mathcal{A}_i$ ensures that its input is forwarded with its output
  - $\vec{U}_i$ forwards all classical information it receives as output
  - For $i > 1$, $\mathcal{A}_i$ receives an extra random variable (a set of paths, details appear later), correlated with $\mathcal{L}$ as input, labelled $\beta^*(s_{i-1})$.
  - both $\vec{U}_i$ and $\mathcal{A}_i$ (implicitly) receive the transcript (classical input/output messages) until they are invoked.
- In the analysis below, we consider $\tilde{n}$ sequences of shadow oracles. Each sequence is denoted by $\vec{\mathcal{M}}_i = (\mathcal{M}_{i,1}, \mathcal{M}_{i,2} \dots \mathcal{M}_{i,d})$, one set for each $\mathcal{C}_i$.
  - We use $\vec{U}_i^{\vec{\mathcal{M}}_i}$ to denote $\Pi_i \circ U_{i,d+1} \circ \mathcal{M}_{i,d} \circ U_{i,d} \circ \dots \mathcal{M}_{i,1} \circ U_{i,1}$.
  - $(\mathcal{M}_{i,j})_j$ are shadows of $\mathcal{L}$ using the sets outputted by Algorithm 72 (and are conditioned on Algorithm 49 succeeding). The input to the algorithm is described later.

[49] Just as for any $x, r \in \{0,1\}$, $r \oplus x$ is uniformly random if r is uniformly random.
[50] See the definition of suitable codes (see Lemma 32) and d-$\mathsf{CodeHashing}$ (see Definition 35).

- Denote by $\mathcal{C}^{\mathcal{M}} := \mathcal{A}^{\mathcal{L}}_{\tilde{n}+1}\vec{U}^{\vec{\mathcal{M}}_{\tilde{n}}}_{\tilde{n}}\mathcal{A}^{\mathcal{L}}_{\tilde{n}}\dots\vec{U}^{\vec{\mathcal{M}}_1}_1\mathcal{A}^{\mathcal{L}}_1(\sigma_0)$, i.e. a $\mathsf{CQ_d}$ circuit with access to only shadow oracles.
- After each circuit $\mathcal{C}_i$, the state is classical and this allows us to consider "transcripts" which we denote by T (the details appear later).
- Parameters for the sampling argument: Use $\delta = \Delta/\tilde{n}$, $\gamma = 2^{-m}$ where $\Delta > 0$ is an arbitrary, small constant and m is such that $m - \tilde{m} \geq \Omega(n)$ where $\tilde{m}$ is the length of the "advice", i.e. the number of bits $\mathcal{A}_i$ sends to $\vec{U}_i$.
- Shorthand for the $\Pr[\text{find} : \dots]$ notation: Suppose $\mathcal{L}$ is an oracle, $\bar{S}$ is a sequence of sets, ρ is a quantum state and T is some variable. We use $\Pr[\text{find} : U^{\mathcal{L}\backslash\bar{S}}, \rho|T]$ to denote the expression $\Pr[\text{find} : V^{\mathcal{N}\backslash\bar{R}}, \sigma]$ where $V = U|T$, $\mathcal{N} = \mathcal{L}|T$, $\sigma = \rho|T$.

Before we begin with the proof, we briefly illustrate how giving additional information to the classical algorithm (and conditioning at the same time) only strengthens our result.

Example 75. Suppose $\mathcal{O}$ is an oracle for Simon's Problem, encoding the period s. Let $\mathcal{A}$ denote an algorithm which takes no input and $\mathcal{B}$ denote an algorithm which takes an input S which is some variable correlated to $\mathcal{O}$. Then[51]

$$\begin{aligned}\max_{\mathcal{A}} \Pr[s \leftarrow \mathcal{A}^{\mathcal{O}}] &= \max_{\mathcal{A}} \sum_{S} \Pr[s \leftarrow \mathcal{A}^{\mathcal{O}|S}]\Pr[S] \\ &\leq \max_{\mathcal{B}} \sum_{S} \Pr[s \leftarrow \mathcal{B}^{\mathcal{O}|S}(S)]\Pr[S].\end{aligned}$$

Proof. For the overall template, we follow the proof of QNC_d hardness (see Lemma 53). Run algorithm 49 on $\mathcal{L}$ and let E be the event that it does not abort. Observe that

$$\left|\sum_{\mathrm{x}\in X_{\mathrm{valid}}} \Pr[\mathrm{x} \leftarrow \mathcal{C}^{\mathcal{L}}] - \sum_{\mathrm{x}\in X_{\mathrm{valid}}} \Pr[\mathrm{x} \leftarrow \mathcal{C}^{\mathcal{L}}|E]\right| \leq \mathrm{negl}(n) \tag{7}$$

as was the case before (recall X_{valid} was the set of valid solutions to d-$\mathsf{CodeHashing}$). We will show in *step one*, that $\mathcal{C}^{\mathcal{L}}|E$ and $\mathcal{C}^{\mathcal{M}}|E$ have essentially the same behaviour, i.e.

$$\left|\sum_{\mathrm{x}\in X_{\mathrm{valid}}} \Pr[\mathrm{x} \leftarrow \mathcal{C}^{\mathcal{L}}|E] - \sum_{\mathrm{x}\in X_{\mathrm{valid}}} \Pr[\mathrm{x} \leftarrow \mathcal{C}^{\mathcal{M}}|E]\right| \leq \mathrm{negl}(n) \tag{8}$$

and then in *step two*, that $\mathcal{C}^{\mathcal{M}}|E$ succeeds with at most negligible probability at solving d-$\mathsf{CodeHashing}$ (see Definition 35). These two steps, together with Equation (7), entail that $\mathcal{C}^{\mathcal{L}}$ solves d-$\mathsf{CodeHashing}$ with at most $\mathrm{negl}(n)$ probability.

In the rest of this proof, we *implicitly condition everything on the event* E and do not explicitly state this, for notational convenience. Let $\bar{S}_0$ be the output of Algorithm 49. Figure 9 may help in conveying the overarching idea.

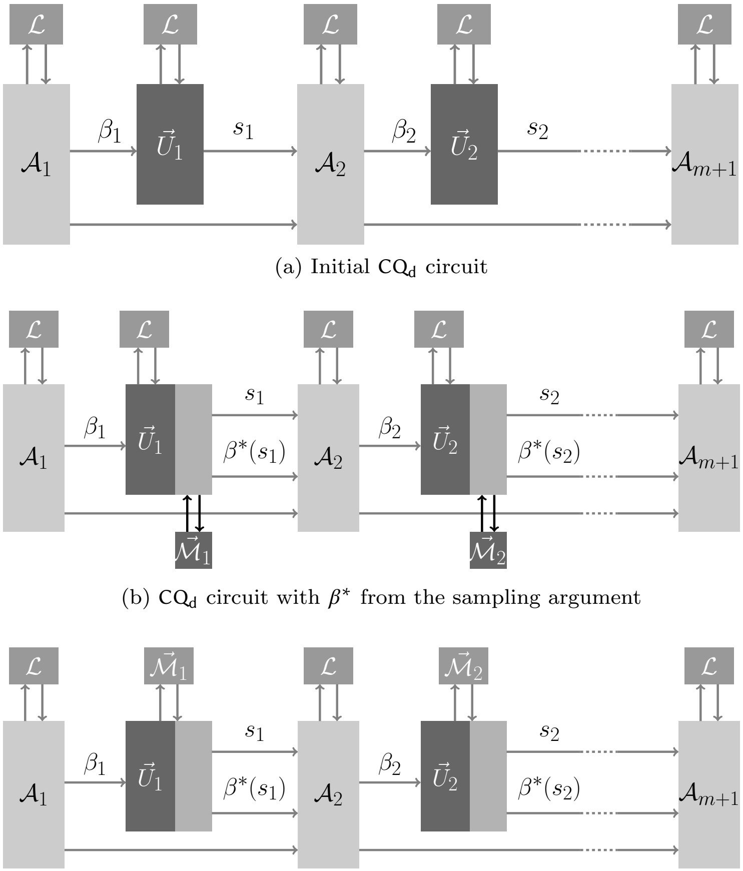
Step One. $\mathcal{C}^{\mathcal{L}}|E$ and $\mathcal{C}^{\mathcal{M}}|E$ have essentially the same behaviour,
Using a hybrid argument, one can bound the LHS of Equation (8) by bounding

$$\mathrm{TD}[\mathcal{C}^{\mathcal{L}},\mathcal{C}^{\mathcal{M}}] = \mathrm{TD}\left[\mathcal{A}^{\mathcal{L}}_{\tilde{n}+1}\vec{U}^{\mathcal{L}}_{\tilde{n}}\mathcal{A}^{\mathcal{L}}_{\tilde{n}}\dots\vec{U}^{\mathcal{L}}_1\mathcal{A}^{\mathcal{L}}_1(\sigma_0), \quad \mathcal{A}^{\mathcal{L}}_{\tilde{n}+1}\vec{U}^{\vec{\mathcal{M}}_{\tilde{n}}}_{\tilde{n}}\mathcal{A}^{\mathcal{L}}_{\tilde{n}}\dots\vec{U}^{\vec{\mathcal{M}}_1}_1\mathcal{A}^{\mathcal{L}}_1(\sigma_0)\right]$$

with

$$\begin{aligned}&\leq \sum_{i=1}^{\tilde{n}} \mathrm{TD}\Big[\mathcal{A}^{\mathcal{L}}_{\tilde{n}+1}\vec{U}^{\mathcal{L}}_{\tilde{n}}\mathcal{A}^{\mathcal{L}}_{\tilde{n}}\dots\vec{U}^{\mathcal{L}}_{i+1}\mathcal{A}^{\mathcal{L}}_{i+1} \quad \vec{U}^{\mathcal{L}}_i\mathcal{A}^{\mathcal{L}}_i\dots\vec{U}^{\mathcal{L}}_1\mathcal{A}^{\mathcal{L}}_1(\sigma_0), \\ &\qquad\qquad \mathcal{A}^{\mathcal{L}}_{\tilde{n}+1}\vec{U}^{\mathcal{L}}_{\tilde{n}}\mathcal{A}^{\mathcal{L}}_{\tilde{n}}\dots\vec{U}^{\mathcal{L}}_{i+1}\mathcal{A}^{\mathcal{L}}_{i+1} \quad \vec{U}^{\vec{\mathcal{M}}_i}_i\mathcal{A}^{\mathcal{L}}_i\dots\vec{U}^{\vec{\mathcal{M}}_1}_1\mathcal{A}^{\mathcal{L}}_1(\sigma_0)\Big] \\ &\leq \sum_{i=1}^{\tilde{n}} \mathrm{TD}\left[\vec{U}^{\mathcal{L}}_i\mathcal{A}^{\mathcal{L}}_i\vec{U}^{\vec{\mathcal{M}}_{i-1}}_{i-1}\dots\vec{U}^{\vec{\mathcal{M}}_1}_1\mathcal{A}^{\mathcal{L}}_1(\sigma_0), \quad \vec{U}^{\vec{\mathcal{M}}_i}_i\mathcal{A}^{\mathcal{L}}_i\vec{U}^{\vec{\mathcal{M}}_{i-1}}_{i-1}\mathcal{A}^{\mathcal{L}}_{i-1}\dots\vec{U}^{\vec{\mathcal{M}}_1}_1\mathcal{A}^{\mathcal{L}}_1(\sigma_0)\right].\end{aligned} \tag{9}$$

[51] For concreteness, if $\mathcal{A}$ and $\mathcal{B}$ are classical algorithms, and S is the period encoded in $\mathcal{O}$, then clearly the upper bound becomes 1 but it is not achievable; illustrating that this procedure can only strengthen the hardness result.



(a) Initial $\mathsf{CQ_d}$ circuit

(b) $\mathsf{CQ_d}$ circuit with β^* from the sampling argument

(c) $\mathsf{CQ_d}$ circuit with β^* where all oracles replaced with shadow oracles.

Figure 9: Variants of the $\mathsf{CQ_d}$ circuit which arise in establishing hardness of solving d-$\mathsf{CodeHashing}$. Observe that Figure 9b can simulate Figure 9a. We analyse the latter and show its behaviour is essentially the same as that of Figure 9c.

The $i = 1$ case

Begin with $i = 1$. Let $\mathcal{A}_1^{\mathcal{L}}(\sigma_0) =: \sigma_1$. One can write

$$\begin{aligned}
\mathrm{TD}[\vec{U}_1^{\mathcal{L}}(\sigma_1), \vec{U}_1^{\vec{\mathcal{M}}_1}(\sigma_1)] &= \mathrm{TD}[\mathcal{L}U_{1,d}\dots\mathcal{L}U_{1,1}(\sigma_1), \quad \mathcal{M}_{1,d}U_{1,d}\dots\mathcal{M}_{1,1}U_{1,1}(\sigma_1)] \\
&\le \sum_{j=1}^{d} \mathrm{TD}[\mathcal{L}U_{1,j} \quad \underbrace{\mathcal{M}_{1,j-1}U_{1,j-1}\dots\mathcal{M}_{1,1}U_{1,1}(\sigma_1)}_{:=\rho_{1,j-1}}, \qquad \text{hybrid argument} \\
&\qquad \mathcal{M}_{1,j}U_{1,j} \quad \overbrace{\mathcal{M}_{1,j-1}U_{1,j-1}\dots\mathcal{M}_{1,1}U_{1,1}(\sigma_1)}].
\end{aligned}$$

Our goal is to bound each term in the sum using the O2H lemma (Lemma 41) as

$$B(\mathcal{L}U_{1,j}\rho_{1,j-1}, \mathcal{M}_{1,j}U_{1,j}\rho_{1,j-1}) \le \sqrt{\Pr[\text{find} : U_{1,j}^{\mathcal{L}\setminus\bar{S}_{1,j}}, \rho_{1,j-1}]} \tag{10}$$

where $\mathcal{M}_{1,j}$ is the shadow of $\mathcal{L}$ wrt $\bar{S}_{1,j}$ and $\bar{S}_{1,j}$ is defined as follows.

Denote by β_1 the set of all paths queried $\mathcal{A}_1^{\mathcal{L}}$. Denote by $\beta_1' \subseteq \beta_1$ the subset of paths queried by $\mathcal{A}_1^{\mathcal{L}}$ wrt $\bar{S}_0$ (see Definition 63), i.e. let β_1' denote the set of path queries made by the first classical part of the $\mathsf{CQ_d}$ circuit within the base sets $\bar{S}_0$. Run Algorithm 72 with $\beta \leftarrow \beta_1'$, $\bar{S}_0 \leftarrow \bar{S}_0$ as inputs and define $\bar{S}_{1,j} \leftarrow \bar{S}_j$ where $\bar{S}_j$ is the output of the algorithm for $j \in \{1\dots d\}$. Let $\hat{\mathcal{L}}_{1,j}$ (resp. $\check{\mathcal{L}}_{1,j}$) be $\mathcal{L}$ inside (resp. outside) $\bar{S}_{1,j}$ (see Notation 43).

To apply Corollary 44 we condition the RHS of Equation (10) on $\check{\mathcal{L}}_{1,j}$ to write $\Pr[\text{find} : U_{1,j}^{\mathcal{L}\setminus\bar{S}_{1,j}}, \rho_{1,j-1}|\check{\mathcal{L}}_{1,j-1}]$. The conditioning ensures that $\rho_{1,j-1}|\check{\mathcal{L}}_{1,j-1}$ is uncorrelated[52] with $\bar{S}_{1,j}|\check{\mathcal{L}}_{1,j-1}$. Using Claim 73, with $\delta \leftarrow 0$, $\beta \leftarrow \beta_1'$ and $\bar{S}_0 \leftarrow \bar{S}_0$, one can apply Corollary 44 to obtain

$$\Pr[\text{find} : U_{1,j}^{\mathcal{L}\setminus\bar{S}_{1,j}}, \rho_{1,j-1}|\check{\mathcal{L}}_{1,j-1}] \le \mathrm{negl}(n)$$

which in turn bounds Equation (10) by $\mathrm{negl}(n)$. □

Before moving to the $i = 2$ and then the general case, we describe an intuitive picture to keep in mind. Observe that the shadow oracles $\{\mathcal{M}_{1,j}\}_j$ were determined, in particular, by the set of paths β_1 queried by the classical algorithm.

In step 2, the shadow oracles $\{\mathcal{M}_{2,j}\}_j$ would be determined by (in addition to β_1) both, the set of paths β_2 queried by the classical algorithm $\mathcal{A}_2^{\mathcal{L}}$ and by the set of paths $\beta(s_1)$ the classical algorithm $\mathcal{A}_2^{\mathcal{L}}$ receives as an extra input.[53] We have not yet defined how the paths $\beta(s_1)$ are specified. They are specified by the sampling argument (Proposition 70). As illustrated in Figure 9, treat the sampling argument as an algorithm which interacts with $\mathcal{M}_{1,d}$ and produces $\beta^*(s_1)$ as output (the star indicates that the last coordinate of some of the paths may be $\perp$; in $\beta(s_1)$ all coordinates are non-$\perp$ as $\mathcal{A}_2$ has access to $\mathcal{L}$). Using the notation in Proposition 70, it outputs the p-many paths (as $\beta^*(s_1)$), which are present in $\mathcal{G}_1$ with probability α_k, for each k.

In step i, proceeding analogously, the shadow oracles $\{\mathcal{M}_{i,j}\}_j$ would be determined by $\beta_1 \cup \beta_2 \dots \cup \beta_i$ and by $\beta(s_1) \cup \dots \cup \beta(s_{i-1})$, i.e. by the "transcript" encoding the paths exposed so far. We would then condition on these paths and use the fact that after conditioning, these distributions stay $(ip, i\delta)$ uniform, which in turn allows us to argue that the analogue of Equation (10) (i.e. $\Pr[\text{find} : U_{i,j}^{\mathcal{L}\setminus\bar{S}_{i,j}}, \rho_{i,j-1}]$ below) stays negligible.

To apply the sampling argument, it would be useful to restrict to the distribution over the base sets which is facilitated by the following notation.

Notation 76 ($\mathsf{inj}[\mathcal{L}'|\beta]$ wrt to $\bar{S}_0$). Suppose $\mathcal{L}' = (H_0', \dots H_d')$ (as in Notation 43) is a random variable sampled from some arbitrary distribution. Let β be a set of paths in $\mathcal{L}'$. Then $\mathsf{inj}[\mathcal{L}'|\beta] =: \Xi$ wrt $\bar{S}_0$ denotes a

[52] This is because, given $\check{\mathcal{L}}_{1,j-1}$ (which, in particular, specifies $\bar{S}_{1,j-1}$ but not the values of $\mathcal{L}$ inside $\bar{S}_{1,j-1}$), $\bar{S}_{1,j}|\check{\mathcal{L}}_{1,j-1}$ is, by construction, a (component-wise) subset of $\bar{S}_{1,j-1}$ but within $\bar{S}_{1,j-1}$, the distribution of $\bar{S}_{1,j}|\check{\mathcal{L}}_{1,j-1}$ is determined by the randomness in Algorithm 72 and by the distribution of $\hat{\mathcal{L}}_{1,j-1}$. The randomness of the algorithm Algorithm 72 is independent of $\mathcal{L}$ and $\rho_{1,j-1}$ contains at most as much information about $\mathcal{L}$ as is present in $\check{\mathcal{L}}_{1,j-1}$ (that is because $\rho_{1,j-1}$ only has access to $\mathcal{M}_{1,1}\dots\mathcal{M}_{1,j-1}$ which block all information about $\mathcal{L}$ inside $\bar{S}_{j-1}$).

[53] Strictly, (as explained later) it receives $\beta^*(s_1)$ which may have some paths with $\perp$ as the last coordinate but since $\mathcal{A}_2^{\mathcal{L}}$ can access $\mathcal{L}$, we assume it learns the last coordinates of all paths in $\beta^*(s_1)$ and denote the complete paths as $\beta(s_1)$.

(random) injective shuffler conditioned on β wrt base sets $\bar{S}_0$ (see Definition 64) such that for all $x_0 \in \Sigma$, $x_1 \in S_{01}, \dots x_d \in S_{0d}, x_{d+1} \in \{0,1\}^n \cup \{\perp\}$

$$\Pr[(x_0, \dots x_{d+1}) \in \mathrm{paths}(\Xi)] = \begin{cases} \Pr[H_0(x_0) = x_1 \wedge \dots H_{d-1}(x_{d-1}) = x_d] & \text{for } x_0 \in \Sigma \backslash X_0(\beta) \\ \Pr[H_0(x_0) = x_1 \wedge \dots H_d(x_d) = x_{d+1}] & \text{otherwise.} \end{cases}$$

We now resume with the proof.

Proof (cont.) **The $i = 2$ case**
Let $\mathcal{A}_2^{\mathcal{L}} \vec{U}_1^{\vec{\mathcal{M}}_1} \mathcal{A}_1^{\mathcal{L}}(\sigma_0) =: \sigma_2$. One can write the $i = 2$ term in the RHS of Equation (9) as

$$\mathrm{TD}[\vec{U}_2^{\mathcal{L}}(\sigma_2), \vec{U}_2^{\vec{\mathcal{M}}_2}(\sigma_2)] \le \sum_{j=1}^{d} \mathrm{TD}[\mathcal{L}U_{2,j}\rho_{2,j-1}, \quad \mathcal{M}_{2,j}U_{2,j}\rho_{2,j-1}]$$

where $\rho_{2,j-1} := \mathcal{M}_{2,j-1}U_{2,j-1} \dots \mathcal{M}_{2,1}U_{2,1}(\sigma_2)$. Using Lemma 41, one can write

$$B[\mathcal{L}U_{2,j}\rho_{2,j-1}, \quad \mathcal{M}_{2,j}U_{2,j}\rho_{2,j-1}] \le \sqrt{\Pr[\mathrm{find} : U_{2,j}^{\mathcal{L}\backslash \bar{S}_{2,j}}, \rho_{2,j-1}]} \tag{11}$$

where $\mathcal{M}_{2,j}$ is the shadow of $\mathcal{L}$ wrt $\bar{S}_{2,j}$ and $\bar{S}_{2,j}$ is defined as follows.

Recall that β_1 denoted the paths queried by $\mathcal{A}_1^{\mathcal{L}}$. Note that $\mathsf{inj}[\mathcal{L}|\beta_1]$ (see Notation 76) is distributed as $\mathbb{F}_{\mathrm{inj}}^{|\beta_1}$. Let the output of $\vec{U}_1^{\vec{\mathcal{M}}_1}$ be[54] s_1. Given that $\Pr[s_1|\beta_1] \ge \gamma$, $\mathsf{inj}[\mathcal{L}|\beta_1 s_1]$ which is distributed as $\mathbb{F}_{\mathrm{inj}}^{|\beta_1}$ may be expressed as a convex combination (as described in Proposition 70) $\mathsf{inj}[\mathcal{L}|s_1\beta_1\beta^*(s_1)]$ distributed as $\mathbb{F}_{\mathrm{inj}}^{(p,\delta)|\beta_1}$ where $|\beta^*(s_1)| \le p \le 2m/\delta$ whenever the convex coefficient is larger than γ. When $\Pr[s_1|\beta_1] < \gamma$, let $\beta^*(s_1) = \varnothing$. Note that this implicitly defines the random variable $\beta^*(s_1)$ which we had initially left unspecified. $\mathcal{A}_2^{\mathcal{L}}$ takes as input s_1 and $\beta^*(s_1)$. $\mathcal{A}_2^{\mathcal{L}}$ learns $\beta(s_1)$ which is $\beta^*(s_1)$ with $\perp$s replaced by the value $\mathcal{L}$ takes in the last coordinate. Let β_2 denote the addition paths queried by $\mathcal{A}_2^{\mathcal{L}}$.

We are now ready to define $\bar{S}_{2,j}$. Let $\beta_2' \subseteq \beta_2$ be the subset of paths in β_2 which are within the base sets $\bar{S}_0$. Run Algorithm 72 with $\beta \leftarrow \beta_2' \cup \beta(s_1) \cup \beta_1'$, $\bar{S}_0 \leftarrow \bar{S}_0$ as inputs and define $\bar{S}_{2,j} \leftarrow \bar{S}_j$ where $\bar{S}_j$ is the output of the algorithm, for $j \in \{1, \dots d\}$. Let $\hat{\mathcal{L}}_{2,j}$ (resp. $\check{\mathcal{L}}_{2,j}$) be $\mathcal{L}$ inside (resp. outside) $\bar{S}_{2,j}$ (see Notation 43).

To apply Corollary 44 we condition the RHS of Equation (11) on $\check{\mathcal{L}}_{2,j-1}$ to write $\Pr[\mathrm{find} : U_{2,j}^{\mathcal{L}\backslash \bar{S}_{2,j}}, \rho_{2,j-1}|\check{\mathcal{L}}_{2,j-1}]$. The conditioning, as before, ensures that $\rho_{2,j-1}|\check{\mathcal{L}}_{2,j-1}$ is uncorrelated with $\bar{S}_{2,j}|\check{\mathcal{L}}_{2,j-1}$ (for exactly the same reason as the $i = 1$ case). However, to apply Claim 73 we condition on the "transcript" until the output of $\mathcal{A}_2^{\mathcal{L}}$, i.e. $T(\sigma_2) := (\beta_1, s_1, \beta(s_1), \beta_2)$, by writing $\Pr[\mathrm{find} : U_{2,j}^{\mathcal{L}\backslash \bar{S}_{2,j}}, \rho_{2,j-1}|\check{\mathcal{L}}_{2,j-1}]$

$$\begin{aligned}
&= \sum_{s_1, \beta_1, \beta(s_1), \beta_2} \Pr[T(\sigma_2)] \cdot \Pr[\mathrm{find} : U_{2,j}^{\mathcal{L}\backslash \bar{S}_{2,j}}, \rho_{2,j-1}|\check{\mathcal{L}}_{2,j-1} \; T(\sigma_2)] \\
&\le \sum_{\substack{s_1 : \Pr[s_1|\beta_1] \ge 2^{-m} \\ \beta_1, \beta(s_1), \beta_2}} \underbrace{\Pr[\beta_2|s_1\beta_1\beta(s_1)] \Pr[\beta(s_1)|s_1\beta_1] \Pr[s_1|\beta_1] \Pr[\beta_1]}_{\Pr[T(\sigma_2)]} \cdot \Pr[\mathrm{find} : U_{2,j}^{\mathcal{L}\backslash \bar{S}_{2,j}}, \rho_{2,j-1}|\check{\mathcal{L}}_{2,j-1} \; T(\sigma_2)] + 2^{-(m-\tilde{m})} \\
&\le \sum_{\substack{s_1 : \Pr[s_1|\beta_1] \ge 2^{-m} \\ \beta(s_1) : \Pr[\beta(s_1)|s_1\beta_1] \ge 2^{-m} \\ \beta_1, \beta_2}} \Pr[\beta_2|s_1\beta_1\beta(s_1)] \Pr[\beta(s_1)|s_1\beta_1] \Pr[s_1|\beta_1] \Pr[\beta_1] \cdot \underbrace{\Pr[\mathrm{find} : U_{2,j}^{\mathcal{L}\backslash \bar{S}_{2,j}}, \rho_{2,j-1}|\check{\mathcal{L}}_{2,j-1} \; T(\sigma_2)]}_{\mathsf{Term\ I}} + 2 \cdot 2^{-(m-\tilde{m})} \\
&\le \mathrm{negl}(n)
\end{aligned}$$

where to obtain the first inequality, we note that for each $s_1 : \Pr[s_1|\beta_1] \ge 2^{-m}$, one can use Proposition 70 and one can account for all $s_1 : \Pr[s_1|\beta_1] < 2^{-m}$, by simply upper bounding the sum by $2^{-(m-\tilde{m})}$ because s_1 is of length $\tilde{m}$. In the second inequality, we use the fact that either the convex weight (i.e. $\Pr[\beta(s_1)|s_1\beta_1]$) as specified in Proposition 70 is less than 2^{-m} (for at most each s_1, therefore it contributes at most $2^{-(m-\tilde{m})}$ to the sum) or it is greater than 2^{-m}. In the latter case, the injective shuffler is $\mathbb{F}_{\mathrm{inj}}^{(p,\delta)|\beta}$ distributed and therefore one can apply Corollary 44 together with Claim 73 with $\delta \leftarrow \delta$, $\beta \leftarrow \beta_2' \cup \beta(s_1) \cup \beta_1'$ and $\bar{S}_0 \leftarrow \bar{S}_0$ to obtain $\mathsf{Term\ I} \le 2^{\delta} \cdot \mathrm{poly}(n) \cdot \mathrm{negl}(n)$.

[54] which in particular, contains β_1

The general $i \in \{1 \dots \tilde{n}\}$ case.
This is a straightforward generalisation of the $i = 2$ case and hence we only outline the key steps. Let $\sigma_i := \mathcal{A}_i^{\mathcal{L}} \vec{U}_{i-1}^{\vec{\mathcal{M}}_{i-1}} \dots \mathcal{A}_2^{\mathcal{L}} \vec{U}_1^{\vec{\mathcal{M}}_1} \mathcal{A}_1^{\mathcal{L}}(\sigma_0)$ where $\mathcal{M}_{i-1,j}$ is the shadow of $\mathcal{L}$ wrt $\bar{S}_{i-1,j}$, let $T(\sigma_i) := (\beta_1, s_1, \beta(s_1), \dots \beta_{i-1}, s_{i-1}, \beta(s_{i-1}), \beta_i)$ where β_i denotes the paths queried by $\mathcal{A}_i^{\mathcal{L}}$, s_{i-1} denotes the output of $\vec{U}_{i-1}^{\vec{\mathcal{M}}_{i-1}}$, $\beta^*(s_{i-1})$ be the paths as in Proposition 70 when $\Pr[s_{i-1} | \beta_{i-1} \dots \beta(s_1) s_1 \beta_1] \geq \gamma$, $\mathsf{inj}[\mathcal{L} | \beta_1 s_1 \beta(s_1) \dots \beta_{i-1}, s_{i-1}]$ is distributed as $\mathbb{F}_{\text{inj}}^{(i-2)\delta | \beta_{i-1} \cup \beta(s_{i-2}) \cup \beta_{i-2} \dots \cup \beta_1}$ so that[55] $\mathsf{inj}[\mathcal{L} | \beta_1 s_1 \beta(s_1) \dots \beta_{i-1} s_{i-1} \beta^*(s_{i-1})]$ is distributed as $\mathbb{F}_{\text{inj}}^{(p,(i-2)\delta) | \beta_{i-1} \cup \dots \cup \beta_1}$ whenever the convex coefficient is larger than $\gamma = 2^{-m}$. $\beta(s_{i-1})$ is $\beta^*(s_{i-1})$ with $\perp$s replaced be the values taken by $\mathcal{L}$ at those coordinates. Let $\beta_i' \subseteq \beta_i$ be the subset of paths in $\bar{S}_0$. Run Algorithm 72 with $\beta \leftarrow \beta_i' \cup \beta(s_{i-1}) \cup \dots \cup \beta_1'$, $\bar{S}_0 \leftarrow \bar{S}_0$ as inputs and define $\bar{S}_{i,j}$ to be $\bar{S}_j$ which is the output of the algorithm for $j \in \{1 \dots d\}$. Let $\hat{\mathcal{L}}_{i,j}$ (resp. $\check{\mathcal{L}}_{i,j}$) be $\mathcal{L}$ inside (resp. outside) $\bar{S}_{i,j}$ (see Notation 43). Let $\mathcal{M}_{i,j}$ be the shadow of $\mathcal{L}$ wrt $\bar{S}_{i,j}$. The ith term in Equation (9) can then be expressed as

$$\mathrm{TD}[\vec{U}_i^{\mathcal{L}}(\sigma_i), \vec{U}_i^{\vec{\mathcal{M}}_i}(\sigma_i)] \leq \sum_{j=1}^{d} \mathrm{TD}[\mathcal{L} U_{i,j} \rho_{i,j-1}, \quad \mathcal{M}_{i,j} U_{i,j} \rho_{i,j-1}]$$

where $\rho_{i,j-1} := \mathcal{M}_{i,j-1} U_{i,j-1} \dots \mathcal{M}_{i,1} U_{i,1}(\sigma_i)$. The square of the jth term, can then be bounded (using Lemma 41) by $\Pr[\mathrm{find} : U_{i,j}^{\mathcal{L} \setminus \bar{S}_{i,j}}, \rho_{i,j-1}]$ which is

$$\leq \sum_{\substack{s_1 : \Pr[s_1 | \beta_1] \geq 2^{-m}, \dots s_{i-1} : \Pr[s_{i-1} | \beta_1 \dots] \geq 2^{-m} \\ \beta(s_1) : \Pr[\beta(s_1) | s_1 \beta_1] \geq 2^{-m}, \dots \beta(s_{i-1}) : \Pr[\beta(s_{i-1}) | s_1 \beta_1 \dots] \geq 2^{-m} \\ \beta_1, \beta_2 \dots \beta_i}} \alpha(T(\sigma_i)) \cdot \underbrace{\Pr[\mathrm{find} : U_{i,j}^{\mathcal{L} \setminus \bar{S}_{i,j}}, \rho_{i,j-1} | \check{\mathcal{L}}_{i,j-1} \ T(\sigma_i)]}_{\textsf{Term I}} + 2 \cdot (i-1) \cdot 2^{-(m - \tilde{m})}$$

$$\leq 2^{\Delta} \cdot \mathrm{poly}(n) \cdot \mathrm{negl}(n) + 2 \cdot (i-1) \cdot 2^{-(m-\tilde{m})} \leq \mathrm{negl}(n)$$

where α is the probability coefficient (bounded by 1), and the distribution of the injective shuffler in Term I is $\mathbb{F}^{i \cdot \delta | \beta_i' \cup \beta(s_{i-1}) \cup \dots \cup \beta_1'}$. This is obtained by repeatedly applying Proposition 70 (for the kth application, $\delta' \leftarrow (k-1)\delta$, $\beta \leftarrow \beta_k' \cup \beta(s_{k-1}) \dots \cup \beta_1'$ and $\bar{S}_0 \leftarrow \bar{S}_0$) and arguing as before to collect terms for which the distribution over the injective shuffler is unknown (but which occur with probability at most 2^{-m}). Independence of $\bar{S}_{i,j}$ from $\rho_{i,j-1}$ can be argued as before once it is conditioned on $\check{\mathcal{L}}_{i,j-1}$ and one can apply Corollary 44 together with Claim 73 (with $\delta \leftarrow i \cdot \delta$, $\beta \leftarrow \beta_i' \cup \beta(s_{i-1}) \dots \cup \beta_1'$ and $\bar{S}_0 \leftarrow \bar{S}_0$) to obtain the stated bound on Term I (recall $\gamma = 2^{-m}$ and $\delta = \Delta / \tilde{n}$).

Step Two. $\mathcal{C}^{\mathcal{M}} | E$ succeeds at solving d-CodeHashing with at most negligible probability. This is analogous to how we argued in the proof of Lemma 57. The quantum part never has any information about $\tilde{H}$ (recall $\tilde{H}(\cdot) = H_d \circ \dots H_0(\cdot)$) which the classical algorithm before it does not already have. Therefore the success probability of $\mathcal{C}^{\mathcal{M}} | E$ is limited by the number of classical queries it makes. Since this is polynomial, from Theorem 34 (second part), it follows that $\mathcal{C}^{\mathcal{M}} | E$ succeeds with negligible probability.

□

7.8 $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$ exclusion

The proof of CQC_d hardness of d-CodeHashing is, *conceptually*, a straightforward combination of $\mathsf{QC_d}$ hardness and of $\mathsf{CQ_d}$ hardness. In the proof of $\mathsf{CQ_d}$ hardness, we analysed each QNC_d circuit by following the ideas behind the QNC_d hardness proof. The difference was that instead of using the random oracles in $\mathcal{L}$ directly (see Definition 35), we used the conditioned oracle $\mathcal{L}|s$ and then relied on the behaviour of the injective shuffler $\mathsf{inj}[\mathcal{L}|s]$ to argue indistinguishability from the appropriate shadow oracles.

We now proceed almost exactly as in the $\mathsf{CQ_d}$ hardness case and analyse each $\mathsf{QC_d}$ circuit by following the ideas behind the $\mathsf{QC_d}$ hardness proof. As before, the difference would be that we would use properties of $\mathsf{inj}[\mathcal{L}|s]$ (see Notation 76) instead of $\mathcal{L}$. Recall that in the analysis of $\mathsf{QC_d}$ we had to introduce the notion of "query paths" (see Definition 54). However, we already introduced "paths" more carefully for analysing $\mathsf{CQ_d}$ hardness and this makes it easier to analyse CQC_d hardness (see Definition 63). Compared to $\mathsf{CQ_d}$, at a high level, the difference would just be that we expose additional paths β after each layer of unitaries. This slightly changes the way shadows are defined, i.e. we need to adapt Algorithm 72 to our setting (currently it is closer to the QNC_d case, Algorithm 50, and we want it to be more like the $\mathsf{QC_d}$ case, Algorithm 55). However, one can still use Claim 56 which was used to argue that shadow oracles are hard to distinguish from the originals as this already accounts for paths β exposed.

[55] Note that $i \geq 2$ when this reasoning is applied because s_{i-1} is s_1 for $i = 2$.

7.8.1 Shadow oracles for CQC_d hardness and their properties

The procedure for generating the sets $S_{i,j}$ in this case, is essentially the same as Algorithm 72 with only one difference: the procedure is applied to each index $i \in \{1\dots d\}$ because the paths exposed by the classical algorithm are only determined after each layer of unitary is applied. To contrast, in the $\mathsf{CQ_d}$ case, these paths (queried by the classical algorithm) were determined before the quantum part of the circuit was executed and one could therefore construct the all $\{S_{i,j}\}_{i,j}$ at once.

Algorithm 77 (Procedure for generating $S_{i,j}$, given β, and the previous sets $\bar{S}_{i-1}$)**.** *Let $\mathcal{L}' = (H'_0, \dots H'_d)$ be $\mathcal{L}$ conditioned on some variable, as in Notation 43, Σ be as in Definition 35 and $S_i = H'_{i-1}(\dots H'_0(\Sigma)\dots)$ be as in Algorithm 49.*
Input:

1. *Index: $i \in \{1\dots d\}$*
2. *Base sets $\bar{S}_0 = (S_{0,j})_{j\in\{1\dots d\}}$ (see Definition 60)*
3. *The set of paths queried: (valid) paths β wrt $\bar{S}_0$ (see Definition 63)*
4. *The previous sequence of sets for creating the shadow oracle: If $i > 0$, then $\bar{S}_{i-1} := (S_{i-1,j})_{j\in\{1\dots d\}}$ where $S_{i-1,j} \subseteq S_{0,j}$ for all $j \in \{1\dots d\}$.*
5. *Whether or not event E happened.*

Output:
If E did not happen, set $S_{ik} = \varnothing$ for all $i, k \in \{1, \dots d\}$. Otherwise, for each $i \in \{1\dots d\}$ do the following.

1. *Define $S_{ik} = \varnothing$ for $1 \le k < i$.*
2. *Sample, uniformly at random, $S_{ii} \subseteq S_{i-1,i} \backslash X_i(\beta)$ such $S_i \backslash X_i(\beta) \subseteq S_{ii}$, and $|S_{ii}|/|S_{i-1,i}| = 1/|\Sigma|$*
3. *Define $S_{ik} := H'_{k-1}(\dots H'_i(S_{ii})\dots)$ for $i < k \le d$.*

In both cases, return $\bar{S}_i := (S_{i1}, \dots S_{id})$.

The key property satisfied by Algorithm 72 was Claim 73. The analogous property for Algorithm 77 is the following which is almost identical to Claim 73 except that one specifies some conditions to ensure $\bar{S}_{i-1}$ is appropriately defined. When we apply the algorithm, as in the $\mathsf{QC_d}$ case, we would begin with $\bar{S}_0$ and successively apply Algorithm 77 to produce $\bar{S}_1, \dots \bar{S}_d$ and the stated conditions would automatically hold.

Claim 78. Let $\mathcal{K} \sim \mathbb{F}^{\delta|\beta}_{\text{inj}}$ be an injective shuffler conditioned on β wrt base sets $\bar{S}_0$, sampled form a δ non-β-uniform distribution (see Definition 64 and Notation 69) where $|\beta| \le \text{poly}(n)$. Suppose $\mathcal{L}'$ is $\mathcal{L}$ conditioned on some variable τ such that $\mathsf{inj}[\mathcal{L}']$ wrt $\bar{S}_0$ (see Notation 76) is exactly $\mathcal{K}$. Suppose Algorithm 77 is run with the following inputs: an index $i \in \{1\dots d\}$, the base sets $\bar{S}_0$, valid paths β, a sequence of sets $\bar{S}_{i-1}$ (defined next) and the assertion that E happened and let its output be S_{ij} for $j \in \{1, \dots d\}$. If $i > 1$, $\bar{S}_{i-1} := (S_{i-1,1}, S_{i-1,2}, \dots S_{i-1,d})$ are arbitrary sets such that

- for $j < i-1$, $S_{i-1,j} = \varnothing$
- for $j = i-1$, $S_{i-1,i-1} \subseteq S_{0,i-1}$, $S_{i-1} \backslash X_{i-1}(\beta) \subseteq S_{i-1,i-1}$ (where S_i is as in Algorithm 49) and $|S_{i-1,i-1}| = |\Sigma|^{d+2-(i-1)} = |\Sigma|^{d+1-i}$
- for $j > i-1$, $S_{i-1,j} = H_j(S_{i-1,j-1}) = H_j(\dots H_{i-1}(S_{i-1,i-1})\dots)$.

Then,

$$\Pr[x \in S_{ij} | \check{\mathcal{L}}'] \le (2^\delta + c) \cdot \text{poly}(n) \cdot \text{negl}(n)$$

where c is some constant (independent of δ, d etc.) $\check{\mathcal{L}}'$ is $\mathcal{L}'$ outside $\bar{S}_{i-1}$ (see Notation 43 with $S^{\text{out}} \leftarrow \bar{S}_{i-1}$ and $\mathcal{L}' \leftarrow \mathcal{L}'$) for all $1 \le i \le j \le d$ where the probability is over the randomness in $\mathcal{K}$ (i.e. from $\mathbb{F}^{\delta|\beta}_{\text{inj}}$) and the randomness in Algorithm 77.

Proof. The same as that of Claim 73 except that there, the proof worked for all $i, j \in \{1\dots d\}$. Here, the same arguments apply for a fixed i and β over all values of $j \in \{1\dots d\}$. □

It might not be clear why it suffices to consider only one path, β in the claim if different paths β_i are specified for different is when $\bar{S}_i$ are created using Algorithm 77.

Remark 79. Suppose $\bar{S}_i$ are created successively using Algorithm 77 with $\beta \leftarrow \cup_{i' \in \{1 \ldots, i-1\}} \beta_{i'} =: \beta_{1:i-1}$ and $\bar{S}_{i-1}$ as inputs for index i. Then, the condition $S_{i-1} \backslash X_{i-1}(\beta_{1:i-1}) \subseteq S_{i-1,i-1}$ holds by construction, and it trivially holds that $S_{i-1} \backslash X_{i-1}(\beta_{1:i}) \subseteq S_{i-1} \backslash X_{i-1}(\beta_{1:i-1})$ because $\beta_{1:i} \supseteq \beta_{1:i-1}$. If Claim 78 is invoked with $\bar{S}_{i-1}$ and $\beta \leftarrow \beta_{1:i}$, then the condition $S_{i-1} \backslash X_{i-1}(\beta) \subseteq S_{i-1,i-1}$ is satisfied as required.

7.8.2 d-CodeHashing is hard for CQC_d

Lemma 80 (d-CodeHashing $\notin \mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$). *Every* CQC_d *circuit succeeds at solving* d-CodeHashing *(see Definition 35) with probability at most* $\mathrm{negl}(\lambda)$ *on input* 1^λ *for* $d \leq \mathrm{poly}(\lambda)$*.*

Following the previous proof, we begin with setting up the notation (recall $n = \Theta(\lambda)$).

- Denote by σ_0 the initial state (containing the input 1^λ and ancillae initialised to zero).
- From Notation 22, recall that CQC_d circuits can be represented as[56] $\mathcal{D} = \mathcal{D}_{\tilde{n}} \circ \cdots \circ \mathcal{D}_1$ where $\mathcal{D}_i = \mathcal{B}_{i,d} \circ \mathcal{B}_{i,d-1} \circ \cdots \circ \mathcal{B}_{i,1}$ is a $\mathsf{QC_d}$ circuit with $\mathcal{B}_{i,j} := \Pi_{i,j} \circ U_{i,j} \circ \mathcal{A}_{c,i,j}$. Here $U_{i,j}$ is a single layer unitary and $\mathcal{A}_{c,i,j}$ is a poly sized classical circuit. We drop the subscript "c" from $\mathcal{A}_{c,i,j}$ for brevity.
- Let $\mathcal{L} = (H_0, \ldots H_{d+1})$, d' and Σ be as in definition 35.
- Denote by $\mathcal{D}^{\mathcal{L}} := \mathcal{D}^{\mathcal{L}}_{\tilde{n}} \ldots \mathcal{D}^{\mathcal{L}}_1$ where $\mathcal{D}^{\mathcal{L}}_i = \mathcal{B}^{\mathcal{L}}_{i,d} \mathcal{B}^{\mathcal{L}}_{i,d-1} \ldots \mathcal{B}^{\mathcal{L}}_{i,1}$ and $\mathcal{B}^{\mathcal{L}}_{i,j} = \Pi_{i,j} \circ \mathcal{L} \circ U_{i,j} \circ \mathcal{A}^{\mathcal{L}}_{i,j}$.
  We make the following assumptions which only makes the result stronger (as explained in the $\mathsf{CQ_d}$ case)
  - Classical information entering $U_{i,j}$ and $\mathcal{A}_{i,j}$ is forwarded with their output (for all i, j in their domain).
  - For $i > 1$, $\mathcal{A}_{i,1}$ receives an extra random variable (a set of paths) correlated with $\mathcal{L}$ as input, labelled $\beta^*(s_{i-1})$.
- In the analysis below, we consider $\tilde{n}$ sequences of shadow oracles. Each sequence is denoted by $\vec{\mathcal{M}}_i = (\mathcal{M}_{i,1}, \mathcal{M}_{i,2} \ldots \mathcal{M}_{i,d})$, one for each $\mathcal{D}_i$.
  - We use $\mathcal{D}^{\vec{\mathcal{M}}_i}_i$ to denote $\mathcal{B}^{\vec{\mathcal{M}}_i}_{i,d} \mathcal{B}^{\vec{\mathcal{M}}_i}_{i,d-1} \ldots \mathcal{B}^{\vec{\mathcal{M}}_i}_{i,1}$ where[57] $\mathcal{B}^{\vec{\mathcal{M}}_i}_{i,j} = \Pi_{i,j} \circ \mathcal{M}_{i,j} \circ U_{i,j} \circ \mathcal{A}^{\mathcal{L}}_{i,j}$.
  - $\vec{\mathcal{M}}_i = (\mathcal{M}_{i,j})_j$ are a sequence of shadows of $\mathcal{L}$ created using the sets outputted by Algorithm 77 (and are conditioned on Algorithm 49 succeeding). The input to the algorithm is described later.
- Denote by $\mathcal{D}^{\mathcal{M}} := \mathcal{D}^{\vec{\mathcal{M}}_{\tilde{n}}}_{\tilde{n}} \ldots \mathcal{D}^{\vec{\mathcal{M}}_1}_1$, i.e. a CQC_d circuit with access to only shadow oracles.

The following are essentially unchanged from the $\mathsf{CQ_d}$ case.

- After each circuit $\mathcal{D}_i$, the state is classical and this allows us to consider "transcripts" which we denote by T (the details appear later).
- Parameters for the sampling argument: Same as the $\mathsf{CQ_d}$ case (the advice is now the number of bits sent by $\mathcal{A}_{i,1}$ to $U_{i,1}$).
- Shorthand for $\Pr[\text{find} : \ldots]$ notation: Same as the $\mathsf{CQ_d}$ case.

Proof. Proceeding as in the $\mathsf{CQ_d}$ case, we run algorithm 49 on $\mathcal{L}$ and let E be the event that it does not abort. Observe that

$$\left| \sum_{\mathrm{x} \in X_{\mathrm{valid}}} \Pr[\mathrm{x} \leftarrow \mathcal{D}^{\mathcal{L}}] - \sum_{\mathrm{x} \in X_{\mathrm{valid}}} \Pr[\mathrm{x} \leftarrow \mathcal{D}^{\mathcal{L}} | E] \right| \leq \mathrm{negl}(n) \tag{12}$$

[56] We dropped the preceding $\mathcal{A}_{c,m+1,1}$ classical circuit. This is without loss of generality because it can be accounted for by adding a $\mathcal{D}_{\tilde{n}+1}$; but $\tilde{n}$ is just an arbitrary polynomial of n.

[57] This is a slight abuse of notation because $\vec{\mathcal{M}}_i$ is just a tuple $(\mathcal{M}_{i,1} \ldots \mathcal{M}_{i,d})$ but $\mathcal{B}^{\vec{\mathcal{M}}_i}_{i,j}$ has $\mathcal{A}^{\mathcal{L}}_{i,j}$ which depends on $\mathcal{L}$ explicitly.

as was the case before (recall X_{valid} was the set of valid solutions to d-$\mathsf{CodeHashing}$). We will show in step one, that $\mathcal{D}^{\mathcal{L}}|E$ and $\mathcal{D}^{\mathcal{M}}|E$ have essentially the same behaviour, i.e.

$$\left|\Pr[\mathrm{x} \leftarrow \mathcal{D}^{\mathcal{L}}|E] - \Pr[\mathrm{x} \leftarrow \mathcal{D}^{\mathcal{M}}|E]\right| \leq \mathrm{negl}(n) \tag{13}$$

and then in step two, that $\mathcal{D}^{\mathcal{M}}|E$ succeeds with at most negligible probability at solving d-$\mathsf{CodeHashing}$ (see Definition 35). These two steps, together with Equation (12), entail that $\mathcal{D}^{\mathcal{L}}$ solves d-$\mathsf{CodeHashing}$ with at most $\mathrm{negl}(n)$ probability.

In the rest of the proof, we *implicitly condition everything on the event E*. Let $\bar{S}_0$ be the output of Algorithm 49. Figure 10 may aid in visualising the overarching idea. We also make the simplifying assumption that all classical algorithms only query paths inside the base set $\bar{S}_0$. The general case changes almost nothing, but makes the notation more involved (especially since in this case we have classical algorithms after every layer of unitaries) and can be handled as in the proof of $\mathsf{CQ_d}$ hardness.

Step One. $\mathcal{D}^{\mathcal{L}}|E$ and $\mathcal{D}^{\mathcal{M}}|E$ have essentially the same behaviour.
Using a hybrid argument, one can bound the LHS of Equation (13) by bounding

$$\mathrm{TD}[\mathcal{D}^{\mathcal{L}}, \mathcal{D}^{\mathcal{M}}] = \mathrm{TD}[\mathcal{D}^{\mathcal{L}}_{\tilde{n}} \ldots \mathcal{D}^{\mathcal{L}}_{1}(\sigma_0), \quad \mathcal{D}^{\vec{\mathcal{M}}_{\tilde{n}}}_{\tilde{n}} \ldots \mathcal{D}^{\vec{\mathcal{M}}_1}_{1}(\sigma_0)]$$

with

$$\begin{aligned} &\leq \sum_{i=1}^{\tilde{n}} \mathrm{TD}[\mathcal{D}^{\mathcal{L}}_{\tilde{n}} \ldots \mathcal{D}^{\mathcal{L}}_{i+1} \quad \mathcal{D}^{\mathcal{L}}_{i} \ldots \mathcal{D}^{\mathcal{L}}_{1}, \quad \mathcal{D}^{\mathcal{L}}_{\tilde{n}} \ldots \mathcal{D}^{\mathcal{L}}_{i+1} \quad \mathcal{D}^{\vec{\mathcal{M}}_i}_{i} \ldots \mathcal{D}^{\vec{\mathcal{M}}_1}_{1}] \\ &\leq \sum_{i=1}^{\tilde{n}} \mathrm{TD}[\mathcal{D}^{\mathcal{L}}_{i} \ldots \mathcal{D}^{\mathcal{L}}_{1}, \quad \mathcal{D}^{\vec{\mathcal{M}}_i}_{i} \ldots \mathcal{D}^{\vec{\mathcal{M}}_1}_{1}]. \end{aligned} \tag{14}$$

The $i = 1$ case:
This $i = 1$ case may be seen as an adaptation of the $\mathsf{QC_d}$ hardness proof, using a slightly more general notation suited for our analysis here. Our goal is to bound $\mathrm{TD}[\mathcal{D}^{\mathcal{L}}_1, \mathcal{D}^{\vec{\mathcal{M}}_1}_1]$ but we have not completely specified $\vec{\mathcal{M}}_1$. To this end, consider $\mathcal{D}^{\vec{\mathcal{M}}_1}_1 = \mathcal{B}^{\vec{\mathcal{M}}_1}_{1,d} \ldots \mathcal{B}^{\vec{\mathcal{M}}_1}_{1,1}$ where recall $\mathcal{B}^{\vec{\mathcal{M}}_1}_{1,j} = \Pi_{1,j} \circ \mathcal{M}_{1,j} \circ U_{1,j} \circ \mathcal{A}^{\mathcal{L}}_{1,j}$. Let $\beta_{1,j}$ denote the set of paths (wrt $\bar{S}_0$; see Definition 63) queried by $\mathcal{A}^{\mathcal{L}}_{1,j}$ when $\mathcal{D}^{\vec{\mathcal{M}}_1}_1$ is executed. For $j \in \{1 \ldots d\}$, let $\bar{S}_{1,j}$ be the output of Algorithm 77 with the index $i \leftarrow j$, base sets $\bar{S}_0 \leftarrow \bar{S}_0$, the paths $\beta \leftarrow \cup_{j' \in \{1 \ldots j\}} \beta_{1,j'}$ and the previous sequence of sets $\bar{S}_{i-1} \leftarrow \bar{S}_{1,j-1}$ as inputs.[58] When $j = 1$, instead of $\bar{S}_{1,j-1}$ use $\bar{S}_{i-1} \leftarrow \bar{S}_0$. Finally, define $\mathcal{M}_{1,j}$ as the shadow of $\mathcal{L}$ wrt $\bar{S}_{1,j}$.

Returning to the bound, one can write

$$\begin{aligned} \mathrm{TD}[\mathcal{D}^{\mathcal{L}}_1, \mathcal{D}^{\vec{\mathcal{M}}_1}_1] &= \mathrm{TD}[\mathcal{B}^{\mathcal{L}}_{1,d} \ldots \mathcal{B}^{\mathcal{L}}_{1,1}(\sigma_0), \quad \mathcal{B}^{\vec{\mathcal{M}}_1}_{1,d} \ldots \mathcal{B}^{\vec{\mathcal{M}}_1}_{1,1}(\sigma_0)] \\ &\leq \sum_{j=1}^{d} \mathrm{TD}[\mathcal{B}^{\mathcal{L}}_{1,j}(\rho_{1,j-1}), \mathcal{B}^{\vec{\mathcal{M}}_1}_{1,j}(\rho_{1,j-1})] \\ &\leq \sum_{j=1}^{d} \sqrt{\Pr[\mathrm{find} : U^{\mathcal{L}\backslash\bar{S}_{1,j}}_{1,j}, \mathcal{A}^{\mathcal{L}}_{1,j}(\rho_{1,j-1})]} \end{aligned} \tag{15}$$

where for $j \in \{1 \ldots d-1\}$, $\rho_{1,j} := \mathcal{B}^{\vec{\mathcal{M}}_1}_{1,j} \ldots \mathcal{B}^{\vec{\mathcal{M}}_1}_{1,1}(\sigma_0)$ and we used Lemma 41 (and the relation between TD and B) to obtain the last inequality. To bound the RHS of Equation (15), one can apply Corollary 44. Use $\bar{S}_{1,j}\backslash X(\beta_{1,j+1})$ (see Definition 63) to denote the sequence of sets $(S_{1,j,k}\backslash X_k(\beta_{1,j+1}))_{k \in \{1 \ldots d\}}$. Let $\check{\mathcal{L}}_{1,j}$ be $\mathcal{L}$ outside $\bar{S}_{1,j}\backslash X(\beta_{1,j+1})$ (see Notation 43). Observe that $\rho_{1,j-1}|\check{\mathcal{L}}_{1,j-1}$ is uncorrelated with $\bar{S}_{1,j}|\check{\mathcal{L}}_{1,j-1}$ because $\mathcal{A}^{\mathcal{L}}_{1,j}(\rho_{1,j-1})$ at most specifies $\check{\mathcal{L}}_{1,j-1}$; the queries made by $\mathcal{A}_{1,j}$ have been exposed in $\check{\mathcal{L}}_{1,j-1}$ and are by construction of Algorithm 77, excluded from $\bar{S}_{1,j}$. Using the notation for conditioning $\Pr[\mathrm{find} : \ldots]$, one can apply, for each $j \in \{1 \ldots d\}$, Corollary 44 together with Claim 78 (where $\delta \leftarrow 0$, $\beta \leftarrow \cup_{j' \in \{1,\ldots,j\}} \beta_{1,j'}$, $\bar{S}_0 \leftarrow \bar{S}_0$ and $\bar{S}_{i-1} \leftarrow \bar{S}_{1,j-1}$) to obtain

$$\Pr[\mathrm{find} : U^{\mathcal{L}\backslash\bar{S}_{1,j}}_{1,j}, \mathcal{A}^{\mathcal{L}}_{1,j}(\rho_{1,j-1})|\check{\mathcal{L}}_{1,j-1}] \leq \mathrm{negl}(n).$$

This in turn bounds the RHS of Equation (15) by $\mathrm{negl}(n)$.

The $i = 2$ case:
Since we analysed the $i = 1$ case using a more general notation (than both the $\mathsf{CQ_d}$ case and the $\mathsf{QC_d}$ case),

[58] And the assertion that E happened. We don't explicitly state this any more.

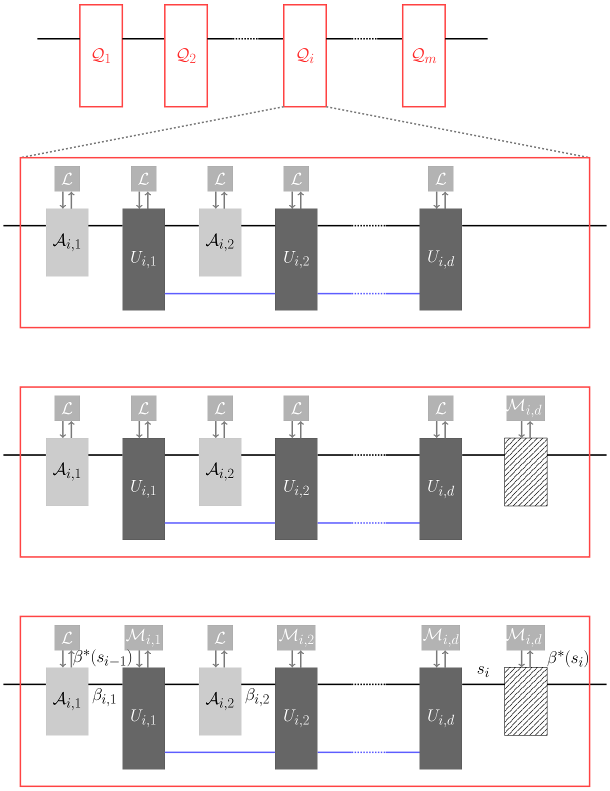


Figure 10: Black lines indicate wires carrying classical information. It is assumed that all circuits append their classical inputs into their classical outputs. The blue wires represent wires carrying quantum information. The figure is meant to illustrate three types of CQC_d circuit, obtained by replacing the red blocks with the three respective circuits enclosed in red boxes. The second circuit is at least as powerful as the first, which in turn is the circuit we wish to study. The shaded circuit in the second red box represents the implementation of the sampling argument.
Notation in the third circuit: For each classical wire, only new information contained in that wire is labelled. For $i \in \{1 \dots \tilde{n}\}$ and $j \in \{1 \dots \tilde{n}\}$, $\beta_{i,j}$ represents the paths in $\mathcal{L}$ queried by $\mathcal{A}_{i,j}$, s_i denotes the measurement outcome after $U_{i,d}$, and $\beta^*(s_i)$ denotes the paths exposed by the sampling argument (the * indicates that the last coordinate may not be known). For $j = 1$, $\mathcal{A}_{i,j}$ also outputs $\beta(s_{i-1})$ which specifies the path $\beta^*(s_{i-1})$ with the last coordinate also revealed.
We show that the behaviour of the second and third circuits is essentially the same where the third has its quantum parts only connected to shadow oracles. It is not hard to establish that the third circuit only solves d-$\mathsf{CodeHashing}$ with at most negligible probability. Together, these prove d-$\mathsf{CodeHashing}$ is hard for CQC_d.

we proceed as in that case but additionally, apply the sampling argument to account for the output of $\mathcal{D}_1^{\vec{\mathcal{M}}_1}$. Let $\sigma_1 := \mathcal{D}_1^{\vec{\mathcal{M}}_1}(\sigma_0)$. Our goal is to bound $\mathrm{TD}[\mathcal{D}_2^{\mathcal{L}}(\sigma_1), \mathcal{D}_2^{\vec{\mathcal{M}}_2}(\sigma_1)]$ but we have not yet specified $\vec{\mathcal{M}}_2$.

To this end, we apply the sampling argument. Let[59] $\beta_1 := \cup_{j=1}^{d} \beta_{1,j}$ be the set of paths queried by the classical algorithms in $\mathcal{D}_1^{\vec{\mathcal{M}}_1}$. Note that $\mathsf{inj}[\mathcal{L}|\beta_1]$ (see Notation 76) is distributed as $\mathbb{F}_{\text{inj}}^{|\beta_1}$. Let the string s_1 denote the output of $\mathcal{D}_1^{\vec{\mathcal{M}}_1}$. Given that $\Pr[s_1|\beta_1] \geq \gamma$, $\mathsf{inj}[\mathcal{L}|\beta_1 s_1]$ may be expressed as a convex combination (as described in Proposition 70) over $\mathsf{inj}[\mathcal{L}|s_1\beta_1\beta^*(s_1)]$ which are distributed as $\mathbb{F}_{\text{inj}}^{(p,\delta)|\beta_1}$ where $|\beta^*(s_1)| \leq p \leq 2m/\delta$ whenever the convex coefficient is larger than γ. When $\Pr[s_1|\beta_1] < \gamma$, let $\beta^*(s_1) = \varnothing$. This implicitly defines the random variable $\beta^*(s_1)$ which was initially left unspecified. The first (classical) circuit of $\mathcal{D}_2^{\vec{\mathcal{M}}_2}$, i.e. $\mathcal{A}_{2,1}^{\mathcal{L}}$, takes as input s_1 and $\beta^*(s_1)$. We assume (without loss of generality) $\mathcal{A}_{2,1}^{\mathcal{L}}$ learns $\beta(s_1)$ which is $\beta^*(s_1)$ with $\perp$s replaced by the value $\mathcal{L}$ takes in the last coordinate.

We now proceed as in the $i = 1$ case and consider $\mathcal{D}_2^{\vec{\mathcal{M}}_2} = \mathcal{B}_{2,d}^{\vec{\mathcal{M}}_2} \dots \mathcal{B}_{2,1}^{\vec{\mathcal{M}}_2}$ acting on σ_1 where $\mathcal{B}_{2,j}^{\vec{\mathcal{M}}_2} = \Pi_{2,j} \circ \mathcal{M}_{2,j} \circ U_{2,j} \circ \mathcal{A}_{2,j}^{\mathcal{L}}$. Let $\beta_{1,j}$ denote the set of paths queried by $\mathcal{A}_{2,j}^{\mathcal{L}}$ when $\mathcal{D}_2^{\vec{\mathcal{M}}_2}$ is executed (for $j = 1$, $\beta_{1,j}$ counts paths distinct from $\beta(s_1)$). For $j \in \{1 \dots d\}$, let $\bar{S}_{2,j}$ be the output of Algorithm 77 with index $i \leftarrow j$, base sets $\bar{S}_0 \leftarrow \bar{S}_0$, the paths $\beta \leftarrow \cup_{j' \in \{1 \dots j\}} \beta_{2,j'} \cup \beta_1 \cup \beta(s_1)$, and the previous sequence of sets $\bar{S}_{i-1} \leftarrow \bar{S}_{2,j-1}$ as inputs. When $j = 1$, instead use $\bar{S}_{i-1} \leftarrow \bar{S}_0$. Finally, define $\mathcal{M}_{2,j}$ as the shadow of $\mathcal{L}$ wrt $\bar{S}_{2,j}$.

Returning to the bound, one can write

$$\begin{aligned}
\mathrm{TD}[\mathcal{D}_2^{\mathcal{L}}(\sigma_1), \mathcal{D}_2^{\vec{\mathcal{M}}_2}(\sigma_1)] &\leq \sum_{j=1}^{d} \mathrm{TD}[\mathcal{B}_{2,j}^{\mathcal{L}}(\rho_{2,j-1}), \mathcal{B}_{2,j}^{\vec{\mathcal{M}}_2}(\rho_{2,j-1})] \\
&\leq \sum_{j=1}^{d} \sqrt{\Pr[\mathrm{find} : U_{2,j}^{\mathcal{L}\backslash \bar{S}_{2,j}}, \mathcal{A}_{2,j}^{\mathcal{L}}(\rho_{2,j-1})} \qquad (16)
\end{aligned}$$

where for $j \in \{1 \dots d\}$, $\rho_{2,j} := \mathcal{B}_{2,j}^{\vec{\mathcal{M}}_2} \dots \mathcal{B}_{2,1}^{\vec{\mathcal{M}}_2}(\sigma_1)$ and we used Lemma 41 to get the last inequality. To bound the RHS Equation (16), one can apply Corollary 44. Let $\check{\mathcal{L}}_{2,j}$ be $\mathcal{L}$ outside $\bar{S}_{2,j} \backslash X(\beta_{2,j+1})$ (see Notation 43). Observe that $\rho_{2,j-1}|\check{\mathcal{L}}_{2,j-1}$ is uncorrelated with $\bar{S}_{2,j}|\check{\mathcal{L}}_{2,j-1}$ (for the same reason as the $i = 1$ case). However, to apply Claim 78 we condition on the "transcript" until the output of $\mathcal{A}_{2,1}$, i.e. $T_2 =: T(\mathcal{A}_{2,1}(\sigma_1)) := (\beta_1, s_1, \beta(s_1), \beta_{2,1})$, by writing $\Pr[\mathrm{find} : U_{2,j}^{\mathcal{L}\backslash \bar{S}_{2,j}}, \mathcal{A}_{2,j}^{\mathcal{L}}(\rho_{2,j-1})|\check{\mathcal{L}}_{2,j-1}]$

$$\begin{aligned}
&= \sum_{s_1, \beta_1, \beta(s_1), \beta_{2,1}} \Pr[T_2] \cdot \Pr[\mathrm{find} : U_{2,j}^{\mathcal{L}\backslash \bar{S}_{2,j}}, \mathcal{A}_{2,j}^{\mathcal{L}}(\rho_{2,j-1})|\check{\mathcal{L}}_{2,j-1} T_2] \\
&\leq \sum_{\substack{s_1 : \Pr[s_1|\beta_1] \geq 2^{-m} \\ \beta_1, \beta(s_1), \beta_{2,1}}} \underbrace{\Pr[\beta_{2,1}|s_1\beta_1\beta(s_1)] \Pr[\beta(s_1)|s_1\beta_1] \Pr[s_1|\beta_1] \Pr[\beta_1]}_{\Pr[T_2]} \cdot \Pr[\mathrm{find} : U_{2,j}^{\mathcal{L}\backslash \bar{S}_{2,j}}, \mathcal{A}_{2,j}^{\mathcal{L}}(\rho_{2,j-1})|\check{\mathcal{L}}_{2,j-1} T_2] + 2^{-(m-\tilde{m})} \\
&\leq \sum_{\substack{s_1 : \Pr[s_1|\beta_1] \geq 2^{-m} \\ \beta(s_1) : \Pr[\beta(s_1)|s_1\beta_1] \geq 2^{-m} \\ \beta_1, \beta_{2,1}}} \Pr[\beta_{2,1}|s_1\beta_1\beta(s_1)] \Pr[\beta(s_1)|s_1\beta_1] \Pr[s_1|\beta_1] \Pr[\beta_1] \cdot \underbrace{\Pr[\mathrm{find} : U_{2,j}^{\mathcal{L}\backslash \bar{S}_{2,j}}, \mathcal{A}_{2,j}^{\mathcal{L}}(\rho_{2,j-1})|\check{\mathcal{L}}_{2,j-1} T_2]}_{\text{Term I}} + 2 \cdot 2^{-(m-\tilde{m})} \\
&\leq \mathrm{negl}(n)
\end{aligned}$$

where to obtain the first inequality (proceeding almost exactly as in the $\mathsf{CQ_d}$ case), we note that for each $s_1 : \Pr[s_1|\beta_1] \geq 2^{-m}$, one can use Proposition 70 and one can account for all $s_1 : \Pr[s_1|\beta_1] < 2^{-m}$, by simply upper bounding the sum by $2^{-(m-\tilde{m})}$ because s_1 is of length $\tilde{m}$. In the second inequality, we use the fact that either the convex weight (i.e. $\Pr[\beta(s_1)|s_1\beta_1]$) as specified in Proposition 70 is less than 2^{-m} (for at most each s_1, there it contributes at most $2^{-(m-\tilde{m})}$ to the sum) or it is greater than 2^{-m}. In the latter case, the injective shuffler is distributed as $\mathbb{F}_{\text{inj}}^{\delta|\beta}$ where $\beta \leftarrow \beta_1 \cup \beta(s_1) \cup \beta_{2,1}$. Therefore, one can apply Corollary 44 together with Claim 78 (where $\delta \leftarrow \delta$, $\beta \leftarrow \cup_{j' \in \{1, \dots j\}} \beta_{2,j'} \cup \beta_1 \cup \beta(s_1)$, $\bar{S}_0 \leftarrow \bar{S}_0$ and $\bar{S}_{i-1} \leftarrow \bar{S}_{2,j-1}$) to obtain Term $1 \leq 2^{\delta} \cdot \mathrm{poly}(n) \cdot \mathrm{negl}(n)$.

The general $i \in \{1 \dots \tilde{n}\}$ case:
This is a straightforward generalisation of the $i = 2$ case and hence we only outline the key steps. Let $\sigma_i := \mathcal{D}_i^{\vec{\mathcal{M}}_i} \dots \mathcal{D}_1^{\vec{\mathcal{M}}_1}(\sigma_0)$ where $\mathcal{M}_{i-1,j}$ is the shadow of $\mathcal{L}$ wrt $\bar{S}_{i-1,j}$ which in turn are defined below. It may help to keep the last circuit of Figure 10 in mind. Consider $\mathcal{D}_i^{\vec{\mathcal{M}}_i} = \mathcal{B}_{i,d}^{\vec{\mathcal{M}}_i} \dots \mathcal{B}_{i,1}^{\vec{\mathcal{M}}_i}$ acting on σ_{i-1} where recall

[59] We slightly abuse the notation. By $\cup_{j=1}^{d} \beta_{1,j}$ we mean component-wise union as each $\beta_{1,j}$ is a sequence of sets.

$\mathcal{B}_{i,j}^{\vec{\mathcal{M}}_2} = \Pi_{i,j} \circ \mathcal{M}_{i,j} \circ U_{i,j} \circ \mathcal{A}_{i,j}^{\mathcal{L}}$. Let $\beta_{i,j}$ denote the set of paths queried by $\mathcal{A}_{i,j}^{\mathcal{L}}$ when $\mathcal{D}_i^{\vec{\mathcal{M}}_i}$ is executed. Let $\beta_i := \cup_{j\in\{1,\dots d\}}\beta_{i,j}$. Let s_i denote the string output by $\mathcal{D}_i^{\vec{\mathcal{M}}_i}(\sigma_{i-1})$.

Now, we apply the sampling argument to $\mathsf{inj}[\mathcal{L}|\beta_1 s_1 \beta(s_1)\dots\beta_{i-1}]$. Let $\beta^*(s_{i-1})$ be the paths as in Proposition 70 such that when $\Pr[s_{i-1}|\beta_1 s_1\beta(s_1)\dots\beta_{i-1}] \geq \gamma$, $\mathsf{inj}[\mathcal{L}|\beta_1 s_1 \beta(s_1)\dots\beta_{i-1}s_{i-1}]$ is distributed as $\mathbb{F}_{\mathrm{inj}}^{(i-2)\delta|\beta_1\cup\dots\cup\beta_{i-1}\cup\beta(s_1)\cup\dots\beta(s_{i-1})}$ so that[60] $\mathsf{inj}[\mathcal{L}|\beta_1 s_1\beta(s_1)\dots\beta_{i-1}s_{i-1}\beta^*(s_{i-1})]$ is distributed as $\mathbb{F}_{\mathrm{inj}}^{(p,(i-2)\delta)|\beta_1\cup\dots\cup\beta_{i-1}\cup\beta(s_1)\cup\dots\cup\beta(s_{i-1})}$ whenever the convex coefficient (i.e. probability associated with $\beta^*(s_{i-1})$) is larger than $\gamma = 2^{-m}$. $\beta(s_{i-1})$ is $\beta^*(s_{i-1})$ with $\perp$s replaced by the values taken by $\mathcal{L}$ at those coordinates.

Returning to $\bar{S}_{i,j}$, define it to be the output of Algorithm 77 with index $i \leftarrow j$, base sets $\bar{S}_0 \leftarrow \bar{S}_0$, the paths $\beta \leftarrow \left(\cup_{j'\in\{1\dots j\}}\beta_{i,j'}\right) \cup (\beta_{i-1}\cup\dots\cup\beta_1)\cup(\beta(s_{i-1})\cup\dots\cup\beta(s_1))$, and the previous sequence of sets $\bar{S}_{i-1} \leftarrow \bar{S}_{i,j-1}$ as inputs. When $j=1$, use $\bar{S}_{i-1}\leftarrow\bar{S}_0$ instead.

To obtain the bound, we need two more definitions. Let the "transcript" be denoted by $T_i :=: T(\mathcal{A}_{i,1}(\sigma_{i-1})) := (\beta_1, s_1, \beta(s_1), \dots \beta_{i-1}, s_{i-1}, \beta(s_{i-1}), \beta_{i,1})$. Let $\check{\mathcal{L}}_{i,j}$ be $\mathcal{L}$ outside $\bar{S}_{i,j}\backslash X(\beta_{i,j+1})$ (see Notation 43).

We bound the ith term in Equation (14) by expressing it as

$$\mathrm{TD}[\mathcal{D}_i^{\mathcal{L}}(\sigma_{i-1}), \mathcal{D}_i^{\vec{\mathcal{M}}_i}(\sigma_{i-1})] \leq \sum_{j=1}^{d}\mathrm{TD}[\mathcal{B}_{i,j}^{\mathcal{L}}(\rho_{i,j-1}), \mathcal{B}_{i,j}^{\vec{\mathcal{M}}_i}(\rho_{i,j-1})]$$

where $\rho_{i,j} := \mathcal{B}_{i,j}^{\vec{\mathcal{M}}_i}\dots\mathcal{B}_{i,j}^{\vec{\mathcal{M}}_i}(\sigma_{i-1})$. The square of the jth term can then be bounded (using Lemma 41) by $\Pr[\mathrm{find}: U_{i,j}^{\mathcal{L}\backslash\bar{S}_{i,j}}, \mathcal{A}_{i,j}^{\mathcal{L}}(\rho_{i,j-1})]$ which is

$$\leq \sum_{\substack{s_1:\Pr[s_1|\beta_1]\geq 2^{-m},\dots,s_{i-1}:\Pr[\beta_1\dots]\geq 2^{-m}\\ \beta(s_1):\Pr[\beta(s_1)|s_1\beta_1]\geq 2^{-m},\dots\beta(s_{i-1}):\Pr[\beta(s_{i-1})|s_1\beta_1\dots]\geq 2^{-m}\\ \beta_1,\beta_2\dots\beta_{i-1},\ \beta_{i,1}}} \Pr[T_i]\cdot\underbrace{\Pr[\mathrm{find}: U_{i,j}^{\mathcal{L}\backslash\bar{S}_{i,j}}, \mathcal{A}_{i,j}^{\mathcal{L}}(\rho_{i,j-1})|\check{\mathcal{L}}_{i,j-1}, T_i]}_{\text{Term I}} + 2\cdot(i-1)\cdot 2^{-(m-\tilde{m})}$$

$$\leq 2^{\Delta}\cdot\mathrm{poly}(n)\cdot\mathrm{negl}(n) + 2\cdot(i-1)\cdot 2^{-(m-\tilde{m})} \leq \mathrm{negl}(n)$$

where the distribution of the injective shuffler in Term I is $\mathbb{F}^{i\cdot\delta|(\beta(s_{i-1})\cup\dots\beta(s_1))\cup(\beta_{i-1}\cup\dots\beta_1)\cup\beta_{i,1}}$. This is obtained by repeatedly applying Proposition 70 (for the kth application, $\delta' \leftarrow (k-1)\delta$, $\beta \leftarrow \beta_{k,1}\cup(\beta(s_{k-1})\cup\dots\beta(s_1))\cup(\beta_{k-1}\cup\dots\beta_1)$ and $\bar{S}_0\leftarrow\bar{S}_0$) and arguing as before to collect terms for which the distribution over the injective shuffler is unknown (but which occur with probability at most 2^{-m}). Independence of $\bar{S}_{i,j}$ from $\rho_{i,j-1}$ can be argued as before once it is conditioned on $\check{\mathcal{L}}_{i,j-1}$ and one can apply Corollary 44 together with Claim 78 (with $\delta\leftarrow i\cdot\delta$, $\beta\leftarrow\left(\cup_{j'\in\{1,\dots,j\}}\beta_{i,j'}\right)\cup(\beta_{i-1}\cup\dots\beta_1)\cup(\beta(s_{i-1})\cup\dots\beta(s_1))$, $\bar{S}_0\leftarrow\bar{S}_0$ and $\bar{S}_{i-1}\leftarrow\bar{S}_{i,j-1}$) to obtain the stated bound on Term I (recall $\gamma = 2^{-m}$ and $\delta = \Delta/\tilde{n}$).

Step Two. $\mathcal{D}^{\mathcal{M}}|E$ succeeds at solving d-CodeHashing with at most negligible probability. The argument for the $\mathsf{CQ_d}$ case go through with the only change that there are more classical algorithms to account for but this does not affect the conclusion. □

8 Proof of Quantum Depth

Since YZ's CodeHashing can be efficiently verified (i.e. it is NP), it is evident that d-CodeHashing can also be efficiently verified. Therefore d-CodeHashing also serves a proof of quantum depth. However, in the cryptographic context, one would ideally like security against oracle dependent adversaries (in our proofs so far, we assumed the adversary is oracle independent). Fortunately, this issue can be resolved generically and to this end, we first formalise what we mean by a proof of quantum depth. YZ also followed a similar approach for their proof of quantumness which is based on CodeHashing.

8.1 The Definition

It may help to recall the definitions of uniform and non-uniform oracle dependent adversaries (see Subsection 3.2.2).

Definition 81 (Proof of d Quantum Depth in the Random Oracle Model)**.** Consider three algorithms, $(\mathsf{Gen}, \mathsf{Verify}^H)$ and Prove^H.

[60] Note that $i \geq 2$ here because s_{i-1} is s_1 when $i = 2$.

$\mathsf{Gen}(1^\lambda)$. A PPT algorithm which returns $(\mathsf{sk}, \mathsf{pk})$.

$\mathsf{Verify}^H(\mathsf{sk}, \mathsf{pk}, \pi)$. A PPT algorithm that makes at most $\mathrm{poly}(\lambda)$ queries to H and outputs 0 or 1.

$\mathsf{Prove}^H(\mathsf{pk})$. Consider an oracle independent quantum circuit family $\{\mathcal{C}_n\}_n$. $\mathsf{Prove}^H(\mathsf{pk})$ executes $\mathcal{C}_{|\mathsf{pk}|}$ with input pk.

The algorithms $(\mathsf{Gen}, \mathsf{Verify}^H)$ and Prove^H constitute a Proof of d Quantum Depth in the Random Oracle Model, if the following holds for every sufficiently large security parameter λ.

- *Completeness.* There is an honest prover which applies a poly-sized quantum circuit, i.e. $\{\mathcal{C}_n\} \in \mathsf{QPT}$ for all n, with the following property. Let $\mathsf{Prove}^H(\mathsf{pk})$ be $\mathcal{C}_{|\mathsf{pk}|}$ with input pk. Then, the verifier interacts with the prover and accepts with overwhelming probability, i.e.

$$\Pr_H\left[\mathsf{Verify}^H(\mathsf{sk}, \mathsf{pk}, \pi) = 1 : \begin{smallmatrix}(\mathsf{sk},\mathsf{pk})\leftarrow\mathsf{Gen}(1^\lambda)\\ \pi\leftarrow\mathsf{Prove}^H(\mathsf{pk})\end{smallmatrix}\right] \geq 1 - \mathrm{negl}(\lambda).$$

- *Soundness.* Consider any arbitrary prover which applies a CQC_d circuit, i.e. $\{\mathcal{C}_n\}_n$ where each $\mathcal{C}_n \in \mathsf{CQC}_d$. Let $\mathsf{Prove}^H(\mathsf{pk})$ be $\mathcal{C}_{|\mathsf{pk}|}$ with input pk. Then, the verifier interacting with any such prover accepts with negligible probability, i.e.

$$\Pr_H\left[\mathsf{Verify}^H(\mathsf{sk}, \mathsf{pk}, \pi^*) = 1 : \begin{smallmatrix}(\mathsf{sk},\mathsf{pk})\leftarrow\mathsf{Gen}(1^\lambda)\\ \pi^*\leftarrow\mathsf{Prove}^H(\mathsf{pk})\end{smallmatrix}\right] \leq \mathrm{negl}(\lambda)$$

  for all Prove^H.

Soundness against uniformly and non-uniformly oracle dependent provers is defined analogously. When $|\mathsf{pk}| = 0$, the prover is given 1^λ as input.

Observe that the protocol above is a two-message protocol (the verifier sends pk and the prover sends π). In fact, observe that any two-message protocol (where the verifier is classical and sends the first message) can be cast in the aforementioned form by splitting the verification algorithm into two $(\mathsf{Gen}, \mathsf{Verify})$ and have all the information from Gen passed to Verify and some information from Gen passed to Prove as the first message. In addition to being two-message, the protocol above may also have the following properties if the appropriate conditions are satisfied.

Publicly verifiable: If $|\mathsf{sk}| = 0$ the proof can be publicly verified by looking at the transcript.[61] If, in addition, $|\mathsf{pk}| > 0$, then we call the proof of quantum depth *keyed*.

Non-interactive (or *keyless*): If $|\mathsf{pk}| = 0$ the verifier does not need to send any information to the prover. Note that soundness in this case cannot hold against non-uniform adversaries.[62]

We conclude by noting that in our definition of proof of quantum depth, we allowed the completeness to be BQP which may not be practical. This is analogous to the definition of proof of quantumness where the soundness is against BPP and completeness is again BQP. In both cases, it is desirable to have low depth circuits[63] suffice for establishing completeness. Nonetheless, they are meaningful formalisations because they do certify the respective notions of quantum depth and quantumness.

8.2 Salting and oracle dependent adversaries

For non-interactive proofs of quantum depth (as for non-interactive proofs of quantumness [YZ22]) in the random oracle model, security holds only against oracle-independent adversaries, i.e. adversaries that are fixed before the random oracle is chosen, but not against non-uniform oracle-dependent adversaries, i.e. adversaries that receive advice strings after the random oracle has been chosen. To see this, observe that the advice can be arbitrarily correlated with the chosen random oracle. For instance, in our setting, the advice could simply be the codeword that hashes as required. Then, an adversary which simply outputs the advice it receives can already break the security of the proof of quantum depth protocol.

[61] It is standard practice to assume that the algorithms themselves are public knowledge.

[62] The proof can be hardcoded into the prover's advice.

[63] Ideally, $\mathsf{QNC}_{\mathcal{O}(1)}$ for quantumness and $\mathsf{QNC}_{\mathcal{O}(d+1)}$ for quantum depth

To also achieve security against non-uniform oracle-dependent adversaries, we rely on a result of Chung et al. [CGLQ20]. In this work, it was shown (among other results) that salting, i.e. appending a random string to the query, can be used to render the oracle-dependent advice useless. This is a quantum adaptation of the results of [CDGS18] and can be used in our setting to turn a non-interactive proof of quantum depth secure against oracle-independent adversaries into an interactive (two message) proof of quantum depth secure against non-uniform oracle-dependent adversaries, in which the first message (sent by the verifier) only consists simply of a random string. More formally, the following holds.

Theorem 82. *Let* $\mathsf{PoQD} = (\mathsf{Prove}^H, \mathsf{Verify}^H(\pi))$ *be a keyless proof of quantum depth secure against oracle-independent adversaries, then* $\mathsf{PoQD}' = (\mathsf{Gen}'(1^\lambda), \mathsf{Prove}'^H(\mathsf{pk}), \mathsf{Verify}'^H(\mathsf{pk}, \pi))$ *is a keyed proof of quantum depth secure against oracle-dependent non-uniform adversaries, where* $\mathsf{Gen}(1^\lambda)$ *simply outputs a random* pk*, i.e.* $\mathsf{pk} \leftarrow \{0,1\}^\lambda$.

This theorem is an immediate consequence of [CGLQ20, Theorem 7.4] where a proof of quantum depth is viewed as a publicly verifiable security game [CGLQ20, Definition 3.3]. Yamakawa and Zhandry used the same result to lift their oracle-independent security to non-uniform security for the case of proofs of quantumness, one-way functions and collision resistant functions [YZ22, Theorem 3.7 & 3.8].

8.3 A Proof of d Quantum Depth

We give a *non-interactive* Proof of d Quantum Depth protocol, sound against oracle independent adversaries (see Definition 81). In the following, let $\tilde{H}$, and $\mathcal{C}$ be as in Definition 35.

$\mathsf{Verify}^H(1^\lambda, \pi)$**.** Verify^H parses π as $\mathrm{x} = (\mathrm{x}_1, \dots \mathrm{x}_n)$ and checks if (a) $\mathrm{x} \in \mathcal{C}$ and (b) $\tilde{H}(\mathrm{x}) = 1$. If both conditions are satisfied, it outputs 1, otherwise it outputs 0.

$\mathsf{Prove}^H(1^\lambda)$**.** It runs the QPT machine in Theorem 34 with $\tilde{H}$ as the random oracle and returns the output x as π.

Completeness is immediate from Theorem 34. Soundness against oracle independent d depth circuits (i.e. circuits in CQC_d) follows directly from Lemma 36. As discussed in Section 8 above, using known results, we obtain the following.

Theorem 83 (d-Proof of Quantum Depth)**.** *There is a publicly verifiable Proof of d Quantum Depth (see Definition 81) sound against non-uniform oracle dependent adversaries.*

9 Improved Upper Bound

To obtain the fine-grained separation $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}} \subsetneq \mathsf{BPP}^{\mathsf{QNC}_{2d+\mathcal{O}(1)}^{\mathsf{BPP}}}$, we introduce $\mathsf{CollisionHashing}$.

9.1 CollisionHashing

$\mathsf{CollisionHashing}$ is essentially the same problem used by [BKVV20] to obtain a proof of quantumness protocol except that instead of using claw-free function, we use a random function. $\mathsf{CollisionHashing}$ shows $\mathsf{QNC}_{\mathcal{O}(1)} \notin \mathsf{BPP}$, relative to a random oracle and satisfies classical query soundness. The main limitation of $\mathsf{CollisionHashing}$ is that it cannot be efficiently verified, unlike YZ's $\mathsf{CodeHashing}$.

The following elementary result about the probability of producing a superposition of two pre-images relative to a random oracle, would be useful in analysing $\mathsf{CollisionHashing}$.

Claim 84. Let $g : A \to B$ be a random function where A and B are finite sets with $|A| \geq |B|$ and $\log |A|, \log |B| \leq \mathrm{poly}(n)$. Then there is a QNC_2 circuit with oracle access to g which produces the state

$$\frac{|a_0\rangle + |a_1\rangle}{\sqrt{2}} \tag{17}$$

with probability at least c where $\{a_0, a_1\} = g^{-1}(b)$ for some $b \in B$ and $0 < c < 1$ depends only on $|A|$ and $|B|$. Further, $\lim_{|A| \to \infty} c \geq k^2/2(e^k - 1)$ when $|A| = k|B|$ for $k \in \mathbb{N}$.

Proof. Producing $\sum_{a\in A}|a\rangle|g(a)\rangle/\sqrt{|A|}$ takes one layer of Hadamards and a call to the oracle for g and therefore QNC_2 can prepare this state. If the second register is measured, the probability that the first register holds (up to normalisation) $|a_0\rangle+|a_1\rangle$ is then $\Pr[|g^{-1}(b)|=2|b\in g(A)]$ where the probability is over $g\overset{\$}{\leftarrow}\text{Functions}(A\to B)$. That in turn, for any fixed b, may be computed as follows[64]

$$c(|A|,|B|)=\frac{\left|\{g:|g^{-1}(b)|=2\}\right|}{|\{g:b\in g(A)\}|}=\frac{\frac{|A|\cdot(|A|-1)}{2!}\cdot(|B|-1)^{|A|-2}}{(|B|^{|A|}-(|B|-1)^{|A|})}$$

where to obtain the numerator, we count the number of ways of choosing exactly two points in A (which are mapped to b) and the number of ways of assigning non-b values to the remaining $|A|-2$ points. To obtain the denominator, we count the number of functions from A to B and subtract from it all functions which do not map to b, i.e. none of the $|A|$ points are assigned the value b. Using $|A|=k|B|$, $\lim_{|A|\to\infty}(1-k/|A|)^{|A|}=e^{-k}$ and with some simplification, one obtains $\lim_{|A|\to\infty}c\geq k^2/2(e^k-1)$. □

We now state the $\mathsf{CollisionHashing}$ problem as follows.

Definition 85 ($\mathsf{CollisionHashing}$)**.** The $\mathsf{CollisionHashing}$ problem is defined by (Gen,R_H) where $\mathsf{Gen}(1^\lambda)=1^\lambda$ and the relation R_H is specified as follows: Let $g:\{0,1\}^{\lambda+1}\to\{0,1\}^\lambda$ be a random function, let $H':\{0,1\}^*\to\{0,1\}$ be another random function (both generated using H in some canonical way) and let c be as in Claim 84. We say $(1^\lambda,((y_i,m_i,r_i)_{i\in\{1\ldots\lambda\}})\in R_H$ if the following hold

1. all y_i are distinct
2. $\frac{|I|}{\lambda}>\frac{3c}{4}$ where $I\subseteq[1\ldots\lambda]$ is the subset of indices satisfying $\left|g^{-1}(y_i)\right|=2$ for all $i\in I$.
3. $\frac{\text{count}}{|I|}>\frac{3}{4}$ where count $=\sum_{i\in I}\text{valid}(i)$ and valid(i) returns 1 if the following holds, otherwise it returns 0: $m_i=r_i\cdot(z_{i0}\oplus z_{i1})\oplus H'(z_{i0})\oplus H'(z_{i1})$ where $\{z_{i0},z_{i1}\}=g^{-1}(y_i)$.

$\mathsf{CollisionHashing}$ satisfies the following properties.

Lemma 86. *Let* $\mathsf{CollisionHashing}$ *be as stated in Claim 84. It satisfies the following properties*

- *Completeness:* QNC_{10} *can solve* $\mathsf{CollisionHashing}$ *with probability* $1-\text{negl}(\lambda)$
- *Soundness:* $\mathsf{CollisionHashing}$ *satisfies classical query soundness.*
- *Bounded Oracle Domain:* $\mathsf{CollisionHashing}$ *has a bounded oracle domain of size at most* $2^{3\lambda}$.

Proof sketch. Completeness: [BKVV20] showed that if one is given poly(λ) copies of the state Equation (17), then using at most 7 layers of quantum operations, one can solve $\mathsf{CollisionHashing}$ with probability $1-\text{negl}(\lambda)$. The aforesaid state can be generated with probability c (in at most 2 quantum layers) therefore the probability of generating $0.75c\lambda$ states is $1-\text{negl}(n)$ (using Chernoff). Thus, QNC_{10} can solve the problem with $1-\text{negl}(n)$ probability.

Classical query soundness: [BKVV20] showed that every PPT machine solves $\mathsf{CollisionHashing}$ with probability at most[65] negl(λ). Their argument is more general. Their proof showed that succeeding with non-negligible probability implies one can find collisions which is assumed to be hard. More precisely, they neither require the machine to be PPT (only that access to the oracle H' is classical), nor that *PPT* machines cannot find collisions in the function (which is g in this case) but only assert that collisions can be extracted. For establishing classical query soundness, it suffices to show that with only polynomially many classical queries to H, no (potentially unbounded) machine can solve $\mathsf{CollisionHashing}$. It is known that finding collisions in g (a random function) with non-negligible probability requires at least $\Omega(2^{n/O(1)})$ (quantum) queries. Using [BKVV20]'s argument (see their Section 3.2) on elements in the set $I\subseteq\{1\ldots\lambda\}$, we deduce that solving $\mathsf{CollisionHashing}$ with non-negligible probability implies there is an algorithm that finds collisions in g by making only polynomially many (classical) queries to g which in turn violates the previous statement. Thus, we conclude $\mathsf{CollisionHashing}$ satisfies classical query soundness.

[64] Note that the aforementioned probability is the same for every fixed b and the probability (over g) of getting any fixed b upon measurement of the second register is also the same by symmetry.

[65] They show it for their problem but the results carry over unchanged.

Bounded Oracle Domain: By inspection, it is clear that H' is only queried on a domain of size $2^{2\lambda}$ and g is only queried on a domain of size $2^{2\lambda}$. Since both are generated using H, we take $2^{3\lambda}$ as a loose upper bound on the oracle domain. □

Problem	$\in$		$\notin$	**Assumption**	**Verification**	**Interpretation**	**Remarks**
$\mathsf{CollisionHashing}$	$\mathsf{QNC}_{\mathcal{O}(1)}$	$\not\subseteq$	BPP	RO	No	Even the simplest constant quantum depth is hard to simulate	Definition 85
d-$\mathsf{Rec}[\mathsf{CollisionHashing}]$	$\mathsf{QNC}_{2d+\mathcal{O}(1)} \subseteq \mathsf{BPP}^{\mathsf{QNC}^{\mathsf{BPP}}_{2d+\mathcal{O}(1)}}$	$\not\subseteq$	$\mathsf{BPP}^{\mathsf{QNC}^{\mathsf{BPP}}_{d}}$	RO	No	Finer refutation of Jozsa's conjecture in ROM	Theorem 87

Table 5: We tighten the quantum depth bounds to $\mathsf{BPP}^{\mathsf{QNC}^{\mathsf{BPP}}_{d}} \subsetneq \mathsf{BPP}^{\mathsf{QNC}^{\mathsf{BPP}}_{2d+\mathcal{O}(1)}}$ relative to the random oracle.

9.2 Jozsa's conjecture/Aaronson's challenge

Using Lemma 30, observe that d-$\mathsf{Rec}[\mathsf{CollisionHashing}]$ can be solved in $\mathsf{QNC}_{2d+\mathcal{O}(1)}$. From Lemma 31, observe that d-$\mathsf{Rec}[\mathsf{CollisionHashing}]$ cannot be solved in CQC_d. We therefore have the following.

Theorem 87 (Stronger refutation of Jozsa's conjecture.)**.** *With respect to a random oracle, the following hold:* $\mathsf{QNC}_{2d+\mathcal{O}(1)} \not\subseteq \mathsf{BPP}^{\mathsf{QNC}^{\mathsf{BPP}}_{d}}$*, which implies* $\mathsf{BPP}^{\mathsf{QNC}^{\mathsf{BPP}}_{d}} \subsetneq \mathsf{BPP}^{\mathsf{QNC}^{\mathsf{BPP}}_{2d+\mathcal{O}(1)}}$.

Part II
Separations of Hybrid Quantum Depth

In the previous discussions, we studied the relation of $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$ with $\mathsf{BPP}^{\mathsf{QNC}_{d'}^{\mathsf{BPP}}}$. In particular, we showed that relative a random oracle, $\mathsf{BQP} \not\subseteq \mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$ and we tightened this result to $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}} \not\subseteq \mathsf{BPP}^{\mathsf{QNC}_{d+\mathcal{O}(1)}^{\mathsf{BPP}}}$. We now study the relation between $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$, $\mathsf{QNC}_d^{\mathsf{BPP}}$ and $\mathsf{BPP}^{\mathsf{QNC}_d}$. We define a problem and prove that it can be solved by $\mathsf{QNC}_{\mathcal{O}(1)}^{\mathsf{BPP}}$ but not by $\mathsf{BPP}^{\mathsf{QNC}_d}$ and conversely, a problem that can be solved by $\mathsf{BPP}^{\mathsf{QNC}_{\mathcal{O}(1)}}$ but not by $\mathsf{QNC}_d^{\mathsf{BPP}}$. The former shows that having constant quantum depth with adaptive control cannot be simulated by repeating constant quantum depth machines without adaptive control. The latter does not seem to have as clear an interpretation. However, we can combine these ideas to construct another problem which also shows $\mathsf{BPP}^{\mathsf{QNC}_{\mathcal{O}(1)}^{\mathsf{BPP}}} \not\subseteq \mathsf{QNC}_d^{\mathsf{BPP}} \cup \mathsf{BPP}^{\mathsf{QNC}_d}$. giving further evidence that it is important to show soundness against $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$ when considering quantum depth because even with constant quantum depth, $\mathsf{BPP}^{\mathsf{QNC}_{\mathcal{O}(1)}^{\mathsf{BPP}}}$ already contains problems which are neither in $\mathsf{BPP}^{\mathsf{QNC}_d}$ nor in $\mathsf{QNC}_d^{\mathsf{BPP}}$. Therefore, it is crucial establishing that every proof of quantum depth is sound against $\mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$.

The results in this section are summarised in Table 6. Establishing $\mathsf{BPP}^{\mathsf{QNC}_d} \not\subseteq \mathsf{QNC}_d^{\mathsf{BPP}}$ is the most involved and requires the use of the compressed oracle simulation technique. We defer it to the end and instead first establish a general lifting theorem which takes almost any proof of quantumness and excludes it from $\mathsf{QNC}_d^{\mathsf{BPP}}$. We apply it to $\mathsf{CollisionHashing}$ to establish that $\mathsf{BPP}^{\mathsf{QNC}_{\mathcal{O}(1)}} \not\subseteq \mathsf{QNC}_d^{\mathsf{BPP}}$ in the random oracle model.

10 $\mathsf{BPP}^{\mathsf{QNC}_{\mathcal{O}(1)}} \not\subseteq \mathsf{QNC}_d^{\mathsf{BPP}}$

The idea behind d-$\mathsf{Ser}[\mathcal{P}]$ is quite intuitive. Suppose $\mathcal{P}$ is a problem which is specified by the relation R_H where H is the random oracle. For simplicity, suppose the input to the problem is 1^λ and $(1^\lambda, c) \in R_H$ means that c is a solution. Then d-$\mathsf{Ser}[\mathcal{P}]$ is a relation R'_H where $(1^\lambda, (c_0 \dots c_d)) \in R'_H$ if $(1^\lambda, c_0) \in R_H$, $(1^\lambda, c_1) \in R_{H(c_0\|\cdot)}$, $(1^\lambda, c_2) \in R_{H(c_0,c_1\|\cdot)}$ and so on. The rationale is that until the first problem is solved, the subsequent problems are not even specified. The problems must therefore be solved serially—they cannot be solved in parallel. So far, we have not constrained the model of computation. We want d-$\mathsf{Ser}[\mathcal{P}]$ to be hard for $\mathsf{QC_d}$ whenever $\mathcal{P}$ is hard for BPP (but can be solved by adding quantumness, e.g. in QNC_0). Recall that for d-$\mathsf{Rec}[\mathcal{P}]$ we wanted $\mathcal{P}$ to satisfy classical query soundness. In this case, we require $\mathcal{P}$ to satisfy a different property which we call *offline soundness*. Intuitively, suppose after running a classical algorithm to solve $\mathcal{P}$, access to H is revoked and thereafter unbounded computation is allowed. Offline soundness requires that even in this case, $\mathcal{P}$ cannot be solved with non-negligible probability.

The main difference between d-Ser and d-Rec is that in d-Ser one need not maintain "coherence" across all the problems (which use different oracles); it suffices to individually solve the problems. In d-Rec, even to access the oracle $\tilde{H} = H_d \circ \dots \circ H_0$, one had to maintain coherence across d layers.

10.1 Offline Soundness

We state offline soundness formally first.

Definition 88 (Offline Soundness)**.** As in Definition 28, let $H : \{0,1\}^* \to \{0,1\}$ be a random oracle. Define a problem $\mathcal{P}$ by a tuple $(\mathcal{X}, R_H)$ where $\mathcal{X}$ is a procedure which on input 1^λ generates a problem instance of size $\mathrm{poly}(\lambda)$ and $R_H = \{0,1\}^* \times \{0,1\}^*$ is a relation which depends on H. We define *offline soundness* as follows.

Let $\mathcal{A}^H$ be a PPT algorithm with access to H. Let $\tau[\mathcal{A}^H(x)]$ be the tableaux (or the computational transcript) obtained by running $\mathcal{A}^H$ on input $x \in \mathcal{X}$. Let $\mathcal{B}$ be an unbounded machine with no access to H which takes τ as input. We say $\mathcal{P}$ satisfies *offline soundness* if

$$\Pr_H\left[(x,y) \in R_H : \begin{array}{c} (x,y) \leftarrow \mathcal{B}(\tau) \\ \tau = \tau[\mathcal{A}^H(x)] \\ x \leftarrow \mathcal{X}(1^\lambda) \end{array}\right] \le \mathrm{negl}(\lambda)$$

for all $\mathcal{B}$ and $\mathcal{A}^H$.

Offline soundness is clearly a special case of classical query soundness and therefore both $\mathsf{CollisionHashing}$ and $\mathsf{CodeHashing}$ satisfy it.

Problem	$\in$		$\notin$	**Assumption**	**Verification**	**Interpretation**	**Remarks**
d-hCollisionHashing	$\mathsf{QNC}^{\mathsf{BPP}}_{\mathcal{O}(1)}$	$\nsubseteq$	$\mathsf{BPP}^{\mathsf{QNC}_d}$	RO	No	Even the simplest constant depth adaptive quantum control cannot be simulated by running a d depth quantum circuit poly many times.	Subsection 11.1
d-Ser[CollisionHashing]	$\mathsf{BPP}^{\mathsf{QNC}_{\mathcal{O}(1)}}$	$\nsubseteq$	$\mathsf{QNC}^{\mathsf{BPP}}_d$	RO	No	(Perhaps unsurprisingly) repeating a constant depth quantum circuit cannot be simulated with running a d adaptive quantum depth circuit once.	Subsection 10.5
d-Ser[d-hCollisionHashing]	$\mathsf{BPP}^{\mathsf{QNC}^{\mathsf{BPP}}_{\mathcal{O}(1)}}$	$\nsubseteq$	$\mathsf{BPP}^{\mathsf{QNC}_d} \cup \mathsf{QNC}^{\mathsf{BPP}}_d$	RO	No	Evidence that CQC_d is the right notion of depth.	Subsection 11.2

Table 6: A summary of the relations between $\mathsf{BPP}^{\mathsf{QNC}^{\mathsf{BPP}}_d}$, $\mathsf{QNC}^{\mathsf{BPP}}_d$ and $\mathsf{BPP}^{\mathsf{QNC}_d}$.

Lemma 89. CodeHashing *and* CollisionHashing *satisfy offline soundness.*

It appears reasonable to expect offline soundness to be a strictly weaker requirement than classical query soundness. Indeed, this is true because there are problems which satisfy offline soundness but not classical query soundness, e.g. the problem considered by [BKVV20].

10.2 The d-Ser[$\mathcal{P}$] Problem

With offline soundness in place, we can define d-Ser[$\mathcal{P}$] as follows.

Definition 90 (d-Ser[$\mathcal{P}$])**.** Let $\mathcal{P} = (\mathcal{X}, R)$ be a problem (see Definition 28) defined with respect to a random oracle $H : \{0,1\}^* \to \{0,1\}$, having a bounded oracle domain (as specified in Definition 28) and satisfying offline soundness (as defined in Definition 88).

Define d-Ser[$\mathcal{P}$] as follows: On input 1^λ, sample $d+1$ independent instances of $\mathcal{P}$ as $(x_0, \dots x_d)$ where $x_i \leftarrow \mathcal{X}(1^\lambda)$ for each $i \in \{0 \dots d\}$. Accept $(y_0, \dots y_d)$ if for each $i \in \{1 \dots d\}$, $(x_i, y_i) \in R_{H((x_0,y_0,\dots x_{i-1},y_{i-1})\|\cdot)}$ and for $i = 0$, $(x_0, y_0) \in R_{H(\cdot)}$.

10.3 Lower-bounds

In this section, we analyse everything for a fixed λ and introduce some notation to that end. Since $\mathcal{P}$ has bounded oracle domain, one can consider $d+1$ oracles with bounded domains which in turn make the analysis easier.

Notation 91. Fix a λ. Let $\mathcal{P} = (\mathcal{X}, R)$ be as in Definition 90 where the bounded oracle domain of $\mathcal{P}$ is $\mathcal{D} := \{0,1\}^{p(\lambda)}$. Fix an input instance $(x_0 \dots x_d)$ of d-Ser[$\mathcal{P}$]. With respect to this, let $\mathcal{S}_{i,H} := \{(x_i, y_i) : (x_i, y_i) \in R_H\}$ denote all pairs (x_i, y_i) in R_H. Let $\mathcal{S} := \mathcal{X} \times \mathcal{Y}$ where $\mathcal{Y}$ is the set of all ys. It would be useful to consider $d+1$ oracles with bounded domains instead of considering $H : \{0,1\}^* \to \{0,1\}$. More precisely, let $H_0 : \mathcal{D} \to \{0,1\}$, $H_1 : \mathcal{S} \times \mathcal{D} \to \{0,1\}, \dots H_d : \mathcal{S}^d \times \mathcal{D} \to \{0,1\}$. Let $\mathcal{L} := (H_0, \dots H_d)$ denote the sequence of oracles H_is.

It would be helpful to define the analogue of Definition 45 for our sequence of oracles.

Definition 92 (Shadow Oracles wrt $\bar{S}$ for d-Ser[$\mathcal{P}$])**.** Let $\mathcal{L} = (H_0 \dots H_d)$, p and $\mathcal{S}$ be as in Notation 91. Let $\bar{S} = (S_1, \dots S_d)$ be a tuple of d sets where each set $S_i \subseteq \mathcal{S}^i \times \{0,1\}^p$. The random shadow oracle $\mathcal{M}$ of $\mathcal{L}$ wrt $\bar{S}$ is defined as $\mathcal{M} := (M_0, \dots M_d)$ where for each $i \in \{1 \dots d\}$, M_i is the shadow oracle of H_i wrt S_i (as in Definition 45) and $M_0 = H_0$.

10.3.1 Exclusion from QNC_d

We first show that d-$\mathsf{Ser}[\mathcal{P}]$ is hard for QNC_d and then extend the analysis to $\mathsf{QC_d}$. To this end, we first introduce the shadow oracles by describing the sets we hide. Let $\mathcal{D} = \{0,1\}^p$ denote the oracle domain of $\mathcal{P}$.

- Define the hidden sets for $i, j \in \{1 \dots d\}$ as $S_{ij} \doteq$

$$\begin{bmatrix} \mathcal{S}_{H_0} \times \mathcal{D} & \mathcal{S}_{H_0} \times \mathcal{S} \times \mathcal{D} & \mathcal{S}_{H_0} \times \mathcal{S}^2 \times \mathcal{D} & \dots & \mathcal{S}_{H_0} \times \mathcal{S}^{d-1} \times \mathcal{D} \\ \varnothing & \bigcup_{(x_0 y_0) \in \mathcal{S}_0} (x_0 y_0) \times \mathcal{S}_{H_1(x_0, y_0 \| \cdot)} \times \mathcal{D} & \bigcup_{(x_0 y_0) \in \mathcal{S}_0} (x_0 y_0) \times \mathcal{S}_{H_1(x_0, y_0 \| \cdot)} \times \mathcal{S} \times \mathcal{D} & \dots & \bigcup_{(x_0 y_0) \in \mathcal{S}_0} (x_0 y_0) \times \mathcal{S}_{H_1(x_0, y_0 \| \cdot)} \times \mathcal{S}^{d-2} \times \mathcal{D} \\ \varnothing & \varnothing & \bigcup_{(x_0 y_0 x_1 y_1) \in \mathcal{S}_{0:1}} (x_0 y_0 x_1 y_1) \times \mathcal{S}_{H_2(x_0 y_0 x_1 y_1 \| \cdot)} \times \mathcal{D} & \dots & \bigcup_{(x_0 y_0 x_1 y_1) \in \mathcal{S}_{0:1}} (x_0 y_0 x_1 y_1) \times \mathcal{S}_{H_2(x_0 y_0 x_1 y_1 \| \cdot)} \times \mathcal{S}^{d-3} \times \mathcal{D} \\ & & & \ddots & \end{bmatrix}$$

$$= \begin{bmatrix} \mathcal{S}_{0:0} \times \mathcal{D} & \mathcal{S}_{0:0} \times \mathcal{S} \times \mathcal{D} & \mathcal{S}_{0:0} \times \mathcal{S}^2 \times \mathcal{D} & \dots & \mathcal{S}_{0:0} \times \mathcal{S}^{d-1} \times \mathcal{D} \\ \varnothing & \mathcal{S}_{0:1} \times \mathcal{D} & \mathcal{S}_{0:1} \times \mathcal{S} \times \mathcal{D} & \dots & \mathcal{S}_{0:1} \times \mathcal{S}^{d-2} \times \mathcal{D} \\ \varnothing & \varnothing & \mathcal{S}_{0:2} \times \mathcal{D} & \dots & \mathcal{S}_{0:2} \times \mathcal{S}^{d-3} \times \mathcal{D} \\ & & & \ddots & \end{bmatrix} \tag{18}$$

  where the union in the first matrix is over "correct solutions", i.e. (a) $\mathcal{S}_{0:0} =: \mathcal{S}_0 := \mathcal{S}_{H_0} = \mathcal{S}_H$, denotes the set of solutions (corresponding to λ) to $\mathcal{P}$ wrt H_0, (b) $\mathcal{S}_{0:1} = \cup_{(x_0, y_0) \in \mathcal{S}_0} (x_0, y_0) \times \mathcal{S}_{H_1(x_0 y_0 \| \cdot)}$, denotes the set of solutions to $\mathcal{P}$ wrt H_0 (in the first two coordinates) and corresponding to each solution, the set of solutions to $\mathcal{P}$ wrt $H_1(x_0 y_0 \| \cdot)$ (in the last two coordinates) and (c) in general $\mathcal{S}_{0:i} = \cup_{(x_0, y_0 \dots x_{i-1} y_{i-1}) \in \mathcal{S}_{0:i-1}} (x_0 y_0 \dots x_{i-1} y_{i-1}) \times \mathcal{S}_{H_i(x_0 \dots y_{i-1} \| \cdot)}$. By $\mathcal{S}_{H_i(s \| \cdot)}$ we mean $\mathcal{S}_{i, H_i(s \| \cdot)}$ where s is some string.

- We now try to justify this definition. The main structure of the proof is similar to the proof of QNC_d hardness of d-$\mathsf{CodeHashing}$, i.e. Lemma 53. Let $\bar{S}_i = (S_{i1}, \dots S_{id})$ denote the ith row of the matrix above. Let $\mathcal{M}_1$ denote the shadow of $\mathcal{L}$ wrt $\bar{S}_1$. As in the proof of Lemma 53, we want to ensure that the information contained in $\mathcal{M}_1$ is not enough to guess $\bar{S}_2$ which will be used to define $\mathcal{M}_2$. This would allow us to apply Lemma 42 as before. Once this is clear, the remaining steps are straightforward. Observe that $\mathcal{M}_1$ specifies H_0 and therefore (information theoretically) specifies $\mathcal{S}_{H_0}$. It also specifies H_1 partially—it does not specify H_1 on $\mathcal{S}_{H_0} \times \mathcal{D}$. Note that, in particular, this means that $\mathcal{M}_1$ contains no information about $\mathcal{S}_{H_1(x_0 y_0 \| \cdot)}$ for $(x_0 y_0) \in \mathcal{S}_{H_0} = \mathcal{S}_{0:0}$. That, in turn, means that none of the sets in $\bar{S}_2 | H_0$ are correlated with $\mathcal{M}_1 | H_0$.

- Let us look at the next case as well, as it would help with the general argument in the proof. Suppose $\mathcal{M}_2$ is the shadow of $\mathcal{L}$ wrt $\bar{S}_2$. We want to argue that even knowing $\mathcal{M}_2$ it is hard to find $\bar{S}_3$. Observe that $\mathcal{M}_2$ specifies H_0 and H_1. It also specifies H_2 partially—it does not specify H_2 at $\bigcup_{(x_0 y_0) \in \mathcal{S}_0} (x_0 y_0) \times \mathcal{S}_{H_1(x_0 y_0 \| \cdot)} \times \mathcal{D}$. This, in particular, means that $\mathcal{M}_2$ contains no information about $\mathcal{S}_{H_2(x_0 y_0 x_1 y_1 \| \cdot)}$ for $(x_0 y_0 x_1 y_1) \in \mathcal{S}_{0:1}$. That in turn means that none of the sets in $\bar{S}_3$ are correlated with $\mathcal{M}_2$, given H_0, H_1.

- Intuitively, $\mathcal{M}_{i-1}$ completely specifies $H_0, \dots H_{i-2}$ but it does not specify H_{i-1} completely and $\bar{S}_i$ (conditioned on $H_0 \dots H_{i-2}$) depends only on this unspecified part of H_{i-1}.

Algorithm 93. *Let $\mathcal{L} = (H_0 \dots H_d)$, $\mathcal{S}$, $\mathcal{S}_{i,H}$ and p be as in Notation 91. Assume λ and the input instance $(x_0, \dots x_d)$ have been implicitly specified. We use $\mathcal{S}_{H_i(s \| \cdot)}$ to denote $\mathcal{S}_{i, H_i(s \| \cdot)}$. Define, for each $i \in \{1, \dots d\}$, S_{ij} as follows.*

1. *If $i = 1$, define $S_{1j} := \mathcal{S}_{H_0} \times \mathcal{S}^{j-1} \times \mathcal{D}$*
2. *If $i > 1$,*
   - *(a) Define $S_{ij} := \varnothing$ for $1 \leq j < i$ and*
   - *(b) otherwise,*

$$\begin{aligned} S_{ij} &:= \bigcup_{(x_0 y_0 \dots x_{i-2} y_{i-2}) \in \mathcal{S}_{0:i-2}} (x_0 y_0 \dots x_{i-2} y_{i-2}) \times \mathcal{S}_{H_{i-1}(x_0 y_0 \dots x_{i-2} y_{i-2} \| \cdot)} \times \mathcal{S}^{i-j} \times \mathcal{D} \\ &= \mathcal{S}_{0:i-1} \times \mathcal{S}^{i-j} \times \mathcal{D} \end{aligned}$$

where

$$\mathcal{S}_{0:i} = \begin{cases} \mathcal{S}_0 := \mathcal{S}_{H_0} & i = 0 \\ \bigcup_{(x_0, y_0 \dots x_{i-1} y_{i-1}) \in \mathcal{S}_{0:i-1}} (x_0 y_0 \dots x_{i-1} y_{i-1}) \times \mathcal{S}_{H_i(x_0 \dots y_{i-1} \| \cdot)} & i > 0. \end{cases}$$

Let $\bar{S}_i := (S_{i1}, \dots S_{id})$. *Return* $(\bar{S}_1, \dots \bar{S}_d)$.

Lemma 94 (d-$\mathsf{Ser}[\mathcal{P}] \notin \mathsf{QNC}_d$). *Every* QNC_d *circuit succeeds at solving* d-$\mathsf{Ser}[\mathcal{P}]$ *(see Definition 90) with probability at most* $\mathrm{negl}(\lambda)$ *on input* 1^λ *for* $d \le \mathrm{poly}(\lambda)$.

Proof. Fix a λ. Let $\mathcal{L} = (H_0, \dots H_d)$ be as in Notation 91. Suppose the problem instance d-$\mathsf{Ser}[\mathcal{P}]$ is specified by $(x_0, \dots x_d)$ and let ρ_0 be the initial state, containing this input. Denote by $\mathcal{A}^{\mathcal{L}}$ an arbitrary QNC_d circuit

$$\mathcal{A}^{\mathcal{L}}(\rho_0) := \Pi_{\text{valid}} \circ U_{d+1} \circ \mathcal{L} \circ U_d \circ \dots \mathcal{L} \circ U_2 \circ \mathcal{L} \circ U_1 \circ \rho_0$$

where Π_{valid} corresponds to projection on all output strings which solve d-$\mathsf{Ser}[\mathcal{P}]$ (for a fixed λ). let $(\bar{S}_1, \dots \bar{S}_d)$ be the output of Algorithm 93. Define

$$\mathcal{A}^{\mathcal{M}}(\rho_0) := \Pi_{\text{valid}} \circ U_{d+1} \circ \mathcal{M}_d \circ U_d \dots \mathcal{M}_2 \circ U_2 \circ \mathcal{M}_1 \circ U_1 \circ \rho_0$$

where $\mathcal{M}_i$ is the random shadow oracle of $\mathcal{L}$ wrt $\bar{S}_i$ (see Definition 92). We proceed in two steps.

Step 1: *$\mathcal{A}^{\mathcal{L}}$ and $\mathcal{A}^{\mathcal{M}}$ behave the same.* We show that the probability that $\mathcal{A}^{\mathcal{L}}$ and $\mathcal{A}^{\mathcal{M}}$ produce a valid output is negligibly close, i.e. we bound

$$\begin{aligned}&|\mathrm{tr}[\Pi_{\text{valid}} \circ U_{d+1} \circ \mathcal{L} \circ U_d \circ \dots \mathcal{L} \circ U_2 \circ \mathcal{L} \circ U_1 \circ \rho_0] - \mathrm{tr}[\Pi_{\text{valid}} \circ U_{d+1} \circ \mathcal{M}_d \circ U_d \dots \mathcal{M}_2 \circ U_2 \circ \mathcal{M}_1 \circ U_1 \circ \rho_0]| \\ &\le \sum_{i=1}^{d} B[\mathcal{L} \circ U_i(\rho_{i-1}), \mathcal{M}_i \circ U_i(\rho_{i-1})] \le \sum_{i=1}^{d} \sqrt{2\,\mathrm{Pr}[\mathrm{find} : U_i^{\mathcal{L}\backslash \bar{S}_i}, \rho_{i-1}]}\end{aligned}$$

where $\rho_i = \mathcal{M}_i \circ U_i \circ \dots \mathcal{M}_1 \circ U_1 \circ \rho_0$ for $i > 0$, we used (as in the proof of Lemma 53) the triangle inequality, monotonicity of the trace distance, the relation between trace distance and Bures distance and finally applied Lemma 41. To bound the RHS above, one can use Lemma 42 if it holds that ρ_{i-1} is uncorrelated with $\bar{S}_i$. It suffices to show that $\mathcal{M}_{i-1}$ is uncorrelated with $\bar{S}_i$, given $H_0, \dots H_{i-2}$. We argued the $i = 1, 2$ case above. In general, for $i > 2$ (for notational ease), observe that $\mathcal{M}_{i-1}$ specifies $H_0, \dots H_{i-2}$ completely and specifies H_{i-1} at all points except at $\bigcup_{(x_0 y_0 \dots x_{i-3} y_{i-3}) \in \mathcal{S}_{0:i-3}} (x_0 y_0 \dots x_{i-3} y_{i-3}) \times \mathcal{S}_{H_{i-2}(x_0 y_0 \dots x_{i-3} y_{i-3} \| \cdot)} \times \mathcal{D} = \mathcal{S}_{0:i-2} \times \mathcal{D}$. This in particular means that $\mathcal{M}_{i-1}$ contains no information about $\mathcal{S}_{H_{i-1}(x_0 y_0 \dots x_{i-2} y_{i-2} \| \cdot)}$ for $(x_0 y_0 \dots x_{i-2} y_{i-2}) \in \mathcal{S}_{0:i-2}$ where $\{\mathcal{S}_{0:i}\}_i$ are as defined in Algorithm 93. This, in turn, entails that $\bar{S}_i$ is uncorrelated with $\mathcal{M}_{i-1}$, given $H_0, \dots H_{i-2}$ as asserted. From offline soundness of $\mathcal{P}$, and the aforesaid, it follows that

$$\mathrm{Pr}\left[(x_i, y_i) \in R_{H_i(x_0 y_0 \dots x_{i-1} y_{i-1} \| \cdot)} \middle| \mathcal{M}_{i-1} : \substack{H \xleftarrow{\$} \mathrm{Funcs}[\{0,1\}^* \to \{0,1\}] \\ (x_0, \dots x_i) \leftarrow \mathcal{X}(1^\lambda)}\right] \le \mathrm{negl}(\lambda)$$

for all $(x_0 y_0 \dots x_{i-1} y_{i-1}) \in \mathcal{S}_{0:i-1}$. This entails that, for $i \le k \le d$, $\mathrm{Pr}[(x_0 y_0 \dots x_k y_k) \in S_{ik} | \mathcal{M}_{i-1}] \le \mathrm{negl}(\lambda)$ which means, via Lemma 42, $\mathrm{Pr}[\mathrm{find} : U_i^{\mathcal{L}\backslash \bar{S}_i}, \rho_{i-1} | \mathcal{M}_{i-1}] \le \mathrm{negl}(\lambda)$ (the conditioning notation for $\mathrm{Pr}[\mathrm{find} : \dots]$ is the same as the last bullet after the $\mathsf{CQ_d}$ Lemma 74; all variables involved $(\rho_{i-1}, \bar{S}_i, \mathcal{L})$ are conditioned on $\mathcal{M}_{i-1}$).

Step 2: *$\mathcal{A}^{\mathcal{M}}$ cannot succeed with non-negligible probability.* Note that by construction, $\mathcal{M}_d$ contains all the information in $\mathcal{M}_1 \dots \mathcal{M}_{d-1}$. Further, observe that $\mathcal{M}_d$ does not contain any information about $\mathcal{S}_{H_d(x_0 y_0 \dots x_{d-1} y_{d-1} \| \cdot)}$ for $(x_0 y_0 \dots x_{d-1} y_{d-1}) \in \mathcal{S}_{0:d-1}$ (which includes the set of valid answers until $d-1$). From offline soundness of $\mathcal{P}$, it follows that $\mathcal{A}^{\mathcal{M}}$ cannot find $(x_d, y_d) \in \mathcal{S}_{H_d(x_0 \dots y_{d-1} \| \cdot)}$ with probability greater than $\mathrm{negl}(\lambda)$ which upper bounds the success probability of $\mathcal{A}^{\mathcal{M}}$. □

10.3.2 Exclusion from $\mathsf{QNC}_d^{\mathsf{BPP}}$

Recall that $\mathsf{QC_d}$ circuits are represented as $\mathcal{B}^{\mathcal{L}} = \mathcal{A}^{\mathcal{L}}_{c,d+1} \circ \mathcal{B}^{\mathcal{L}}_d \circ \dots \mathcal{B}^{\mathcal{L}}_1 \circ \rho_0$ where $\mathcal{B}^{\mathcal{L}}_i := \Pi_i \circ \mathcal{L} \circ U_i \circ \mathcal{A}^{\mathcal{L}}_{c,i}$. Here $\mathcal{A}^{\mathcal{L}}_{c,i}$ represent classical algorithms and we drop "c" in this section. Since the oracles $H_0, \dots H_d$ have different domains, we make the following assumption about the classical algorithms in the $\mathsf{QC_d}$ circuit. This simplifies our analysis and only makes our impossibility result stronger.

Assumption: if H_k is queried at $(x_0 y_0 \dots x_k y_k)$, then for all $i \in \{0 \dots k-1\}$ H_i is also queried at $(x_0 y_0 \dots x_i y_i)$.

It would also be helpful to setup some notation for describing the classical queries. Since $\mathcal{A}$ makes queries on different domains, the set of queries is simply a collection of strings with varying number of "coordinates". For example, if H_k is queried at $(x_0 y_0 \dots x_k y_k)$, by the jth coordinate we would mean $(x_j y_j)$.

Suppose T abstractly denotes all the queries made by a classical algorithm $\mathcal{A}$. We use $XY_{i:k}(T)$ to denote all the tuples $(x_i y_i \dots x_k y_k)$ queried by $\mathcal{A}_c$ from the ith to kth coordinate. We use $XY_i(T)$ to denote $XY_{i:i}(T)$, i.e. pairs $(x_i y_i)$ queried by T at the ith coordinate. In the following, when not explicitly stated, we assume the security parameter is fixed to be λ and assume that the problem instance for $\boldsymbol{d}\text{-}\mathsf{Ser}[\mathcal{P}]$ is specified by $(x_0, \ldots x_d)$.

As before, we use Notation 91 below.

- The following simple but crucial observation will be used repeatedly in our analysis. It adapts offline soundness to our setting.
  - Let $x \leftarrow \mathcal{X}(1^\lambda)$. Let $\mathcal{A}^\mathcal{L}$ be a PPT algorithm (trying to solve $\mathcal{P}$, i.e. finding an $(x,y) \in \mathcal{S}_{H_0}$). Run $\mathcal{A}^H(x)$ and denote its query transcript by T.
  - Let E denote the event that $XY_0(T) \cap S_{H_0} = \varnothing$.
  - Fix any $y \in \mathcal{Y}$. The assertion is that for any fixed (x,y), it holds that $\Pr[(x,y) \in \mathcal{S}_{H_0} | T \wedge E] \le \text{negl}(\lambda)$ when $\mathcal{A}^H$ is executed.
    - Suppose the assertion is false. Then for some (x,y) it holds that $\Pr[(x,y) \in \mathcal{S}_{H_0} | T \wedge E] \ge \mu(\lambda)$ where μ is noticeable (i.e. non-negligible).
    - If E does not happen, then T already contains some $(x,y') \in \mathcal{S}_{H_0}$. If E does happen, then $(x,y) \in \mathcal{S}_{H_0}$ with non-vanishing probability (as stated above). In both cases, an element in $\mathcal{S}_{H_0}$ is found with non-negligible probability. However, this violates offline soundness of $\mathcal{P}$.
- Let us build some intuition by starting with a simple circuit of the form $\mathcal{L} \circ U_1 \circ \mathcal{A}_1^\mathcal{L}$ (on input $(x_0 \ldots x_d)$) and comparing it to $\mathcal{M}_1 \circ U_1 \circ \mathcal{A}_1^\mathcal{L}$ where $\mathcal{M}_1$ is going to be the shadow of $\mathcal{L}$ wrt some sequence of sets $\bar{S}_1$. Take $\bar{S}_1$ to be as in the QNC_d case, i.e. $\bar{S}_1 = (\mathcal{S}_{H_0} \times \mathcal{D}, \mathcal{S}_{H_0} \times \mathcal{S} \times \mathcal{D}, \ldots)$. Let T_1 denote the queries made by $\mathcal{A}_1^\mathcal{L}$. If $XY_0(T_1)$ contains any pair $(x_0, y_0) \in \mathcal{S}_{H_0}$, then $\mathcal{L}$ and $\mathcal{M}_1$ can be distinguished (i.e. $\mathcal{L} \circ U_1 \circ \mathcal{A}_1^\mathcal{L}$ and $\mathcal{M}_1 \circ U_1 \circ \mathcal{A}_1^\mathcal{L}$ can behave differently) because they can be queried at $\bar{S}_1$ (which is precisely where $\mathcal{L}$ and $\mathcal{M}_1$ behave differently). Denote by E_1 the event that $XY_0(T_1) \cap \mathcal{S}_{H_0} = XY_0(T_1) \cap \mathcal{S}_{0:0} = \varnothing$. Using the fact that $\mathcal{P}$ satisfies offline soundness (in fact just from soundness against PPT machines), one has that $\Pr[\neg E_1] \le \text{negl}(\lambda)$. From the discussion above, it is also clear that offline soundness ensures
$$\Pr[(x_0, y_0) \in \mathcal{S}_{H_0} | T_1 E_1] \le \text{negl}(\lambda) \tag{19}$$
for all $y_0 \in \mathcal{Y}$. To apply Lemma 42 we would need to ensure that the state received by U_1 is independent of $\bar{S}_1$. Conditioned on T_1 and E_1, this is clearly the case (conditioning only reduces polynomially many possible values of $\mathcal{S}_{H_0}$). Further, the probability of finding an element in $\bar{S}_1$ is negligible because of Equation (19).
- This argument can also be applied to $\mathcal{L} \circ U_2 \circ \mathcal{A}_2^\mathcal{L} \circ \rho_1$ and $\mathcal{M}_2 \circ U_2 \circ \mathcal{A}_2^\mathcal{L} \circ \rho_1$ where $\rho_1 = \mathcal{M}_1 \circ U_1 \circ \mathcal{A}_1^\mathcal{L}(\rho_0)$, once we appropriately condition the variables. It may help to look at the first matrix Equation (18). $\mathcal{M}_1$ corresponds to the first row. We argue that $\mathcal{M}_1$ does not specify H_1 at $\mathcal{S}_{H_0} \times \mathcal{D}$ and since the second row, i.e. $\bar{S}_2$, depends on precisely the values of H_1 on $\mathcal{S}_{H_0} \times \mathcal{D}$, knowing ρ_1 does not help in determining $\bar{S}_1$. This is the same as the QNC_d case. To account for the classical algorithm, we simply condition $\mathcal{A}_1^\mathcal{L}$ on not querying H_0 inside $\mathcal{S}_{H_0}$ and $\mathcal{A}_2^\mathcal{L}$ on not querying H_1 inside $\mathcal{S}_{H_1}$ (or more precisely, inside $\mathcal{S}_{0:1}$). The previous argument then goes through unchanged.
  We now make this reasoning more precise. Let us condition on the event E_1. Then, it is clear that ρ_1 contains no information about $H_1(x_0, y_0 \| \cdot)$ for any $(x_0, y_0) \in \mathcal{S}_{H_0}$. This is because, by definition of E_1, the classical algorithm $\mathcal{A}_1^\mathcal{L}$ never accessed H_1 on the said domain, and $\mathcal{M}_1$ contains no information about H_1 on that domain (by definition of $\bar{S}_1$). (Note that information about H_0 was present in $\mathcal{M}_1$ and therefore, information theoretically, $\mathcal{S}_{H_0}$ could have been determined.) Let T_2 denote the queries made by $\mathcal{A}_2^\mathcal{L}$ in the circuits above and denote by E_2 the event that $XY_{0:1}(T_2) \cap \mathcal{S}_{0:1} = \varnothing$, i.e. the query transcript so far does not contain a solution to $\mathcal{P}$ corresponding to $H_1(x_0 y_0 \| \cdot)$ for any $(x_0, y_0) \in \mathcal{S}_{H_0}$. Again, from (offline) soundness of $\mathcal{P}$, it follows that $\Pr[\neg E_2 | E_1] \le \text{negl}(\lambda)$, i.e. $\mathcal{A}_2^\mathcal{L}$ solves $\mathcal{P}$ corresponding to $H_1(x_0 y_0 \| \cdot)$ for $(x_0 y_0) \in \mathcal{S}_0$ given E_1, because $\mathcal{A}_2^\mathcal{L}$ does not learn anything about $H_1(x_0 y_0 \| \cdot)$ for $(x_0 y_0) \in \mathcal{S}_0$ and E_1 guarantees $\mathcal{A}_1^\mathcal{L}$ did not even query at $(x_0 y_0) \in \mathcal{S}_0$. Given $T_2, E_2, E_1, \mathcal{M}_1$, from offline soundness of $\mathcal{P}$ corresponding to $H_1(x_0 y_0 \| \cdot)$, it holds that
$$\Pr[(x_1, y_1) \in \mathcal{S}_{H_1(x_0 y_0 \| \cdot)} | T_2 T_1 E_2 E_1 \mathcal{M}_1] \le \text{negl}(\lambda) \tag{20}$$
for all $y_1 \in \mathcal{Y}$ and $(x_0, y_0) \in \mathcal{S}_{0:0} = \mathcal{S}_{H_0}$ follow. To apply Lemma 42 one needs to ensure that $\mathcal{A}_2^\mathcal{L} \circ \rho_1$ is independent of $\bar{S}_2$. Conditioning on $v_2 := (T_2 T_1 E_2 E_1 \mathcal{M}_1)$, it is clear that $\mathcal{A}_2^\mathcal{L} \circ \rho_1$ contains no information

about $H_1(x_0y_0\|\cdot)$ for $(x_0y_0) \in \mathcal{S}_{0:0}$. Further, $\bar{S}_2$, conditioned on v_2 only depends on $H_1(x_0y_0\|\cdot)$ for $(x_0y_0) \in \mathcal{S}_{0:0}$ (again, excluding the values in T_2). Therefore, $\bar{S}_2|v_2$ and $\mathcal{A}_2^{\mathcal{L}} \circ \rho_1|v_2$ are uncorrelated. Finally, the probability of finding an element in $\bar{S}_2|v_2$ is negligible due to Equation (20).

This readily generalises and accounting for these arguments in the QNC_d case yields the following.

Lemma 95 (d-$\mathsf{Ser}[\mathcal{P}] \notin \mathsf{QNC}_d^{\mathsf{BPP}}$)**.** *Every* $\mathsf{QC_d}$ *circuit succeeds at solving* d-$\mathsf{Ser}[\mathcal{P}]$ *(see Definition 90) with probability at most* $\mathrm{negl}(\lambda)$ *on input* 1^λ *for* $d \le \mathrm{poly}(n)$.

Proof. Fix a λ. Let $\mathcal{L} = (H_0, \dots H_d)$ be as in Notation 91. Suppose the problem instance d-$\mathsf{Ser}[\mathcal{P}]$ is specified by $(x_0 \dots x_d)$ and let ρ_0 be the initial state. Denote by $\mathcal{B}^{\mathcal{L}}$ an arbitrary $\mathsf{QC_d}$ circuit

$$\mathcal{B}^{\mathcal{L}}(\rho_0) \coloneqq \Pi_{\text{valid}} \circ \mathcal{A}_{d+1}^{\mathcal{L}} \circ \mathcal{B}_d^{\mathcal{L}} \circ \dots \mathcal{B}_1^{\mathcal{L}} \circ \rho_0$$

where $\mathcal{B}_i^{\mathcal{L}} \coloneqq \Pi_i \circ \mathcal{L} \circ U_i \circ \mathcal{A}_i^{\mathcal{L}}$ and Π_{valid} corresponds to projection on all output strings which solve d-$\mathsf{Ser}[\mathcal{P}]$. Let $(\bar{S}_1 \dots \bar{S}_d)$ be the output of Algorithm 93. Define

$$\mathcal{B}^{\mathcal{M}}(\rho_0) \coloneqq \Pi_{\text{valid}} \circ \mathcal{A}_{d+1}^{\mathcal{L}} \circ \mathcal{B}_d^{\mathcal{M}} \circ \dots \circ \mathcal{B}_1^{\mathcal{M}} \circ \rho_0 \tag{21}$$

where $\mathcal{B}_i^{\mathcal{M}} \coloneqq \Pi_i \circ \mathcal{M}_i \circ U_i \circ \mathcal{A}_i^{\mathcal{L}}$ and $\mathcal{M}_i$ is the shadow oracle of $\mathcal{L}$ wrt $\bar{S}_i$ (see Definition 92). We proceed in two steps.

Step 1: $\mathcal{B}^{\mathcal{L}}$ *and* $\mathcal{A}^{\mathcal{M}}$ *behave the same.* We show that the probability that $\mathcal{A}^{\mathcal{L}}$ and $\mathcal{A}^{\mathcal{M}}$ produce a valid output is negligibly close, i.e. we bound

$$\begin{aligned}&\left|\mathrm{tr}[\Pi_{\text{valid}} \circ \mathcal{A}_{d+1}^{\mathcal{L}} \circ \mathcal{B}_{d+1}^{\mathcal{L}} \circ \dots \mathcal{B}_1^{\mathcal{L}} \circ \rho_0 - \Pi_{\text{valid}} \circ \mathcal{A}_{d+1}^{\mathcal{L}} \circ \mathcal{B}_{d+1}^{\mathcal{M}} \circ \dots \mathcal{B}_1^{\mathcal{M}} \circ \rho_0]\right| \\ &\le \sum_{i=1}^{d} B(\mathcal{B}_i^{\mathcal{L}}(\rho_{i-1}), \mathcal{B}_i^{\mathcal{M}}(\rho_{i-1})) \le \sum_{i=1}^{d} \sqrt{2\Pr[\text{find} : U_i^{\mathcal{L}\setminus\bar{S}_i}, \mathcal{A}_i^{\mathcal{L}} \circ \rho_{i-1}]}\end{aligned}$$

where for $i \in \{1, 2 \dots d-1\}$, $\rho_i \coloneqq \mathcal{B}_i^{\mathcal{M}} \circ \dots \mathcal{B}_1^{\mathcal{M}} \circ \rho_0$, proceeding as in the QNC_d case. To bound the RHS above, one can use Lemma 42 if it holds that $\mathcal{A}_i^{\mathcal{L}} \circ \rho_{i-1}$ is uncorrelated with $\bar{S}_i$, upon appropriate conditioning. Let $v_i \coloneqq (T_i \dots T_1 E_i \dots E_1 \mathcal{M}_{i-1})$ denote the random variables we condition on, where T_i is the transcript of queries made by $\mathcal{A}_i^{\mathcal{L}}$, E_i is the event that $XY_{0:i-1}(T_i) \cap \mathcal{S}_{0:i-1} = \varnothing$, i.e. the transcript does not contain a solution to $\mathcal{P}$ corresponding to $H_{i-1}(x_0y_0 \dots x_{i-2}y_{i-2}\|\cdot)$ for any $(x_0y_0 \dots x_{i-2}y_{i-2}) \in \mathcal{S}_{0:i-2}$ ($\mathcal{S}_{i:j}$ are as in Algorithm 93) and $\mathcal{M}_{i-1}$ is the shadow oracle wrt $\bar{S}_{i-1}$ and contains all the information in $\mathcal{M}_1 \dots \mathcal{M}_{i-2}$. As argued above, it is the case that $\mathcal{A}_i^{\mathcal{L}} \circ \rho_{i-1}|v_i$ is uncorrelated with $\bar{S}_i|v_i$ because no classical query has been made to $H_{i-1}(x_0y_0 \dots x_{i-2}y_{i-2}\|\cdot)$ for $(x_0y_0 \dots x_{i-2}y_{i-2}) \in \mathcal{S}_{0:i-2}$ and all previous shadow oracles, $\mathcal{M}_1 \dots \mathcal{M}_{i-1}$ output $\perp$ on the aforesaid domain of H_{i-1} while $\bar{S}_i$ conditioned on v_i depends only on H_{i-1} at the aforementioned domain. It remains to bound the probability of finding an element in $\bar{S}_i|v_i$. To this end, note that given v_i, from the offline soundness of $\mathcal{P}$ corresponding to $H_{i-1}(x_0y_0 \dots x_{i-2}y_{i-2}\|\cdot)$, it follows that

$$\Pr[(x_i, y_i) \in \mathcal{S}_{H_i(x_0 \dots y_{i-2}\|\cdot)}|v_i] \le \mathrm{negl}(\lambda)$$

for all $y_i \in \mathcal{Y}$ and $(x_0 \dots y_{i-2}) \in \mathcal{S}_{0:i-2}$. This entails that for $i \le k \le d$, $\Pr[(x_0y_0 \dots x_ky_k) \in S_{ik}|v_i] \le \mathrm{negl}(\lambda)$. (Offline) soundness of $\mathcal{P}$ also implies that $\Pr[\neg E_i|E_1 \dots E_{i-1}] \le \mathrm{negl}(\lambda)$. Together, these yield $\Pr[\text{find} : U_i^{\mathcal{L}\setminus\bar{S}_i}, \mathcal{A}_i^{\mathcal{L}} \circ \rho_{i-1}] \le \mathrm{negl}(\lambda)$.

Step 2: $\mathcal{B}^{\mathcal{M}}$ *cannot succeed with non-negligible probability.* Consider $\mathcal{B}^{\mathcal{M}}$ as in Equation (21) and let E_i and v_i be as defined above. Since $\Pr[\neg E_i|E_{i-1} \dots E_1] \le \mathrm{negl}(\lambda)$, it holds that $\Pr[E_1 \dots E_d] \ge 1 - \mathrm{negl}(\lambda)$. Conditioned on $E_1 \dots E_d$, note that $\mathcal{M}_1 \dots \mathcal{M}_d$ do not specify $H_d(x_0 \dots y_{d-1}\|\cdot)$ for $(x_0 \dots y_{d-1}) \in \mathcal{S}_{0:d-1}$. Therefore, ρ_d also does not specify $H_d(x_0 \dots y_d\|\cdot)$ in the aforesaid domain. From (offline) soundness of $\mathcal{P}$, it follows that $\mathcal{A}_{d+1}^{\mathcal{L}}(\rho_d)|v_d$ outputs a solution to $\mathcal{P}$ corresponding to $H_d(x_0 \dots y_d\|\cdot)$ is negligible. Together, these yield $\Pr[s \in \mathcal{S}_{0:d} : s \leftarrow \mathcal{B}^{\mathcal{M}}] \le \mathrm{negl}(\lambda)$ proving the assertion.

□

10.4 Upper-bounds

If $\mathcal{P}$ can be solved using $\mathsf{QNC}_{d'}$, then it is evident that for any $d \le \mathrm{poly}(\lambda)$, d-$\mathsf{Ser}[\mathcal{P}]$ can be solved in $\mathsf{CQ_{d'}}$. One simply solves $\mathcal{P}$ corresponding to H_0 using the first $\mathsf{QNC}_{d'}$ circuit in $\mathsf{CQ_{d'}}$, then using this result, solves $\mathcal{P}$ corresponding to H_1 and so son. Since $d \le \mathrm{poly}(\lambda)$, it follows that $\mathsf{CQ_{d'}}$ is sufficient to solve the problem. Similarly, if $\mathcal{P}$ can be solved in $\mathsf{QC_{d'}}$, then d-$\mathsf{Ser}[\mathcal{P}]$ can be solved in $\mathsf{CQC}_{d'}$. This yields the following.

Lemma 96 ($\mathcal{P} \in \mathsf{QNC}_{d'} \implies d\text{-}\mathsf{Ser}[\mathcal{P}] \in \mathsf{BPP}^{\mathsf{QNC}_{d'}}$ and $\mathcal{P} \in \mathsf{QNC}_{d'}^{\mathsf{BPP}} \implies d\text{-}\mathsf{Ser}[\mathcal{P}] \in \mathsf{BPP}^{\mathsf{QNC}_{d'}^{\mathsf{BPP}}}$)**.** *Let $\mathcal{P}$ be a problem (see Definition 28) which can be solved in $\mathsf{QNC}_{d'}$ (resp. $\mathsf{QC}_{\mathsf{d}'}$). Then, for any $d \leq \mathrm{poly}(\lambda)$, it holds that d-$\mathsf{Ser}[\mathcal{P}]$ (see Definition 90) can be solved in $\mathsf{CQ}_{\mathsf{d}'}$ (resp. $\mathsf{CQC}_{d'}$).*

10.5 Consequences

Theorem 97. *Fix any $d \leq \mathrm{poly}(n)$. Then, with respect to a random oracle, it holds that $\mathsf{BPP}^{\mathsf{QNC}_{\mathcal{O}(1)}} \not\subseteq \mathsf{QNC}_d^{\mathsf{BPP}}$.*

Proof. Recall $\mathsf{CollisionHashing}$ from Definition 85. One has that (using Definition 90) d-$\mathsf{Ser}[\mathsf{CollisionHashing}] \in \mathsf{BPP}^{\mathsf{QNC}_{\mathcal{O}(1)}}$ using the fact that $\mathsf{CollisionHashing} \in \mathsf{QNC}_{\mathcal{O}(1)}$ (see Lemma 86) and Lemma 96 with $d' = \mathcal{O}(1)$. One also has that d-$\mathsf{Ser}[\mathsf{CollisionHashing}] \notin \mathsf{QNC}_d^{\mathsf{BPP}}$ because $\mathsf{CollisionHashing}$ satisfies all properties required of $\mathcal{P}$ in the definition of d-$\mathsf{Ser}[\mathcal{P}]$ (see Definition 90, Lemma 86, Lemma 89) and therefore Lemma 95 applies, yielding the asserted exclusion. □

The rest of this article is dedicated to establishing d-$\mathsf{hCollisionHashing}$ is not in $\mathsf{BPP}^{\mathsf{QNC}_d}$. Using d-$\mathsf{Ser}[d\text{-}\mathsf{hCollisionHashing}]$ one immediately obtains the separation, $\mathsf{BPP}^{\mathsf{QNC}_d} \cup \mathsf{QNC}_d^{\mathsf{BPP}} \not\subseteq \mathsf{BPP}^{\mathsf{QNC}_d^{\mathsf{BPP}}}$.

11 $\mathsf{QNC}_{\mathcal{O}(1)}^{\mathsf{BPP}} \not\subseteq \mathsf{BPP}^{\mathsf{QNC}_d}$

In this section, we define the problem d-$\mathsf{hCollisionHashing}$, which is a variant of $\mathsf{CollisionHashing}$. This problem shows that $\mathsf{QNC}_{\mathcal{O}(1)}^{\mathsf{BPP}} \not\subseteq \mathsf{BPP}^{\mathsf{QNC}_d}$, relative to a random oracle.

11.1 The Problem

Some notation before we proceed: for $d \in \mathbb{N}$ and $\Sigma \subset \{0,1\}^*$ define $h := H_d \circ \cdots \circ H_1 \circ H_0$ where $H_0 : \Sigma \to \Sigma^{d'}$, for $j \in \{1, \ldots d-1\}$, $H_j : \Sigma^{d'} \to \Sigma^{d'}$ and $H_d : \Sigma^{d'} \to \Sigma$ are independent random oracles with $d' = 2d + 5$.

Definition 98 (d-$\mathsf{hCollisionHashing}$ or simply $\mathsf{Problem}$)**.** Let $d : \mathbb{N} \to \mathbb{N}$, and[66] $C = 1/(2(e^2 - 1))$. The d-$\mathsf{hCollisionHashing}$ problem is defined as follows. Let λ denote be a security parameter for the problem. Consider the following oracles.

- $G_0, G_1 : \{0,1\}^\lambda \to \{0,1\}^\lambda$ is a random oracle with domain twice as large as co-domain.
- $h : \{0,1\}^\lambda \to \{0,1\}^\lambda$ is a composition of $d(\lambda) + 1$ random oracles (as described above with $\Sigma = \{0,1\}^\lambda$).
- $H : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}$ is a random oracle with one-bit output.

Let $\mathsf{TwoToOne}(G_0, G_1) := \{y \in \{0,1\}^\lambda : |G_0^{-1}(y)| = |G_1^{-1}(y)| = 1\}$. Then, the d-$\mathsf{hCollisionHashing}$ problem (later referred to simply as $\mathsf{Problem}$) is, given access to the oracles $G_0, G_1, H, H_0, \ldots H_d$ (but not to h directly) return $(y_1, \ldots, y_\lambda)$, $(r_1, \ldots, r_\lambda)$, and $(m_1, \ldots, m_\lambda)$ such that the following conditions are satisfied.

- All y_i's are distinct.
- Let $\mathcal{I} = \{i : y_i \in \mathsf{TwoToOne}(G_0, G_1)\}$. Then, $|\mathcal{I}| \geq \frac{3}{4} C\lambda$.
- Let $\mathcal{I}_{\mathsf{win}} = \{i : y_i \in \mathsf{TwoToOne}(G_0, G_1)$ and $r_i \cdot (x_0^{y_i} \oplus x_1^{y_i}) \oplus H(x_0^{y_i}, h(y_i)) \oplus H(x_1^{y_i}, h(y_i)) = m_i\}$, where $x_0^{y_i}$ and $x_1^{y_i}$ are the pre-images of y_i. Then $|\mathcal{I}_{\mathsf{win}}| \geq 3|\mathcal{I}|/4$.

It is helpful to also consider a "single-copy" version of d-$\mathsf{hCollisionHashing}$, that we refer to as $\mathsf{subProblem}$ and define as follows Given the same oracles as in d-$\mathsf{hCollisionHashing}$, output (y, r, m) such that, $y \in \mathsf{TwoToOne}(G)$ and $r \cdot (x_0^y \oplus x_1^y) \oplus H(x_0^y, h(y)) \oplus H(x_1^y, h(y)) = m$, where x_0^y and x_1^y are the pre-images of y under G_0 and G_1 respectively. We call such a (y, r, m) a "valid equation".

From Lemma 86 it is clear that d-$\mathsf{hCollisionHashing} \in \mathsf{QNC}_{\mathcal{O}(1)}^{\mathsf{BPP}}$. The main result of this section is the following.

Lemma 99. *Fix any function $d \leq \mathrm{poly}$. Relative to a random oracle, d-$\mathsf{hCollisionHashing} \notin \mathsf{BPP}^{\mathsf{QNC}_d}$.*

[66] Obtained by setting $C := c/4$ where c is as in *Claim* 84 with $|A| = 2^{\lambda+1}$ and $|B| = 2^\lambda$ in the limit $\lambda \to \infty$; the $1/4$ factor relates the G_0, G_1 based construction to the g based construction. One can treat G_0, G_1 as special cases of g with the first input bit 0 or 1.

11.2 Consequences

Before we get into the proof of Lemma 99, we concisely state its consequences.

Theorem 100. *Fix any function $d \leq \text{poly}$. Then, relative to a random oracle, it holds that* $\mathsf{QNC}^{\mathsf{BPP}}_{\mathcal{O}(1)} \notin \mathsf{BPP}^{\mathsf{QNC}_d}$.

Note that d-$\mathsf{hCollisionHashing}$ satisfies offline soundness because $\mathsf{CollisionHashing}$ satisfies offline soundness. Therefore, using d-$\mathsf{Ser}[d$-$\mathsf{hCollisionHashing}]$ and Lemma 95, we conclude the following.

Theorem 101. *Fix any function $d \leq \text{poly}$. Then, relative to a random oracle, it holds that* $\mathsf{BPP}^{\mathsf{QNC}^{\mathsf{BPP}}_{\mathcal{O}(1)}} \notin \mathsf{QNC}^{\mathsf{BPP}}_d \cup \mathsf{BPP}^{\mathsf{QNC}_d}$.

The rest of this section is dedicated to proving Lemma 99. Since our proof makes use of the compressed oracle technique, we start by introducing it below.

11.3 The compressed oracle technique

11.3.1 An informal overview

In this subsection, we give an informal exposition of Zhandry's compressed oracle technique. This subsection is taken almost verbatim from [CGV22]. A reader who is familiar with the technique should feel free to skip this subsection.

Let $H : \{0,1\}^n \to \{0,1\}$ be a fixed function. For simplicity, in this overview we restrict ourselves to considering boolean functions (since this is also the relevant case for our scheme).

While classically it is always possible to record the queries of the algorithm, in a way that is undetectable to the algorithm itself, this is not possible in general in the quantum case. The issue arises because the quantum algorithm can *query in superposition.* We illustrate this with an example.

Consider an algorithm that prepares the state $\frac{1}{\sqrt{2}}(|x_0\rangle + |x_1\rangle)|y\rangle$, and then makes an oracle query to H. The state after the query is

$$\frac{1}{\sqrt{2}}|x_0\rangle|y \oplus H(x_0)\rangle + \frac{1}{\sqrt{2}}|x_1\rangle|y \oplus H(x_1)\rangle . \tag{22}$$

Suppose we additionally "record" the query made, i.e. we copy the queried input into a third register. Then the state becomes:

$$\frac{1}{\sqrt{2}}|x_0\rangle|y \oplus H(x_0)\rangle|x_0\rangle + \frac{1}{\sqrt{2}}|x_1\rangle|y \oplus H(x_1)\rangle|x_1\rangle \tag{23}$$

Now, suppose that $H(x_0) = H(x_1)$, then it is easy to see that, in the case where we didn't record queries, the state of the first register after the query is exactly $\frac{1}{\sqrt{2}}(|x_0\rangle + |x_1\rangle)$. On the other hand, if we recorded the query, then the third register is now entangled with the first, and as a result the state of the first register is no longer $\frac{1}{\sqrt{2}}(|x_0\rangle + |x_1\rangle)$ (it is instead a mixed state). Thus, recording queries is not possible in general without disturbing the state of the oracle algorithm.

Does this mean that all hope of recording queries is lost in the quantum setting? It turns out, surprisingly, that there is a way to record queries when H is a *uniformly random* oracle.

When thinking of an algorithm that queries a uniformly random oracle, it is useful to purify the quantum state of the algorithm via an oracle register (which keeps track of the function that is being queried). An oracle query is then a unitary that acts in the following way on a standard basis element of the query register (where we omit writing normalizing constants):

$$|x\rangle|y\rangle\sum_H |H\rangle \mapsto \sum_H |x\rangle|y \oplus H(x)\rangle|H\rangle .$$

It is well known that, up to applying a Hadamard gate on the y register before and after a query, this oracle is equivalent to a "phase oracle", which acts in the following way:

$$|x\rangle|y\rangle\sum_H |H\rangle \mapsto \sum_H (-1)^{y \cdot H(x)}|x\rangle|y\rangle|H\rangle \tag{24}$$

Now, to get a better sense of what is happening with each query, let's be more concrete about how we represent H using the qubits in the oracle register.

A natural way to represent H is to use 2^n qubits, with each qubit representing the output of the oracle at one input, where we take the inputs to be ordered lexicographically. In other words, if $|H\rangle = |t\rangle$, where $t \in \{0,1\}^{2^n}$, then this means that $H(x_i) = t_i$, where x_i is the i-th n-bit string in lexicographic order. Using this representation, notice that

$$\frac{1}{\sqrt{2^n}} \sum_H |H\rangle = |+\rangle^{\otimes 2^n} .$$

Now, notice that we can write the RHS of Equation (24) as

$$|x\rangle |y\rangle \sum_H (-1)^{y \cdot H(x)} |H\rangle ,$$

i.e. we can equivalently think of the phase in a phase oracle query as being applied to the oracle register.

Thus, when a phase oracle query is made on a standard basis vector of the query register $|x\rangle |y\rangle$, all that happens is

$$\sum_H |H\rangle \mapsto \sum_H (-1)^{y \cdot H(x)} |H\rangle .$$

Notice that, using the representation for H that we chose above, the latter transformation is:

- When $y = 0$,
$$|+\rangle^{\otimes 2^n} \mapsto |+\rangle^{\otimes 2^n} .$$
- When $y = 1$,
$$|+\rangle^{\otimes 2^n} \mapsto |+\rangle \cdots |+\rangle_{i-1} |-\rangle_i |+\rangle_{i+1} \cdots |+\rangle ,$$
where i is such that x is the i-th string in lexicographic order.

In words, the query does not have any effect when $y = 0$, and the query flips the appropriate $|+\rangle$ to a $|-\rangle$ when $y = 1$. Then, when we query on a general state $\sum_{x,y} \alpha_{xy} |x\rangle |y\rangle$, the state after the query can be written as:

$$\sum_{x,y} \alpha_{xy} |x\rangle |y\rangle |D_{xy}\rangle ,$$

where D_{xy} is the all $|+\rangle$ state, except for a $|-\rangle$ corresponding to x if $y = 1$.

The crucial observation now is that all of these branches are *orthogonal*, and thus it makes sense to talk about "the branch on which a particular query was made": the state of the oracle register reveals exactly the query that has been made on that branch. More generally, after q queries, the state will be in a superposition of branches on which at most q of the $|+\rangle$'s have been flipped to $|-\rangle$'s. These locations correspond exactly to the queries that have been made.

Moreover, the good news is that there is a way to keep track of the recorded queries *efficiently*: one does not need to store all of the (exponentially many) $|+\rangle$'s, but it suffices to keep track only of the locations that have flipped to $|-\rangle$ (which is at most q). If we know that the oracle algorithm makes at most q queries, then we need merely $n \cdot q$ qubits to store the points that have been queried. We will refer to the set of queried points as the *database*. Formally, there is a well-defined isometry that maps a state on 2^n qubits where q of them are in the $|-\rangle$ state, and the rest are $|+\rangle$, to a state on $n \cdot q$ qubits, which stores the q points corresponding to the $|-\rangle$'s in lexicographic order.

Let D denote an empty database of queried points. Then a query to a uniformly random oracle can be thought of as acting in the following way:

$$\begin{cases} |x\rangle |y\rangle |D\rangle \mapsto |x\rangle |y\rangle |D\rangle \, , \text{ if } y = 0 \\ |x\rangle |y\rangle |D\rangle \mapsto |x\rangle |y\rangle |D \cup \{x\}\rangle \, , \text{ if } y = 1 \, . \end{cases}$$

Such an implementation a uniformly random oracle is referred to as a *compressed phase oracle* simulation [Zha19]. Formally, the fact that the original and the compressed oracle simulations are *identical* from the point of view of the oracle algorithm (which does not have access to the oracle register) is because at any point in the execution of the algorithm, the states in the two simulations are both purifications of the same mixed state on the algorithm's registers.

We point out that there are two properties of a uniformly random oracle that make a compressed oracle simulation possible:

- The query outputs at each point are independently distributed, which means that the state of the oracle register is always a product state across all of the 2^n qubits.
- Each query output is uniformly distributed. This is important because in general $\alpha|0\rangle+\beta|1\rangle \not\perp \alpha|0\rangle-\beta|1\rangle$ unless $|\alpha| = |\beta|$.

Notice that the above compressed oracle simulation does not explicitly keep track of the value of the function at the queried points (i.e. a database is just a set of queried points). In the following slight variation on the compressed oracle simulation, also from [Zha19], a database is instead a set of pairs (x, w) representing a queried point and the value of the function at that point. This variation will be more useful for our analysis.

Here D is a database of pairs (x, v), which is initially empty. A query acts as follows on a standard basis element $|x\rangle|y\rangle|D\rangle$:

- If $y = 0$, do nothing.
- If $y = 1$, check if D contains a pair of the form (x, v) for some v.
  - If it does not, add $(x, |-\rangle)$ to the database, where by this we formally mean: $D \mapsto \sum_v (-1)^v |D \cup (x, v)\rangle$
  - If it does, apply the unitary that removes $(x, |-\rangle)$ from the database.

One way to understand this compressed simulation is that our database representation only keeps track of pairs $(x, |-\rangle)$ (corresponding to the queried points), and it does not keep track of the other unqueried points, which in a fully explicit simulation would correspond to $|+\rangle$'s. One can think of the outputs at the unqueried points as being "compressed" in this succinct representation.

It is easy to see that the map above can be extended to a well-defined unitary. In the rest of this overview, we will take this to be our compressed phase oracle. For an oracle algorithm A, we will denote by A^{comp} the algorithm A run with a compressed phase oracle.

11.3.2 A formal introduction

In this subsection, we formally introduce Zhandry's technique for recording queries [Zha19]. This section is loosely based on the explanation in [Zha19]. For a more informal treatment, which carries most of the essence, we suggest starting from the previous section.

Standard and Phase Oracles The quantum random oracle, which is the quantum analogue of the classical random oracle, is typically presented in one of two variations: as a *standard* or as a *phase* oracle.

The standard oracle is a unitary acting on three registers: an n-qubit register representing the input to the function, an m-qubit register for writing the response, and an $m2^n$ qubit register representing the truth table of the queried function $H : \{0,1\}^n \to \{0,1\}^m$. The algorithm that queries the standard oracle has access to the first two registers, while the third register, the oracle's state, is hidden from the algorithm except by making queries. The standard oracle unitary acts in the following way on standard basis states:

$$|x\rangle|y\rangle|H\rangle \mapsto |x\rangle|y \oplus H(x)\rangle|H\rangle .$$

For a uniformly random oracle, the oracle register is initialized in the uniform superposition $\frac{1}{\sqrt{m2^n}} \sum_H |H\rangle$. This initialization is of course equivalent to having the oracle register be in a completely mixed state (i.e. a uniformly chosen H). This equivalence can be seen by just tracing out the oracle register. We denote the standard (uniformly random) oracle unitary by StO. Moreover, for an oracle algorithm A, we will denote by A^{StO} the algorithm A interacting with the standard oracle, implemented as above.

The phase oracle formally gives a different interface to the algorithm making the queries, but is equivalent to the standard oracle up to Hadamard gates. It again acts on three registers: an n-qubit register for the input, an m-qubit "phase" register, and an $m2^n$-qubit oracle register. It acts in the following way on standard basis states:

$$|x\rangle|s\rangle|H\rangle \mapsto (-1)^{s \cdot H(x)}|x\rangle|s\rangle|H\rangle .$$

For a uniformly random oracle, the oracle register is again initialized in the uniform superposition. One can easily see that the standard and phase oracles are equivalent up to applying a Hadamard gate on the phase register before and after a query. We denote the phase oracle unitary by PhO. Moreover, for an oracle algorithm A, we will denote by A^{PhO} the algorithm A interacting with the phase oracle.

Compressed oracle The *compressed oracle* technique, introduced by Zhandry [Zha19], is an equivalent way of implementing a quantum random oracle which (i) is efficiently implementable, and (ii) keeps track of the queried inputs in a meaningful way. This paragraph is loosely based on the explanation in [Zha19].

In a compressed oracle, the oracle register does not represent the full truth table of the queried function. Instead, it represents a *database* of queried inputs, and the values at those inputs. More precisely, if we have an upper bound t on the number of queries, a database D is represented as an element of the set S^t where $S = (\{0,1\}^n \cup \{\bot\}) \times \{0,1\}^m$. Each value in S is a pair (x, y): if $x \neq \bot$, then the pair means that the value of the function at x is y, which we denote by $D(x) = y$; and if $x = \bot$, then the pair is not currently used, which we denote by $D(x) = \bot$. Concretely, let $l \leq t$. Then, for $x_1 < x_2 < \ldots < x_l$ and $y_1, \ldots, y_l$, the database representing $D(x_i) = y_i$ for $i \in [l]$, with the other $t - l$ points unspecified, is represented as

$$\Big((x_1, y_1), (x_2, y_2), \ldots, (x_l, y_l), (\bot, 0^m), \ldots, (\bot, 0^m)\Big)$$

where the number of $(\bot, 0^m)$ pairs is $t - l$. We emphasise that in this database representation, the pairs are always ordered lexicographically according to the input value, and the $(\bot, 0^m)$ pairs are always at the end.

In order to define precisely the action of a compressed oracle query, we need to introduce some additional notation. Let $|D|$ denote the number of pairs (x, y) in database D with $x \neq \bot$. Let t be an upper bound on the number of queries. Then, for a database D with $|D| < t$ and $D(x) = \bot$, we write $D \cup (x, y)$ to denote the new database obtained by deleting one of the $(\bot, 0^m)$ pairs, and by adding the pair (x, y) to D, inserted at the appropriate location (to respect the lexicographic ordering of the input values).

We also define a "decompression" procedure. For $x \in \{0,1\}^n$, Decomp_x is a unitary operation on the database register. If $D(x) = \bot$, it adds a uniform superposition over all pairs (x, y) (i.e. it "uncompressed" at x). Otherwise, if D is specified at x, and the corresponding y register is in a uniform superposition, Decomp removes x and the uniform superposition from D. If D is specified at x, and the corresponding y register is in a state orthogonal to the uniform superposition, then Decomp acts as the identity. More precisely,

- For D such that $D(x) = \bot$ and $|D| < t$,

$$\mathsf{Decomp}_x \left|D\right\rangle = \frac{1}{\sqrt{2^m}} \sum_y \left|D \cup (x, y)\right\rangle .$$

- For D such that $D(x) = \bot$ and $D = t$,

$$\mathsf{Decomp}_x \left|D\right\rangle = \left|D\right\rangle .$$

- For D such that $D(x) \neq \bot$ and $|D| < t$,

$$\mathsf{Decomp}_x \left(\frac{1}{\sqrt{2^m}} \sum_y (-1)^{z \cdot y} \left|D \cup (x, y)\right\rangle \right) = \begin{cases} \frac{1}{\sqrt{2^m}} \sum_y (-1)^{z \cdot y} \left|D \cup (x, y)\right\rangle & \text{if } z \neq 0 \\ \left|D\right\rangle & \text{if } z = 0 \end{cases} \tag{25}$$

Note that we have specified the action of Decomp_x on an orthonormal basis of the database register (with a bound of t on the size of the database). Moreover, it is straightforward to verify that Decomp_x maps this orthonormal basis to another orthonormal basis, and is thus a well-defined unitary. Furthermore, observe that applying Decomp_x twice gives the identity. Let Decomp be the related unitary acting on all the registers x, y, D which acts as

$$\mathsf{Decomp} \left|x, y\right\rangle \otimes \left|D\right\rangle = \left|x, y\right\rangle \otimes \mathsf{Decomp}_x \left|D\right\rangle .$$

So far, we have considered a fixed upper bound on the number of queries. However, one of the advantages of the compressed oracle technique is that an upper bound on the number of queries does not need to be known in advance. To handle a number of queries that is not fixed, we defined the procedure $\mathsf{Increase}$ which simply increases the upper bound on the size of the database by initialising a new register in the state $\left|(\bot, 0^n)\right\rangle$, and appending it to the end. Formally, $\mathsf{Increase} \left|x, y\right\rangle \otimes \left|D\right\rangle = \left|x, y\right\rangle \otimes \left|D\right\rangle \left|(\bot, 0^n)\right\rangle$.

Now, define the unitaries CStO' and CPhO' acting as

$$\begin{aligned} \mathsf{CStO}' \left|x, y\right\rangle \otimes \left|D\right\rangle &= \left|x, y \oplus D(x)\right\rangle \otimes \left|D\right\rangle \\ \mathsf{CPhO}' \left|x, y\right\rangle \otimes \left|D\right\rangle &= (-1)^{y \cdot D(x)} \left|x, y\right\rangle \otimes \left|D\right\rangle \end{aligned} \tag{26}$$

Finally, we define the compressed standard and phase oracles CStO and CPhO as:

$$\begin{aligned}\mathsf{CStO} &= \mathsf{Decomp} \circ \mathsf{CStO}' \circ \mathsf{Decomp} \circ \mathsf{Increase} \\ \mathsf{CPhO} &= \mathsf{Decomp} \circ \mathsf{CPhO}' \circ \mathsf{Decomp} \circ \mathsf{Increase}\end{aligned} \tag{27}$$

For an oracle algorithm A, we denote by A^{CStO} (resp. A^{CPhO}), the algorithm A run with the compressed standard (resp. phase) oracle, implemented as described above. The following lemma establishes that regular and compressed oracles are equivalent.

Lemma 102 ([Zha19])**.** *For any oracle algorithm A, and any input state $|\psi\rangle$, $\Pr[A^{\mathsf{StO}}(|\psi\rangle) = 1] = \Pr[A^{\mathsf{CStO}}(|\psi\rangle) = 1]$. Similarly, for any oracle algorithm B, $\Pr[B^{\mathsf{PhO}}(|\psi\rangle) = 1] = \Pr[B^{\mathsf{CPhO}}(|\psi\rangle) = 1]$.*

In the rest of the section, we choose to work with *phase* oracles and compressed *phase* oracles. Moreover, to use a more suggestive name, we will denote the compressed phase oracle CPhO by comp.

11.4 d-hCollisionHashing $\notin \mathsf{BPP}^{\mathsf{QNC}_d}$

Theorem 103. *Let $d \leq \text{poly}$. Then, any $\mathsf{CQ_d}$ algorithm solves d-$\mathsf{hCollisionHashing}$ with probability at most $1/(1+\frac{C}{3}) + \mathsf{negl}$, for some negligible function negl where $C = 1/(2(e^2-1))$.*

The following lemma captures the intuition that the quantum part of a $\mathsf{CQ_d}$ algorithm does not have sufficient depth to evaluate $h = H_d \circ \cdots \circ H_0$ on its own. We show that, without loss of generality, we can restrict our analysis to potentially unbounded hybrid classical-quantum algorithms where queries to G_0, G_1, H and H_d are polynomially bounded, and moreover the quantum part of the algorithm *does not have access to H_d at all.* To help state this reduction formally, we denote a potentially unbounded hybrid classical-quantum algorithm by CQ_∞. In other words, a CQ_∞ algorithm has the same structure as a $\mathsf{CQ_d}$ algorithm except that its classical and quantum parts are computationally unbounded (but they may be query bounded). Then, for $d \leq \text{poly}$, we denote by $\mathcal{W}_d$ be the set of algorithms $B \in \mathsf{CQ}_\infty$ for d-$\mathsf{hCollisionHashing}$ that satisfy the following properties:

1. B only makes polynomially many queries to G_0, G_1, H, and a (potentially) unbounded number of queries to $H_0, \ldots, H_{d-1}$.

2. The quantum part of B does not have access to H_d.

3. The classical part of B only makes polynomially many queries to H_d.

Lemma 104. *Let $d \leq \text{poly}$. Suppose A is a $\mathsf{CQ_d}$ algorithm that solves d-$\mathsf{hCollisionHashing}$ with probability p. Then, there exists a negligible function negl and an algorithm $B \in \mathcal{W}_d$ that solves the same problem with probability at least $p - \mathsf{negl}$.*

Proof sketch. Following an argument similar to that in the proof of Lemma 74 (on $H_0 \ldots H_d$ which when composed yield h), one can show that a circuit in Figure 9a behaves like the circuit in Figure 9c, i.e. their trace distance is negligible. By inspection, it follows that Figure 9c can be simulated by circuit B above. Therefore if A succeeds with p at any task, B succeeds at the same task with probability at least $p - \text{negl}(\lambda)$. □

From now on, without loss of generality, we restrict to considering algorithms for d-$\mathsf{hCollisionHashing}$ that are in $\mathcal{W}_d$. We will show that no such algorithm can solve d-$\mathsf{hCollisionHashing}$ with probability greater than $1/(1+\frac{C}{3}) + \mathsf{negl}$.

It may be surprising that a seemingly strong class of algorithms $\mathcal{W}_d$ cannot solve d-$\mathsf{hCollisionHashing}$ with probability close to 1. Indeed, the crucial resource that is missing from algorithms in $\mathcal{W}_d$ is that they are unable to maintain coherence while making new queries to H_d. This is because, by definition, H_d can only be queried by the classical part.

From here on, we fix a $d \leq \text{poly}$, and we simply refer to d-$\mathsf{hCollisionHashing}$ as $\mathsf{Problem}$, and to the single-copy version as $\mathsf{subProblem}$. The first step in our proof is to reduce the analysis of algorithms for $\mathsf{Problem}$ to algorithms for $\mathsf{subProblem}$.

Lemma 105. *Suppose there exists an algorithm $B \in \mathcal{W}_d$ that solves $\mathsf{Problem}$ with probability non-negligibly greater than $1/(1+\frac{C}{3})$. Then, there exists an algorithm $A = \{A_\lambda\}_{\lambda\in\mathbb{N}} \in \mathcal{W}_d$ for $\mathsf{subProblem}$, and a non-negligible function $\mathsf{non\text{-}negl}$ such that, for all λ,*

- $\Pr[A_\lambda \text{ outputs } y \text{ s.t. } y \in \mathsf{TwoToOne}(G_0, G_1)] \geq \mathsf{non\text{-}negl}(\lambda)$, *and*
- $\Pr[A_\lambda \text{ wins } | \, y \in \mathsf{TwoToOne}(G_0, G_1)] \geq \frac{1}{2} + \mathsf{non\text{-}negl}(\lambda)$,

where "A_λ wins*" is shorthand for "A_λ outputs a valid equation".*

Proof. Let $B = \{B_\lambda\}_{\lambda \in \mathbb{N}}$ be a $\mathcal{W}_d$ algorithm that solves $\mathsf{Problem}$ with probability non-negligibly greater than $\frac{1}{2}$. Suppose for a contradiction that the lemma does not hold. This implies that, for all $\mathcal{W}_d$ algorithms $A = \{A_\lambda\}_{\lambda \in \mathbb{N}}$ for $\mathsf{subProblem}$, there exists a negligible function negl such that, for all λ,

- $\Pr[A_\lambda \text{ outputs } y \text{ s.t. } y \in \mathsf{TwoToOne}(G_0, G_1)] \leq \mathsf{negl}(\lambda)$, or
- $\Pr[A_\lambda \text{ wins } | \, y \in \mathsf{TwoToOne}(G_0, G_1)] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$.

Let $B^i = \{B^i_\lambda\}_{\lambda \in \mathbb{N}}$ be the algorithm for $\mathsf{subProblem}$ that runs algorithm B and returns the i-th answer of B as output. Since B^i is a $\mathcal{W}_d$ algorithm, the hypothesis above implies that there exists a negligible function negl_i such that, for all λ,

- $\Pr[B^i_\lambda \text{ outputs } y \text{ s.t. } y \in \mathsf{TwoToOne}(G_0, G_1)] \leq \mathsf{negl}_i(\lambda)$, or
- $\Pr[B^i_\lambda \text{ wins } | \, y \in \mathsf{TwoToOne}(G_0, G_1)] \leq \frac{1}{2} + \mathsf{negl}_i(\lambda)$.

Let $\mathsf{negl} = \max_i \mathsf{negl}_i$. This is still a negligible function. Then, we have that, for all $i \in [\lambda]$,

(i) $p_{i,\lambda} \coloneqq \Pr[B^i_\lambda \text{ outputs } y \text{ s.t. } y \in \mathsf{TwoToOne}(G_0, G_1)] \leq \mathsf{negl}(\lambda)$, or

(ii) $q_{i,\lambda} \coloneqq \Pr[B^i_\lambda \text{ wins } | \, y \in \mathsf{TwoToOne}(G_0, G_1)] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$.

Let $\mathcal{J}_\lambda = \{i : p_{i,\lambda} \leq \mathsf{negl}(\lambda)\}$, and let $\bar{\mathcal{J}}_\lambda \coloneqq [\lambda] \setminus \mathcal{J}_\lambda$.

For brevity, denote by $\mathrm{y} = y_1, \ldots, y_\lambda$, and similarly for r, m. It follows from the above and a union bound that, for all λ,

$$\Pr[\exists \, i \in \mathcal{J}_\lambda \text{ s.t. } y_i \in \mathsf{TwoToOne}(G_0, G_1) : (\mathrm{y}, \mathrm{r}, \mathrm{m}) \leftarrow B_\lambda] \leq \mathsf{negl}'(\lambda), \tag{28}$$

where $\mathsf{negl}'(\lambda) = \lambda \cdot \mathsf{negl}(\lambda)$. We can rewrite the latter as

$$\Pr[\mathcal{I} \cap \mathcal{J}_\lambda \neq \varnothing] \leq \mathsf{negl}'(\lambda). \tag{29}$$

Using the same notation as in the description of $\mathsf{Problem}$, we have

$$\Pr[B \text{ wins}] \leq \Pr\left[|\mathcal{I}_{\mathsf{win}}| \geq \frac{3}{4} \cdot |\mathcal{I}|\right], \tag{30}$$

since the event "B wins" is a subset of the event "$|\mathcal{I}_{\mathsf{win}}| \geq \frac{3}{4} \cdot |\mathcal{I}|$". Now, we have

$$\begin{aligned} \Pr\left[|\mathcal{I}_{\mathsf{win}}| \geq \frac{3}{4} \cdot |\mathcal{I}|\right] &= \Pr[\mathcal{I} \cap \mathcal{J}_\lambda = \varnothing] \cdot \Pr\left[|\mathcal{I}_{\mathsf{win}}| \geq \frac{3}{4} \cdot |\mathcal{I}| \,\middle|\, \mathcal{I} \cap \mathcal{J}_\lambda = \varnothing\right] \\ &\quad + \Pr[\mathcal{I} \cap \mathcal{J}_\lambda \neq \varnothing] \cdot \Pr\left[|\mathcal{I}_{\mathsf{win}}| \geq \frac{3}{4} \cdot |\mathcal{I}| \,\middle|\, \mathcal{I} \cap \mathcal{J}_\lambda \neq \varnothing\right] \\ &\leq \Pr[\mathcal{I} \cap \mathcal{J}_\lambda = \varnothing] \cdot \Pr\left[|\mathcal{I}_{\mathsf{win}}| \geq \frac{3}{4} \cdot |\mathcal{I}| \,\middle|\, \mathcal{I} \cap \mathcal{J}_\lambda = \varnothing\right] + \mathsf{negl}'(\lambda) \\ &= \Pr[\mathcal{I} \cap \mathcal{J}_\lambda = \varnothing] \cdot \Pr\left[|\mathcal{I}_{\mathsf{win}} \cap \bar{\mathcal{J}}_\lambda| \geq \frac{3}{4} \cdot |\mathcal{I} \cap \bar{\mathcal{J}}_\lambda| \,\middle|\, \mathcal{I} \cap \mathcal{J}_\lambda = \varnothing\right] + \mathsf{negl}'(\lambda). \end{aligned} \tag{31}$$

where the first inequality is implied by Equation (29), and the final equality is because, conditioned on $\mathcal{I} \cap \mathcal{J}_\lambda = \varnothing$, we have that $\mathcal{I}_{\mathsf{win}} = \mathcal{I}_{\mathsf{win}} \cap \bar{\mathcal{J}}_\lambda$, and $\mathcal{I} = \mathcal{I} \cap \bar{\mathcal{J}}_\lambda$.

Finally, notice that

$$\begin{aligned} &\Pr[\mathcal{I} \cap \mathcal{J}_\lambda = \varnothing] \cdot \Pr\left[|\mathcal{I}_{\mathsf{win}} \cap \bar{\mathcal{J}}_\lambda| \geq \frac{3}{4} \cdot |\mathcal{I} \cap \bar{\mathcal{J}}_\lambda| \,\middle|\, \mathcal{I} \cap \mathcal{J}_\lambda = \varnothing\right] \\ &= \Pr\left[|\mathcal{I}_{\mathsf{win}} \cap \bar{\mathcal{J}}_\lambda| \geq \frac{3}{4} \cdot |\mathcal{I} \cap \bar{\mathcal{J}}_\lambda| \text{ and } \mathcal{I} \cap \mathcal{J}_\lambda = \varnothing\right] \\ &\leq \Pr\left[|\mathcal{I}_{\mathsf{win}} \cap \bar{\mathcal{J}}_\lambda| \geq \frac{3}{4} \cdot |\mathcal{I} \cap \bar{\mathcal{J}}_\lambda|\right] \end{aligned} \tag{32}$$

Combining Equation (30), Equation (31), and Equation (32) gives

$$\Pr[B \text{ wins}] \leq \Pr\left[|\mathcal{I}_{\mathsf{win}} \cap \bar{\mathcal{J}}_\lambda| \geq \frac{3}{4} \cdot |\mathcal{I} \cap \bar{\mathcal{J}}_\lambda|\right] + \mathsf{negl}'(\lambda)\,. \tag{33}$$

Now, notice first that,

$$\begin{aligned}\Pr\left[|\mathcal{I} \cap \bar{\mathcal{J}}_\lambda| \geq \frac{3}{4}C\lambda\right] &\geq \Pr\left[|\mathcal{I} \cap \bar{\mathcal{J}}_\lambda| \geq \frac{3}{4}C\lambda \text{ and } \mathcal{I} \cap \mathcal{J}_\lambda = \varnothing\right] \\ &= \Pr\left[|\mathcal{I}| \geq \frac{3}{4}C\lambda \text{ and } \mathcal{I} \cap \mathcal{J}_\lambda = \varnothing\right] \\ &\geq \Pr\left[|\mathcal{I}| \geq \frac{3}{4}C\lambda\right] - \mathsf{negl}'(\lambda) \\ &\geq \Pr[B \text{ wins}] - \mathsf{negl}'(\lambda)\,, \end{aligned} \tag{34}$$

where the second inequality follows from Equation (29). This implies that

$$\mathbb{E}[|\mathcal{I} \cap \bar{\mathcal{J}}_\lambda|] \geq \Pr[B \text{ wins}] \cdot \frac{3}{4}C\lambda - \mathsf{negl}'(\lambda)\,. \tag{35}$$

Next, we proceed to upper bound $\Pr\left[|\mathcal{I}_{\mathsf{win}} \cap \bar{\mathcal{J}}_\lambda| \geq \frac{3}{4} \cdot |\mathcal{I} \cap \bar{\mathcal{J}}_\lambda|\right]$. Together with Equation (33), this will yield a contradiction.

Notice that, by (i) and (ii), for all $i \in \bar{\mathcal{J}}_\lambda$,

$$\Pr[B^i_\lambda \text{ wins} \mid y_i \in \mathsf{TwoToOne}(G_0, G_1)] \leq \frac{1}{2} + \mathsf{negl}(\lambda)\,. \tag{36}$$

Now, for $i \in \bar{\mathcal{J}}_\lambda$, define E_i to be the random variable such that:

$$E_i = \begin{cases} 1 \text{ if } B^i_\lambda \text{ wins and } y_i \in \mathsf{TwoToOne}(G_0, G_1) \\ 0 \text{ otherwise} \end{cases} \tag{37}$$

Define F_i to be the random variable such that:

$$F_i = \begin{cases} 1 \text{ if } y_i \in \mathsf{TwoToOne}(G_0, G_1) \\ 0 \text{ otherwise} \end{cases} \tag{38}$$

Let $E := \frac{1}{\lambda}\sum_{i \in \bar{\mathcal{J}}_\lambda} E_i$, and $F := \frac{1}{\lambda}\sum_{i \in \bar{\mathcal{J}}_\lambda} F_i$. Note that $E = |\mathcal{I}_{\mathsf{win}} \cap \bar{\mathcal{J}}_\lambda|/\lambda$, and $F = |\mathcal{I} \cap \bar{\mathcal{J}}_\lambda|/\lambda$ Then,

$$\begin{aligned}\mathbb{E}[E] &= \sum_{i \in \bar{\mathcal{J}}_\lambda} \mathbb{E}[E_i] \\ &= \sum_{i \in \bar{\mathcal{J}}_\lambda} \Pr[B^i_\lambda \text{ wins and } y_i \in \mathsf{TwoToOne}(G_0, G_1)] \\ &= \sum_{i \in \bar{\mathcal{J}}_\lambda} \Pr[B^i_\lambda \text{ wins} \mid y_i \in \mathsf{TwoToOne}(G_0, G_1)] \cdot \Pr[y_i \in \mathsf{TwoToOne}(G_0, G_1)] \\ &\leq (\frac{1}{2} + \mathsf{negl}(\lambda)) \cdot \sum_{i \in \bar{\mathcal{J}}_\lambda} \Pr[y_i \in \mathsf{TwoToOne}(G_0, G_1)] \\ &= (\frac{1}{2} + \mathsf{negl}(\lambda)) \cdot \sum_{i \in \bar{\mathcal{J}}_\lambda} \mathbb{E}[F_i] \\ &= (\frac{1}{2} + \mathsf{negl}(\lambda)) \cdot \mathbb{E}[F]\,. \end{aligned} \tag{39}$$

We make use of the following:

Claim 106. Let E and F be random variables taking values in $[0,1]$. Let $\gamma \in [0,1]$.

$$\Pr[E \geq \gamma \cdot F] \leq 1 - \mathbb{E}(F)\left(1 - \frac{\mathbb{E}(E)}{\gamma \cdot \mathbb{E}(F)}\right)$$

Proof. The proof is straightforward and follows from some averaging arguments. It is included in the Appendix for completeness. □

We invoke the claim with E and F defined earlier, and $\gamma = \frac{3}{4}$. In our case, by Equation (35),

$$\mathbb{E}(F) \geq \Pr[B \text{ wins}] \cdot \frac{3}{4}C - \mathsf{negl}'(\lambda)\,,$$

and, by Equation (39),

$$\frac{\mathbb{E}(E)}{\mathbb{E}(F)} \leq \frac{1}{2} + \mathsf{negl}(\lambda)\,.$$

Then, by Claim 106, we have

$$\Pr\left[E \geq \frac{3}{4}F\right] \leq 1 - \frac{\Pr[B \text{ wins}] \cdot C}{3} - \mathsf{negl}''(\lambda)\,, \tag{40}$$

for some negligible function negl''. Combining Equation (40) with Equation (33), and recalling the Definition of E and F, we have

$$\Pr[B \text{ wins}] - \mathsf{negl}(\lambda)' \leq 1 - \frac{\Pr[B \text{ wins}] \cdot C}{3} - \mathsf{negl}''(\lambda)\,,$$

which implies

$$\Pr[B \text{ wins}] \leq 1/\big(1 + \frac{C}{3}\big) + \mathsf{negl}'''(\lambda)\,,$$

for some negligible function negl'''. This is a contradiction.

□

Lemma 107. *Let $A = \{A_\lambda\}_{\lambda\in\mathbb{N}} \in \mathcal{W}_d$ be an algorithm for* **subProblem***. Suppose there exists a function ε such that, for all λ,*

- $\Pr[A_\lambda \text{ outputs } y \text{ s.t. } y \in \mathsf{TwoToOne}(G_0, G_1)] \geq \varepsilon(\lambda)$, *and*
- $\Pr[A_\lambda \text{ wins} \mid y \in \mathsf{TwoToOne}(G_0, G_1)] \geq \frac{1}{2} + \varepsilon(\lambda)\,.$

Then, there exists a (potentially unbounded) oracle algorithm that makes polynomially many queries to G_0, G_1 and outputs a collision with probability at least $\frac{poly(\varepsilon)}{q^3}$, where q is the total number of queries to G_0, G_1, H, H_d made by A.

Recall that q is polynomially bounded, so this quantity is non-negligible when ε is non-negligible. Lemma 104, Lemma 105, and Lemma 107 together clearly imply Theorem 103. The rest of the section is dedicated to proving Lemma 107.

Algorithm 1 below is the algorithm that extracts a collision. We introduce some notation before describing it. Recall that A alternates classical and quantum circuits. Without loss of generality, we can take A_λ to be a quantum circuit that applies the unitary:

$$\big(\mathsf{CNOT}_{(\mathsf{work},\mathsf{query})\to\mathsf{rec}}(U_Q O^G O^H)^{L'}(U_C O^G O^H O^{H_d})^L\big)^N\,,$$

where:

- U_C is a "classical" unitary that is diagonal in the standard basis, and acts on registers work, query. We assume that U_C also includes (a potentially unbounded number of) queries to oracles $H_0, \ldots, H_{d-1}$.
- U_Q is a unitary acting on registers work, query. We again assume that U_Q includes (a potentially unbounded number of) quantum queries to oracles $H_0, \ldots, H_{d-1}$.
- N is the total number of quantum circuits. L and L' are respectively the number of oracles calls in each classical and quantum part.
- $\mathsf{CNOT}_{(\mathsf{work},\mathsf{query})\to\mathsf{rec}}$ is a CNOT gate that "measures" all of the registers after each QNC execution by copying them in another register rec.

Note that we are assuming, without loss of generality, that the BPP and QNC parts share the same registers, but *all* registers are measured after each QNC call.

Finally, for y in the range of G_0, G_1, and $c_0, c_1 \in \{0,1\}$, denote by A_λ^{y,c_0,c_1} the algorithm that is identical to A, except for the following modification: replace oracle queries O^H with O^H_{y,c_0,c_1} defined as follows:

$$O^H_{y,c_0,c_1} |x,w\rangle |z\rangle = \begin{cases} (-1)^{z\cdot H(x)} |x,w\rangle |z\rangle\,, \text{ if } G_0(x), G_1(x) \neq y \\ (-1)^{z\cdot c_0} |x,w\rangle |z\rangle\,, \text{ if } G_0(x) = y \\ (-1)^{z\cdot c_1} |x,w\rangle |z\rangle\,, \text{ if } G_0(x) \neq y \text{ and } G_1(x) = y \end{cases}$$

where O^H_{y,c_0,c_1} is implemented "in place", by querying $G_0, G_1(x)$, computing in an auxiliary register which of the three cases one is in, applying a controlled unitary based on the value of the control register, and uncomputing everything except the controlled unitary (which returns the auxiliary register to zero). Crucially, O^H_{y,c_0,c_1} can be computed at the cost of one query to G_0 and G_1.

Algorithm 1 (Extract a collision)**.**
Input: a security parameter 1^λ
Oracle access to: $G_0, G_1 : \{0,1\}^{n(\lambda)} \to \{0,1\}^{n(\lambda)}$
Run a simulation of the following algorithm, where oracle calls to H are simulated via a compressed oracle simulation, and calls to $H_0, \ldots, H_d$ are simulated inefficiently (by sampling these functions uniformly at random and using a truth table to answer queries). Calls to G_0, G_1 are made directly to the oracles G_0, G_1.

(i) Pick $\mathsf{i} \leftarrow [N \cdot L]$ (where notice that the latter is the total number of oracle calls that A_λ makes to h). Let N_i, L_i be such that $\mathsf{i} = N_i \cdot L + L_i$, with $0 \leq L_i < N$.

(ii) Run A_λ up until just before the i-th query to H_d, i.e. apply the unitary

$$(U_C O^G O^H O^{H_d})^{L_i - 1} \circ \Big(\mathsf{CNOT}_{\mathsf{out},\mathsf{rec}} (U_Q O^G O^H)^{L'} (U_C O^G O^H O^{H_d})^L\Big)^{N_i}\,.$$

Then, measure registers work *and* query*, and let adv be the outcome. Moreover, let z be the i-th query to H_d. Let $\mathsf{h}^i_{\mathsf{data}}$ denote the set $\mathsf{h}^i_{\mathsf{data}} = h^{-1}(H_d(z))$ (this set can be computed inefficiently by querying $H_0, \ldots, H_{d-1}$ everywhere).*

(iii) Pick $\tilde{y} \leftarrow \mathsf{h}^i_{\mathsf{data}}$. Pick $c_0, c_1, c_0', c_1' \leftarrow \{0,1\}$, and $j, j' \leftarrow [(N - N_i) \cdot (L + L') - L_i]$ (where notice that the latter is the total number of remaining oracle calls to H that the partial run of A_λ in step (ii) did not perform). Let V_j and $V'_{j'}$ be unitaries corresponding to the continuation of the execution of A from where it stopped in step (ii), for respectively j and j' additional queries to H, where we additionally replace oracle calls O^H with oracle calls $O^H_{\tilde{y},c_0,c_1}$ and $O^H_{\tilde{y},c_0',c_1'}$ for V_j and $V'_{j'}$ respectively (we describe these formally after the description of the algorithm).

(iv) Initialize new registers work' *and* query' *in the state $|adv\rangle$. Run*[67]

$$(V_j \otimes V'_{j'})\big(\, |adv\rangle_{\mathsf{work},\mathsf{query}} \otimes |adv\rangle_{\mathsf{work}',\mathsf{query}'}\,\big)$$

(vi) Measure the query registers of H in query *and* query' *and output a collision if one is found.*

To avoid any confusion, we give a formal definition of V_j and $V'_{j'}$. Let N_j, L_j be such that $j = N_j \cdot (L + L') + L_j$, where $0 \leq L_j < L + L'$. Let $\mathsf{CNOT}_{\to\mathsf{rec}}$ be short for $\mathsf{CNOT}_{(\mathsf{work},\mathsf{query})\to\mathsf{rec}}$. Define

$$V_j := W_j \circ \Big((U_C O^G O^H_{\tilde{y},c_0,c_1} O^{H_d})^{L_i - 1} \mathsf{CNOT}_{\to\mathsf{rec}} (U_Q O^G O^H_{\tilde{y},c_0,c_1})^{L'} (U_C O^G O^H_{\tilde{y},c_0,c_1} O^{H_d})^{L - L_i + 1}\Big)^{N_j}\,,$$

where

$$W_j := \begin{cases} (U_C O^G O^H_{\tilde{y},c_0,c_1} O^{H_d})^{L_j} \text{ if } L_j \leq L - L_i + 1 \\ (U_Q O^G O^H_{\tilde{y},c_0,c_1})^{L_j - (L - L_i + 1)} (U_C O^G O^H_{\tilde{y},c_0,c_1} O^{H_d})^{L - L_i + 1} \text{ if } L - L_i + 1 < L_j \leq L - L_i + 1 + L' \\ (U_C O^G O^H_{\tilde{y},c_0,c_1} O^{H_d})^{L_j - (L' + L - L_i + 1)} \mathsf{CNOT}_{\to\mathsf{rec}} (U_Q O^G O^H_{\tilde{y},c_0,c_1})^{L'} (U_C O^G O^H_{\tilde{y},c_0,c_1} O^{H_d})^{L - L_i + 1} \text{ otherwise} \end{cases} \quad (41)$$

$V'_{j'}$ is defined analogously (with c_0, c_1 replaced by c_0', c_1').

[67] Note that, while the oracle queries in the "left" and "right" unitaries act on distinct registers query and query', one can equivalently replace this unitary with one in which there is a single shared query register, by having one algorithm swap the contents of a local register into the shared query register, query the oracle, and swap out the contents back into the local register.

11.4.1 A technical lemma

Let A be an oracle algorithm making q queries to a uniformly random function $H : \{0,1\}^n \to \{0,1\}$. Denote by work and query the registers of A, where the former is a work register and the latter a query register to H.

Suppose one runs a compressed oracle simulation of A on some initial state $|\psi\rangle$. We prove an intuitive lemma that directly relates the probability of the final database register containing a particular query x^* to the probability of finding the register query in the state x^*, if this were to be measured before a uniformly selected query. A bit more precisely, we show that if the final compressed oracle state has weight Δ on databases containing a particular query x^*, then if one were to run A and measure register query before one of the q queries, selected uniformly at random, the measurement outcome would be x^* with probability at least Δ/q. In fact, we show an even more general statement that will be useful in our proof, which lower bounds the probability that measuring a uniformly random query yields x^*, and that decompressing the database everywhere yields a particular H.

We denote by Decomp the unitary that decompresses the database at every point. Formally, Decomp applies $\mathsf{StdDecomp_x}$ for every x. For a set $S \subseteq \{0,1\}^n$, denote by $\mathcal{F}(\{0,1\}^n \setminus S, \{0,1\})$ the set of functions from $\{0,1\}^n \setminus S$ to $\{0,1\}$. For $\tilde{H} \in \mathcal{F}(\{0,1\}^n \setminus S, \{0,1\})$, let $\Pi_{\tilde{H}}$, acting on the (decompressed) database register, be the projector onto functions H that are consistent with $\tilde{H}$ outside of S. Formally,

$$\Pi_{\tilde{H}} := \sum_{H : H|_{\{0,1\}^n \setminus S} = \tilde{H}} |H\rangle \langle H| \,,$$

where here we are implicitly identifying databases with the functions they specify.

For convenience, we will abuse notation slightly and write $D \ni x$ to mean that D contains a pair (x, w) for some w. Moreover, for $x \in \{0,1\}$, let $\Pi_{D\ni x}$, acting on the compressed database register D, be the projector onto databases containing x, i.e.

$$\Pi_{D\ni x} = \sum_{D \ni x} |D\rangle \langle D|$$

Without loss of generality, we let A be the algorithm that applies the unitary $(UO)^q$ followed by a measurement of an output register, where U acts on work, query and O represents the oracle call, which we think of as acting on query, and an "oracle register" O containing the description of H. When running a compressed oracle simulation of A, the unitary O is replaced by the compressed oracle call O^{comp}, where $\mathsf{Decomp} \circ O^{\mathsf{comp}} = O \circ \mathsf{Decomp}$.

Denote by $\mathcal{X}$ the domain of H. In what follows, we use the following notation. For $D \subseteq \mathcal{X}$, we let

$$|D\rangle := \sum_{w_x \in \{0,1\} : x \in D} (-1)^{w_x} |\{(x, w_x) : x \in D\rangle$$

Denote by $\mathcal{S}_{\mathsf{comp}}$ the set of all (normalized) states of the form:

$$\sum_{z,x,e,D} \alpha_{z,x,e,D} |z\rangle_{\mathsf{work}} |x, e\rangle_{\mathsf{query}} |D\rangle_{\mathsf{O}} \,. \tag{42}$$

These are states that can be reached by running a compressed oracle simulation.

Lemma 108. *Let $x^* \in \{0,1\}^n$. Let $S \subseteq \{0,1\}^n$ be such that $x^* \in S$. Let $\tilde{H} \in \mathcal{F}(\{0,1\}^n \setminus S, \{0,1\})$. Let $|\Psi_0\rangle \in \mathcal{S}_{\mathsf{comp}}$. Let $|\Psi_{final}\rangle = (UO^{\mathsf{comp}})|\Psi_0\rangle$. Let*

$$\Delta_{0,x^*,\tilde{H}} := \| \Pi_{\tilde{H}} \mathsf{Decomp}\, \Pi_{D\ni x^*} |\Psi_0\rangle \|^2 \,,$$

and let

$$\Delta_{final,x^*,\tilde{H}} := \| \Pi_{\tilde{H}} \mathsf{Decomp}\, \Pi_{D\ni x^*} |\Psi_{final}\rangle \|^2 \,.$$

Then,

$$\mathbb{E}_{l\leftarrow\{0,\dots,q-1\}} \| \, |x^*\rangle \langle x^*| \, (UO)^l \Pi_{\tilde{H}} \mathsf{Decomp} \, |\Psi_0\rangle \|^2 \geq \frac{1}{q} (\Delta_{final,x^*,\tilde{H}} - \Delta_{0,x^*,\tilde{H}}) \,.$$

The special case where $S = \{0,1\}^n$ gives the following corollary.

Corollary 109. *Let $x^* \in \{0,1\}^n$. Let $|\Psi_0\rangle \in \mathcal{S}_{\mathsf{comp}}$. Let $|\Psi_{final}\rangle = (UO^{\mathsf{comp}})|\Psi_0\rangle$. Let*

$$\Delta_{0,x^*} := \| \Pi_{D\ni x^*} |\Psi_0\rangle \|^2 \,,$$

and let

$$\Delta_{final,x^*} := \| \Pi_{D \ni x^*} \left| \Psi_{final} \right\rangle \|^2 .$$

Then,

$$\mathbb{E}_{l \leftarrow \{0,\dots,q-1\}} \| \left| x^* \right\rangle \left\langle x^* \right| (UO)^l \mathsf{Decomp} \left| \Psi_0 \right\rangle \|^2 \geq \frac{1}{q} (\Delta_{final,x^*} - \Delta_{0,x^*}) .$$

Proof of Lemma 108. For the rest of the section, we write $\mathbb{E}_l$ as short for $\mathbb{E}_{l \leftarrow \{0,\dots,q-1\}}$. Using the fact that $\Pi_{\tilde{H}}$ commutes with both $(UO)^l$ and $\left| x^* \right\rangle \left\langle x^* \right|$, that $\mathsf{Decomp} \circ O^{\mathsf{comp}} = O \circ \mathsf{Decomp}$, and that $\left| x^* \right\rangle \left\langle x^* \right|$ commutes with Decomp, we have that

$$\begin{aligned} & \mathbb{E}_l \| \left| x^* \right\rangle \left\langle x^* \right| (UO)^l \Pi_{\tilde{H}} \mathsf{Decomp} \left| \Psi_0 \right\rangle \|^2 \\ = & \mathbb{E}_l \| \left| x^* \right\rangle \left\langle x^* \right| \Pi_{\tilde{H}} \mathsf{Decomp} (UO^{\mathsf{comp}})^l \left| \Psi_0 \right\rangle \|^2 \\ = & \mathbb{E}_l \| \Pi_{\tilde{H}} \mathsf{Decomp} \left| x^* \right\rangle \left\langle x^* \right| (UO^{\mathsf{comp}})^l \left| \Psi_0 \right\rangle \|^2 \end{aligned} \tag{43}$$

We can write the state $(UO^{\mathsf{comp}})^l \left| \Psi_0 \right\rangle$ as

$$(UO^{\mathsf{comp}})^l \left| \Psi_0 \right\rangle = \sum_{\substack{z,x,e \\ D:|D| \leq l}} \alpha^l_{z,x,e,D} \left| z \right\rangle_{\mathsf{work}} \left| x, e \right\rangle_{\mathsf{query}} \left| D \right\rangle . \tag{44}$$

for some $\alpha^l_{z,x,e,D}$. For brevity, we will denote by D^l a database with at most l pairs.

Then, by Equation (43), we have

$$\begin{aligned} & \mathbb{E}_l \| \left| x^* \right\rangle \left\langle x^* \right| (UO)^l \Pi_{\tilde{H}} \mathsf{Decomp} \left| \Psi_0 \right\rangle \|^2 \\ = & \mathbb{E}_l \left\| \Pi_{\tilde{H}} \mathsf{Decomp} \sum_{z,e,D^l} \alpha^l_{z,x^*,e,D^l} \left| z, x^*, e \right\rangle \left| D^l \right\rangle \right\|^2 . \end{aligned} \tag{45}$$

Now, notice that, for any $D \not\ni x^*$ and $D' \ni x^*$, we have $\Pi_{\tilde{H}} \mathsf{Decomp} \left| D \right\rangle \perp \Pi_{\tilde{H}} \mathsf{Decomp} \left| D' \right\rangle$. This is because we can write

$$\mathsf{Decomp} \left| D \right\rangle = \bigotimes_{x \in D} \left| - \right\rangle_{\mathsf{x}} \otimes \bigotimes_{x \notin D} \left| + \right\rangle_{\mathsf{x}} = \left| + \right\rangle_{\mathsf{x}^*} \otimes \bigotimes_{\substack{x \in D \\ x \neq x^*}} \left| - \right\rangle_{\mathsf{x}} \otimes \bigotimes_{\substack{x \notin D \\ x \neq x^*}} \left| + \right\rangle_{\mathsf{x}} ,$$

and

$$\mathsf{Decomp} \left| D' \right\rangle = \bigotimes_{x \in D'} \left| - \right\rangle_{\mathsf{x}} \otimes \bigotimes_{x \notin D'} \left| + \right\rangle_{\mathsf{x}} = \left| - \right\rangle_{\mathsf{x}^*} \otimes \bigotimes_{\substack{x \in D' \\ x \neq x^*}} \left| - \right\rangle_{\mathsf{x}} \otimes \bigotimes_{\substack{x \notin D' \\ x \neq x^*}} \left| + \right\rangle_{\mathsf{x}} ,$$

where x denotes the sub-register of the decompressed database register corresponding to the value of the oracle at x. Finally, notice that $\Pi_{\tilde{H}}$ acts as the identity on register x^*, since $\tilde{H} \in \mathcal{F}(\{0,1\}^n \setminus S, \{0,1\})$ and $x^* \in S$. Thus, $\Pi_{\tilde{H}} \mathsf{Decomp} \left| D \right\rangle$ and $\Pi_{\tilde{H}} \mathsf{Decomp} \left| D' \right\rangle$ are orthogonal, since they are orthogonal on register x^*.

Then, we have

$$\begin{aligned} \text{Equation (45)} = & \; \mathbb{E}_l \left\| \Pi_{\tilde{H}} \mathsf{Decomp} \sum_{\substack{z,e, \\ D^l \ni x^*}} \alpha^l_{z,x^*,e,D^l} \left| y, x^*, e \right\rangle \left| D^l \right\rangle \right\|^2 \\ & + \mathbb{E}_l \left\| \Pi_{\tilde{H}} \mathsf{Decomp} \sum_{\substack{z, \\ D^l \not\ni x^*}} \alpha^l_{z,x^*,e=0,D^l} \left| z, x^*, e = 0 \right\rangle \left| D^l \right\rangle \right\|^2 \\ & + \mathbb{E}_l \left\| \Pi_{\tilde{H}} \mathsf{Decomp} \sum_{\substack{z, \\ D^l \not\ni x^*}} \alpha^l_{z,x^*,e=1,D^l} \left| z, x^*, e = 1 \right\rangle \left| D^l \right\rangle \right\|^2 \\ & \geq \mathbb{E}_l \left\| \Pi_{\tilde{H}} \mathsf{Decomp} \sum_{\substack{z, \\ D^l \not\ni x^*}} \alpha^l_{z,x^*,e=1,D^l} \left| z, x^*, e = 1 \right\rangle \left| D^l \right\rangle \right\|^2 \end{aligned} \tag{46}$$

where the first equality is due to the fact that components with $D \not\ni x^*$ and with $D \ni x^*$ are orthogonal, and, of course, components with $e = 0$ and with $e = 1$ are also orthogonal.

We will prove the following lemma.

Lemma 110.

$$\mathbb{E}_l \left\| \Pi_{\tilde{H}} \mathsf{Decomp} \sum_{\substack{z \\ D^l \not\ni x^*}} \alpha^l_{z,x^*,e=1,D^l} \left| z, x^*, e=1 \right\rangle \left| D^l \right\rangle \right\|^2 \geq \frac{1}{q} \left(\Delta_{final,x^*,\tilde{H}} - \Delta_{0,x^*,\tilde{H}} \right). \tag{47}$$

Combining Equation (45), Equation (46), and Lemma (110) immediately yields Lemma 108. Thus, to conclude the proof of Lemma 108, we are left with proving Lemma 110.

Proof. Notice, via a telescopic sum, that

$$\begin{aligned}
&\mathbb{E}_{l \leftarrow \{0,\ldots,q-1\}} \Bigg[\left\| \Pi_{\tilde{H}} \mathsf{Decomp} \sum_{\substack{z,x,e, \\ D^{l+1} \ni x^*}} \alpha^{l+1}_{z,x,e,D^{l+1}} \left| z,x,e \right\rangle \left| D^{l+1} \right\rangle \right\|^2 \\
&\qquad - \left\| \Pi_{\tilde{H}} \mathsf{Decomp} \sum_{\substack{z,x,e, \\ D^l \ni x^*}} \alpha^l_{z,x,e,D^l} \left| z,x,e \right\rangle \left| D^l \right\rangle \right\|^2 \Bigg] \\
&\geq \frac{1}{q} \Bigg(\left\| \Pi_{\tilde{H}} \mathsf{Decomp} \sum_{\substack{z,x,e, \\ D \ni x^*}} \alpha^q_{z,x,e,D} \left| z,x,e \right\rangle \left| D \right\rangle \right\|^2 \\
&\qquad - \left\| \Pi_{\tilde{H}} \mathsf{Decomp} \sum_{\substack{z,x,e, \\ D \ni x^*}} \alpha^0_{z,x,e,D} \left| z,x,e \right\rangle \left| D \right\rangle \right\|^2 \Bigg) \\
&= \frac{1}{q} \left(\Delta_{final,x^*,\tilde{H}} - \Delta_{0,x^*,\tilde{H}} \right).
\end{aligned} \tag{48}$$

For convenience, we will denote the quantities inside the square brackets on the LHS of Equation (48) as X_{l+1} and X_l.

Then,

$$\begin{aligned}
X_{l+1} :=& \left\| \Pi_{\tilde{H}} \mathsf{Decomp} \sum_{\substack{z,x,e, \\ D^{l+1} \ni x^*}} \alpha^{l+1}_{z,x,e,D^{l+1}} \left| z,x,e \right\rangle \left| D^{l+1} \right\rangle \right\|^2 \\
=& \left\| \left(U^{-1} \otimes \Pi_{\tilde{H}} \mathsf{Decomp} \right) \sum_{\substack{z,x,e, \\ D^{l+1} \ni x^*}} \alpha^{l+1}_{z,x,e,D^{l+1}} \left| z,x,e \right\rangle \left| D^{l+1} \right\rangle \right\|^2 \\
=& \Bigg\| \Pi_{\tilde{H}} \mathsf{Decomp} \circ \ O^{\mathsf{comp}} \Bigg[\sum_{\substack{z,x, \\ D^l \ni x^*}} \alpha^l_{z,x,e=0,D^l} \left| z,x,e=0 \right\rangle \left| D^l \right\rangle \\
&\qquad + \sum_{\substack{z,x \neq x^*,w, \\ D^l \ni x^*}} \alpha^l_{z,x,e=1,D^l} \left| z,x,e=1 \right\rangle \left| D^l \right\rangle \\
&\qquad + \sum_{\substack{z, \\ D^l \not\ni x^*}} \alpha^l_{z,x^*,e=1,D^l} \left| z,x^*,e=1 \right\rangle \left| D^l \right\rangle \Bigg] \Bigg\|^2
\end{aligned} \tag{49}$$

where the last equality follows from the definition of the compressed oracle call O^{comp} and the α^l coefficients. In words, the three terms in the last expression correspond to the three ways in which a database containing x^* after the $(l+1)$-th query can originate.

Using the fact that $\mathsf{Decomp} \circ O^{\mathsf{comp}} = O \circ \mathsf{Decomp}$, and that $\Pi_{\tilde{H}}$ commutes with O (since $\Pi_{\tilde{H}}$ is diagonal in the control basis of O), we have

$$\begin{aligned}
Equation(49) =& \Bigg\| O \circ \left(\Pi_{\tilde{H}} \mathsf{Decomp}\right) \Bigg[\sum_{\substack{z,x,\\ D^l \ni x^*}} \alpha^l_{z,x,e=0,D^l} \left|z, x, e = 0\right\rangle \left|D^l\right\rangle \\
&+ \sum_{\substack{y,x \neq x^*,w,\\ D^l \ni x^*}} \alpha^l_{z,x,e=1,D^l} \left|z, x, e = 1\right\rangle \left|D^l\right\rangle \\
&+ \sum_{\substack{z,\\ D^l \not\ni x^*}} \alpha^l_{z,x^*,e=1,D^l} \left|z, x^*, e = 1\right\rangle \left|D^l\right\rangle \Bigg] \Bigg\|^2 \\
=& \Bigg\| \Pi_{\tilde{H}} \mathsf{Decomp} \Bigg[\sum_{\substack{z,x,\\ D^l \ni x^*}} \alpha^l_{z,x,e=0,D^l} \left|z, x, e = 0\right\rangle \left|D^l\right\rangle \\
&+ \sum_{\substack{z,x \neq x^*,\\ D^l \ni x^*}} \alpha^l_{z,x,e=1,D^l} \left|z, x, e = 1\right\rangle \left|D^l\right\rangle \\
&+ \sum_{\substack{y,w,\\ D^l \not\ni x^*}} \alpha^l_{z,x^*,e=1,D^l} \left|z, x^*, e = 1\right\rangle \left|D^l\right\rangle \Bigg] \Bigg\|^2 \\
=& \Bigg\| \Pi_{\tilde{H}} \mathsf{Decomp} \sum_{\substack{z,x,\\ D^l \ni x^*}} \alpha^l_{z,x,e=0,D^l} \left|z, x, e = 0\right\rangle \left|D^l\right\rangle \Bigg\|^2 \\
+& \Bigg\| \Pi_{\tilde{H}} \mathsf{Decomp} \sum_{\substack{z,x \neq x^*,\\ D^l \ni x^*}} \alpha^l_{z,x,e=1,D^l} \left|z, x, e = 1\right\rangle \left|D^l\right\rangle \Bigg\|^2 \\
+& \Bigg\| \Pi_{\tilde{H}} \mathsf{Decomp} \sum_{\substack{z,\\ D^l \not\ni x^*}} \alpha^l_{z,x^*,e=1,D^l} \left|z, x^*, e = 1\right\rangle \left|D^l\right\rangle \Bigg\|^2, \qquad (50)
\end{aligned}$$

where the last equality is because the three terms in the sum are orthogonal.

Now,

$$\begin{aligned}
X_l :=& \Bigg\| \Pi_{\tilde{H}} \mathsf{Decomp} \sum_{\substack{z,x,e,\\ D^l \ni x^*}} \alpha^l_{z,x,e,D^l} \left|z, x, e\right\rangle \left|D^l\right\rangle \Bigg\|^2 \\
=& \Bigg\| \Pi_{\tilde{H}} \mathsf{Decomp} \sum_{\substack{z,x,\\ D^l \ni x^*}} \alpha^l_{z,x,e=0,D^l} \left|z, x, e = 0\right\rangle \left|D^l\right\rangle \Bigg\|^2 \\
+& \Bigg\| \Pi_{\tilde{H}} \mathsf{Decomp} \sum_{\substack{z,x \neq x^*,\\ D^l \ni x^*}} \alpha^l_{z,x,e=1,D^l} \left|z, x, e = 1\right\rangle \left|D^l\right\rangle \Bigg\|^2 \\
+& \Bigg\| \Pi_{\tilde{H}} \mathsf{Decomp} \sum_{z, D^l \ni x^*} \alpha^l_{z,x^*,e=1,D^l} \left|z, x^*, e = 1\right\rangle \left|D^l\right\rangle \Bigg] \Bigg\|^2 \qquad (51)
\end{aligned}$$

Equations Equation (50) and Equation (51) imply

$$\begin{aligned}
&\Bigg\| \Pi_{\tilde{H}} \mathsf{Decomp} \sum_{z, D^l \not\ni x^*} \alpha^l_{y,x^*,e=1,D^l} \left|z, x^*, e = 1\right\rangle \left|D^l\right\rangle \Bigg\|^2 \qquad (52) \\
&\geq X^{l+1} - X^l \,. \qquad (53)
\end{aligned}$$

Thus, we have

$$\begin{aligned}
&\mathbb{E}_l \left\| \Pi_{\tilde{H}} \mathsf{Decomp} \sum_{\substack{z \\ D^l \not\ni x^*}} \alpha^l_{y,x^*,e=1,D^l} \left|z, x^*, e = 1\right\rangle \left|D^l\right\rangle \right\|^2 && (54)\\
&\geq \mathbb{E}_l[X^{l+1} - X^l] && (55)\\
&\geq \frac{1}{q}(\Delta_{final,x^*,\tilde{H}} - \Delta_{0,x^*,\tilde{H}}) \,. && (56)
\end{aligned}$$

where the last line is from Equation (48). This concludes the proof of Lemma 110, and thus the proof of Lemma 108.

□

□

11.4.2 The structure of strategies that produce valid equations

In this section, we prove properties about the structure of strategies that succeed at $\mathsf{subProblem}$. We will later leverage these properties to argue that any algorithm in $\mathcal{W}_d$ (where recall that $\mathcal{W}_d$ was defined before Lemma 104) that succeeds at $\mathsf{subProblem}$ with non-negligible advantage implies there exists an efficient an algorithm to extract collisions of G_0, G_1. We emphasize that all of the results in this subsection hold for any algorithm that makes a polynomially-bounded number of queries to G_0, G_1 and H. Only later in Subsection 11.4.3, we will make use of the additional structure of algorithms in $\mathcal{W}_d$.

Let $H : \{0,1\}^n \to \{0,1\}$ be a uniformly random oracle. Let $\mathcal{S}_{\mathsf{comp}}$ be the set of compressed oracle states on registers $\mathsf{Y}, \mathsf{D}, \mathsf{M}, \mathsf{AUX}, \mathsf{O}$, where Y, D, M correspond to outputs y, d, m[68] for $\mathsf{subProblem}$, AUX includes auxiliary registers, input registers, and query registers, and O is the compressed database register for H. Formally, $\mathcal{S}_{\mathsf{comp}}$ is defined as in Equation (42), except with a different naming of the registers.

Fix oracles G_0, G_1, h for $\mathsf{subProblem}$. Let $y \in \mathsf{TwoToOne}(G_0, G_1)$. For $b \in \{0,1\}$, we denote $\tilde{x}^y_b := (x^y_b, h(y))$.

Let Π_{valid}, acting on decompressed databases, be the projector onto valid equations, i.e.

$$\Pi_{\mathsf{valid}} := \sum_{\substack{y,d,m,D: \\ m = d \cdot (x^y_0 \oplus x^y_1) \oplus D(\tilde{x}^y_0) \oplus D(\tilde{x}^y_1)}} \left|y, d, m\right\rangle \left\langle y, d, m\right| \otimes \left|D\right\rangle \left\langle D\right| \,.$$

We invoke the following "structure" theorem, adapted from [CGV22]. We will then extend this structure theorem in Lemma 112.

Note that, in general $\tilde{x}^y_b$ could be any function of y, and the following structure theorem would hold verbatim. However, for concreteness, we consider $\tilde{x}^y_b = (x^y_b, h(y))$ as this is the relevant choice for $\mathsf{subProblem}$.

When a state $\left|\Psi\right\rangle \in \mathcal{S}_{\mathsf{comp}}$ is clear from the context, we denote

$$\Pr[\mathsf{win}] := \left\| \Pi_{\mathsf{valid}} \mathsf{Decomp} \left|\Psi\right\rangle \right\|^2 ,$$

and we denote

$$\Pr[\mathsf{win}|\, y] := \frac{\left\| \Pi_{\mathsf{valid}} \mathsf{Decomp} \left|y\right\rangle \left\langle y\right| \left|\Psi\right\rangle \right\|^2}{\left\| \left|y\right\rangle \left\langle y\right| \left|\Psi\right\rangle \right\|^2} \,. \qquad (57)$$

As earlier, denote by O the unitary that performs an oracle query, and by O^{comp} the compressed oracle version of it.

Lemma 111 (Adapted from [CGV22]). *Fix G_0, G_1, h. Let $\left|\Psi\right\rangle \in \mathcal{S}_{\mathsf{comp}}$. Suppose*

$$\left|\Psi\right\rangle = \sum_{y,d,m,aux,D} \alpha_{y,d,m,aux,D} \left|y, d, m, aux\right\rangle \left|D\right\rangle \,.$$

Let $y^* \in \mathsf{TwoToOne}(G_0, G_1)$. Let x_0, x_1 be such that $G_0(x_0) = G_1(x_1) = y^*$. Let $\tilde{x}_0 = (x_0, h(y^*))$ and $\tilde{x}_1 = (x_1, h(y^*))$. Let $\epsilon := \Pr[\mathsf{win}|\, y^*] - \frac{1}{2}$. Suppose, for some $\delta \geq 0$, that

$$\sum_{\substack{d,m,aux, \\ D \ni \tilde{x}_0, \tilde{x}_1}} \left|\alpha_{y^*,d,m,aux,D}\right|^2 \leq \delta \cdot \left\| \left|y^*\right\rangle \left\langle y^*\right| \left|\Psi\right\rangle \right\|^2 .$$

Then,

[68] From here on, for the rest of the proof, we switch notation and denote the outputs of the algorithm by y, d, m instead of y, r, m.

(i)

$$\sum_{\substack{d,m,aux,\\ |D\cap\{\tilde{x}_0,\tilde{x}_1\}|=1}} |\alpha_{y^*,d,m,aux,D}|^2 \geq 2(\varepsilon-\sqrt{\delta})\cdot \| \, |y^*\rangle\langle y^*|\,|\Psi\rangle \, \|^2 \,.$$

(ii)

$$\sum_{\substack{d,m,aux,\\ D\not\ni\tilde{x}_0,\tilde{x}_1}} \left|\alpha_{y^*,d,m,aux,D\cup\{\tilde{x}_0\}} - \alpha_{y,d,m,aux,D\cup\{\tilde{x}_1\}}\right|^2 \leq \sum_{\substack{d,m,aux,\\ |D\cap\{\tilde{x}_0,\tilde{x}_1\}|=1}} |\alpha_{y^*,d,m,aux,D}|^2 - 2(\varepsilon-\sqrt{\delta})\,.$$

Proof. This is a simple adaptation of the proof of a similar lemma in [CGV22]. □

We now prove a refinement of the structural property about strategies that produce valid equations, by combining Lemma 108 with Lemma 111. The following lemma essentially establishes that strategies that are successful at producing valid equations are such that, with high probability over oracles H, the algorithm queries H at a superposition of pre-images the output y. In what follows, for $\tilde{H}\in\mathcal{F}(S,\{0,1\})$, we denote by $|\tilde{H}\rangle\langle\tilde{H}|$ the projector onto oracles H such that $H|_S=\tilde{H}$. Let $\Pi_{\tilde{H}} := \mathsf{Decomp}^{-1}|\tilde{H}\rangle\langle\tilde{H}|\,\mathsf{Decomp}$. Moreover, recall the notation $\Pr[\mathsf{win}|y]$ from Equation (57).

In the following Lemma, $\Xi : [0,\frac{1}{2}]\times[0,1]\times[0,1]\to[0,1]$ is a function with the following properties. Suppose $\delta_1,\delta_2:\mathbb{N}\to[0,1]$ are non-negligible functions. Then,

- If $\varepsilon_1:\mathbb{N}\to[0,\frac{1}{2}]$ is a non-negligible function, then $1-\Xi(\varepsilon_1,\delta_1,\delta_2)$ is a non-negligible function.
- There exists a constant $c>0$ such that, for any $\mu\in[0,\frac{1}{2}]$,

$$\Xi\left(\frac{1}{2}-\mu,\delta_1,\delta_2\right)\leq\mu^c\,.$$

The exact form of Ξ is given in Equation (72).

Lemma 112. *Fix any G_0,G_1,h. Let $|\Psi_0\rangle\in S_{\mathrm{comp}}$. Suppose $|\Psi_0\rangle=\sum_{y,d,m,aux,D}\beta_{y,d,m,aux,D}\,|y,d,m,aux\rangle\,|D\rangle$. Let $y^*\in\mathsf{TwoToOne}(G_0,G_1)$. Let x_0,x_1 be such that $G_0(x_0)=G_1(x_1)=y^*$. Let $\tilde{x}_0=(x_0,h(y^*))$ and $\tilde{x}_1=(x_1,h(y^*))$. Let*

$$\delta_1 := \sum_{\substack{y,d,m,aux\\ D\ni\tilde{x}_0\text{ or }\tilde{x}_1}} |\beta_{y,d,m,aux,D}|^2\,.$$

Let U be a local unitary, and O^{comp} a compressed oracle call, and $q\in\mathbb{N}$. Let

$$|\Psi_{\mathrm{final}}\rangle=(UO^{\mathrm{comp}})^q\,|\Psi_0\rangle=\sum_{y,d,m,aux,D}\alpha_{y,d,m,aux,D}\,|y,d,m,aux\rangle\,|D\rangle\,.$$

Let $\varepsilon_1:=\Pr[\mathsf{win}|y^*]-\frac{1}{2}$. Let $\delta_2:=\sum_{\substack{d,m,aux\\ D\ni\tilde{x}_0,\tilde{x}_1}}|\alpha_{y^*,d,m,aux,D}|^2/\|\,|y^*\rangle\langle y^*|\,|\Psi_{\mathrm{final}}\rangle\,\|^2$. Then, there exists $\mathcal{H}_{good}\subseteq\mathcal{F}(\{0,1\}^n\smallsetminus\{\tilde{x}_0,\tilde{x}_1\},\{0,1\})$ such that

(i)

$$\sum_{\tilde{H}\in\mathcal{H}_{good}}\frac{\|\Pi_{\tilde{H}}\,|y^*\rangle\langle y^*|\,|\Psi_{final}\rangle\,\|^2}{\|\,|y^*\rangle\langle y^*|\,|\Psi_{final}\rangle\,\|^2}\geq 1-\Xi(\varepsilon_1,\delta_1,\delta_2)\,.$$

(ii) for all $\tilde{H}\in\mathcal{H}_{good}$, $b\in\{0,1\}$,

$$\frac{\mathbb{E}_{l\leftarrow[q]}[\|\,|x_b\rangle\langle x_b|\,(UO)^l\,|\tilde{H}\rangle\langle\tilde{H}|\,\mathsf{Decomp}\,|\Psi_0\rangle\,\|^2]}{\|\,|y^*\rangle\langle y^*|\,(UO)^q\,|\tilde{H}\rangle\langle\tilde{H}|\,\mathsf{Decomp}\,|\Psi_0\rangle\,\|^2}\geq\frac{1}{2q}\cdot(1-\Xi(\varepsilon_1,\delta_1,\delta_2))\,,$$

As a special case, Lemma 112 gives the following characterization of strategies that succeed at the proof of quantumness of [BKVV20]. In the following, denote by $G_0,G_1:\mathcal{X}\to\mathcal{Y}$ the pair of trapdoor claw-free functions used in the proof of quantumness[69]. Denote by H the random oracle. For a set $S\subseteq\mathcal{X}$, denote by $\mathcal{F}(S,\{0,1\})$ the set of all functions from S to $\{0,1\}$. Denote by $H|_S$ the restriction of H to domain S. Moreover,

[69] Our characterization applies equally when G_0,G_1 are a pair of uniformly random permutations. It suffices for our characterization that it is hard to find collisions between G_0 and G_1.

for an oracle algorithm A, and $l \in \mathbb{N}$, denote by $\mathsf{Ext}_l(A)$ the oracle algorithm that runs A up until right before the l-th query, and outputs the outcome of measuring the query register. We include a subscript λ when we intend to make the dependence on the security parameter explicit. In the following Lemma, $\Xi' : [0, \frac{1}{2}] \to [0,1]$ are functions with the following properties.

- If $\varepsilon : \mathbb{N} \to [0, \frac{1}{2}]$ is a non-negligible function, then $1 - \Xi'(\varepsilon)$ is also a non-negligible function.
- There exists a constant $c > 0$ such that, for any $\mu \in [0, \frac{1}{2}]$,
$$\Xi'\left(\frac{1}{2} - \mu\right) \leq \mu^c \,.$$

Corollary 113 (Structure theorem for BKVV)**.** *Let A be an algorithm that succeeds at the (single-copy) proof of quantumness of [BKVV20] with probability $1-\mu$, where μ is a function of the security parameter such that $1-\mu$ is at least non-negligibly greater than $\frac{1}{2}$. Then, there exists a negligible function* negl *such that, for all λ, there exists a set $\mathcal{Y}'_\lambda \subseteq \mathcal{Y}_\lambda$ such that*

- $\Pr[y \in \mathcal{Y}'_\lambda : (y,d,m) \leftarrow A^H_\lambda] \geq 1 - \Xi'(\mu(\lambda)) - \mathsf{negl}(\lambda)$.
- *For all $y \in \mathcal{Y}'_\lambda$ the following holds. Let $x_0 = G_0^{-1}(y)$ and $x_1 = G_1^{-1}(y)$. Let $S = \mathcal{X} \setminus \{x_0, x_1\}$. Then, there exists a set $\mathcal{H}_{good} \subseteq \mathcal{F}(S, \{0,1\})$ such that*
$$\Pr[H|_S \in \mathcal{H}_{good} | A^H_\lambda \text{ outputs } y] \geq 1 - \Xi'(\mu(\lambda)) - \mathsf{negl}(\lambda) \,.$$
Moreover, for all $\tilde{H} \in \mathcal{H}_{good}$, $b \in \{0,1\}$,
$$\mathbb{E}_{l \leftarrow [q]}\left[\Pr[H|_S = \tilde{H} \,\wedge\, \mathsf{Ext}^H_l(A) \text{ outputs } x_b]\right] \geq \frac{1}{2q} \cdot \big(1 - \Xi'(\mu(\lambda)) - \mathsf{negl}(\lambda)\big) \cdot \Pr[H|_S = \tilde{H} \,\wedge\, A^H_\lambda \text{ outputs } y] \,.$$

Proof. First, notice that, when considering the proof of quantumness from BKVV, there is no function h. So, in Lemma 112, one can take $\tilde{x_0} = x_0$ and $\tilde{x_1} = x_1$.

Since by hypothesis $\Pr[A_\lambda \text{ wins}] \geq \frac{1}{2} + \varepsilon$, we deduce by an averaging argument that, for all λ, there exists a set $\mathcal{Y}'_\lambda \subseteq \mathcal{Y}_\lambda$ such that

(a) $\Pr[y \in \mathcal{Y}'_\lambda : (y,d,m) \leftarrow A^H_\lambda] \geq 1 - \sqrt{1-2\varepsilon}$.

(b) For all $y \in \mathcal{Y}'_\lambda$, $\Pr[A_\lambda \text{ wins} | A_\lambda \text{ outputs } y] \geq 1 - \sqrt{1-2\varepsilon}$.

Denote by $\mathsf{Y}, \mathsf{D}, \mathsf{M}, \mathsf{AUX}$ the register on which A acts. Consider a compressed oracle simulation of A and additionally denote by O the compressed oracle register. Let $|\Psi_0\rangle = |0\rangle_{\mathsf{Y},\mathsf{D},\mathsf{M},\mathsf{AUX}} |D = \varnothing\rangle_{\mathsf{O}}$ be the initial state of a compressed oracle simulation of A. Let $|\Psi_{final}\rangle = \sum_{y,d,m,aux,D} \alpha_{y,d,m,aux,D} |y,d,m,aux\rangle |D\rangle$ be the final state of a compressed simulation of A, right before the final measurement. Notice that there must exist a negligible function negl, such that, for all λ, there exists a set $\mathcal{Y}''_\lambda \subseteq \mathcal{Y}'_\lambda$ such that:

- $\Pr[y \in \mathcal{Y}''_\lambda : (y,d,m) \leftarrow A^H_\lambda] \geq 1 - \sqrt{1-2\varepsilon} - \mathsf{negl}(\lambda)$,
- for all $y^* \in \mathcal{Y}''_\lambda$,
$$\sum_{\substack{d,m,aux \\ D \ni \tilde{x}_0, \tilde{x}_1}} |\alpha_{y^*,d,m,aux,D}|^2 / \| \, |y^*\rangle \langle y^*| \, |\Psi_{\text{final}}\rangle \, \|^2 \leq \mathsf{negl}(\lambda) \,. \tag{58}$$

Suppose for a contradiction that the above were not the case, then it is easy to see that by running a compressed simulation of A, and measuring the database register, one finds a collision with non-negligible probability.

Now, fix any λ and any $y \in \mathcal{Y}''_\lambda$. Using the notation from Lemma 112, we invoke Lemma 112 with $y^* = y$, and:

- $\delta_1 = 0$, which holds since the database is empty in $|\Psi_0\rangle$.
- $\varepsilon_1 = 1 - \sqrt{1-2\varepsilon} - \frac{1}{2} = \frac{1}{2} - \sqrt{1-2\varepsilon}$, which holds by condition (b), since $\mathcal{Y}''_\lambda \subseteq \mathcal{Y}'_\lambda$,
- $\delta_2 = \mathsf{negl}(\lambda)$, which we established in (58).

It is straightforward to verify that one obtains a function $\Xi'(\varepsilon)$ with the desired properties. □

The crux in proving Lemma 112 is to prove the following.

Lemma 114. *Let* $|\Psi\rangle = \sum_{y,d,m,aux,D} \alpha_{y,d,m,aux,D} |y,d,m,aux\rangle |D\rangle$*. Fix* y^*, x_0, x_1*. Let* $\tilde{x}_0, \tilde{x}_1$ *as in Lemma 111. Suppose, for some* $\mu_2 > 0$*,*

$$\sum_{\substack{d,m,aux,\\ D\not\ni\tilde{x}_0,\tilde{x}_1}} \left|\alpha_{y^*,d,m,aux,D\cup\{\tilde{x}_0\}} - \alpha_{y^*,d,m,aux,D\cup\{\tilde{x}_1\}}\right|^2 \leq (1-\mu_2) \sum_{\substack{d,m,aux,\\ |D\cap\{\tilde{x}_0,\tilde{x}_1\}|=1}} \left|\alpha_{y^*,d,m,aux,D}\right|^2 . \tag{59}$$

For $b \in \{0,1\}$*, let* $|\phi_b\rangle$ *be the un-normalized state*

$$\begin{aligned} |\phi_b\rangle :&= \sum_{\substack{d,m,aux\\ D\ni\tilde{x}_b \wedge D\not\ni\tilde{x}_{\bar{b}}}} \alpha_{y^*,d,m,aux,D} |y^*,d,m,aux\rangle |D\rangle \\ &= \sum_{\substack{d,m,aux\\ D\not\ni\tilde{x}_0,\tilde{x}_1}} \alpha_{y^*,d,m,aux,D\cup\{x_b\}} |y^*,d,m,aux\rangle |D\cup\tilde{x}_b\rangle \ . \end{aligned}$$

Then, there exists $\mathcal{H}_{good} \subseteq \mathcal{F}(\{0,1\}^n \setminus \{\tilde{x}_0,\tilde{x}_1\},\{0,1\})$ *such that*

(i)

$$\sum_{\tilde{H}\in\mathcal{H}_{good}} \frac{\|\Pi_{\tilde{H}}|\phi_0\rangle\|^2 + \|\Pi_{\tilde{H}}|\phi_1\rangle\|^2}{\||\phi_0\rangle\|^2 + \||\phi_1\rangle\|^2} \geq 1 - \sqrt{1-\mu_2} ,$$

(ii) for all $\tilde{H} \in \mathcal{H}_{good}$*,* $b \in \{0,1\}$*,*

$$\frac{\left\|\Pi_{\tilde{H}}|\phi_b\rangle\right\|^2}{\|\Pi_{\tilde{H}}|\phi_0\rangle\|^2 + \|\Pi_{\tilde{H}}|\phi_1\rangle\|^2} \geq \frac{1}{2} - \frac{\sqrt{1-(1-\sqrt{1-\mu_2})^2}}{2}$$

Assuming Lemma 114, we can prove Lemma 112.

Proof of Lemma 112. Since by hypothesis $\epsilon_2 = \Pr[\mathsf{win}|y^*] - \frac{1}{2}$ and

$$\delta_2 := \frac{\sum_{\substack{d,m,aux\\ D\ni\tilde{x}_0,\tilde{x}_1}} |\alpha_{y^*,d,m,aux,D}|^2}{\||y^*\rangle\langle y^*||\Psi_{final}\rangle\|^2} ,$$

we can apply Lemma 111 to deduce that

(a)

$$\sum_{\substack{d,m,aux,\\ |D\cap\{\tilde{x}_0,\tilde{x}_1\}|=1}} \left|\alpha_{y^*,d,m,aux,D}\right|^2 \geq 2(\varepsilon_1 - \sqrt{\delta_2}) \cdot \||y^*\rangle\langle y^*||\Psi_{final}\rangle\|^2 , \tag{60}$$

(b)

$$\sum_{\substack{d,m,aux,\\ D\not\ni\tilde{x}_0,\tilde{x}_1}} \left|\alpha_{y^*,d,m,aux,D\cup\{\tilde{x}_0\}} - \alpha_{y,d,m,aux,D\cup\{\tilde{x}_1\}}\right|^2 \leq \sum_{\substack{d,m,aux,\\ |D\cap\{\tilde{x}_0,\tilde{x}_1\}|=1}} \left|\alpha_{y^*,d,m,aux,D}\right|^2 - 2(\varepsilon_1 - \sqrt{\delta_2}) .$$

For $b \in \{0,1\}$, let $|\phi_b\rangle$ be the un-normalized state

$$\begin{aligned} |\phi_b\rangle :&= \sum_{\substack{d,m,aux\\ D\ni\tilde{x}_b \wedge D\not\ni\tilde{x}_{\bar{b}}}} \alpha_{y^*,d,m,aux,D} |y^*,d,m,aux\rangle |D\rangle \\ &= \sum_{\substack{d,m,aux\\ D\not\ni\tilde{x}_0,\tilde{x}_1}} \alpha_{y^*,d,m,aux,D\cup\{x_b\}} |y^*,d,m,aux\rangle |D\cup\tilde{x}_b\rangle \ . \end{aligned}$$

Then, we can write (a) equivalently as

$$\||\phi_0\rangle\|^2 + \||\phi_1\rangle\|^2 \geq 2(\varepsilon_1 - \sqrt{\delta_2}) \cdot \||y^*\rangle\langle y^*||\Psi_{final}\rangle\|^2 , \tag{61}$$

We can apply Lemma 114 with $\mu_2 = 2(\varepsilon_1 - \sqrt{\delta_2})$ to deduce that there exists $\mathcal{H}_{good} \subseteq \mathcal{F}(\{0,1\}^n \setminus \{\tilde{x}_0,\tilde{x}_1\},\{0,1\})$ such that

(i)

$$\sum_{\tilde{H} \in \mathcal{H}_{good}} \frac{\|\Pi_{\tilde{H}} |\phi_0\rangle\|^2 + \|\Pi_{\tilde{H}} |\phi_1\rangle\|^2}{\| |\phi_0\rangle\|^2 + \| |\phi_1\rangle\|^2} \geq 1 - \sqrt{1 - 2(\varepsilon_1 - \sqrt{\delta_2})} =: 1 - \xi_1(\varepsilon_1, \delta_2) ,$$

(ii) for all $\tilde{H} \in \mathcal{H}_{good}$, $b \in \{0,1\}$,

$$\frac{\left\|\Pi_{\tilde{H}} |\phi_b\rangle\right\|^2}{\|\Pi_{\tilde{H}} |\phi_0\rangle\|^2 + \|\Pi_{\tilde{H}} |\phi_1\rangle\|^2} \geq \frac{1}{2} - \frac{\sqrt{1 - (1 - \sqrt{1 - 2(\varepsilon_1 - \sqrt{\delta_2})})^2}}{2} =: \frac{1}{2} - \xi_2(\varepsilon_1, \delta_2) .$$

Using Lemma 108, we get that, for all $\tilde{H} \in \mathcal{H}_{good}$, $b \in \{0,1\}$,

$$\begin{aligned}
&\mathbb{E}_{l \leftarrow [q]}[\| |x_b\rangle \langle x_b| (UO)^l |\tilde{H}\rangle \langle \tilde{H}| \mathsf{Decomp} |\Psi_0\rangle \|^2] \\
&\geq \frac{1}{q}\left(\left\|\Pi_{\tilde{H}} |\phi_b\rangle\right\|^2 - \|\Pi_{\tilde{H}} \sum_{\substack{y,d,m,aux \\ D \ni \tilde{x}_b}} \beta_{y,d,m,aux,D} |y,d,m,aux\rangle |D\rangle \|^2 \right) \\
&\geq \frac{\frac{1}{2} - \xi_2(\varepsilon_1, \delta_2)}{q} \cdot \left(\|\Pi_{\tilde{H}} |\phi_0\rangle\|^2 + \|\Pi_{\tilde{H}} |\phi_1\rangle\|^2 \right) \\
&\qquad - \frac{1}{q} \|\Pi_{\tilde{H}} \sum_{\substack{y,d,m,aux \\ D \ni \tilde{x}_0 \text{ or } \tilde{x}_1}} \beta_{y,d,m,aux,D} |y,d,m,aux\rangle |D\rangle \|^2 . \\
&= \frac{1}{2q} \cdot \Bigg(\left(1 - 2\xi_2(\varepsilon_1, \delta_2)\right) \cdot \left(\|\Pi_{\tilde{H}} |\phi_0\rangle\|^2 + \|\Pi_{\tilde{H}} |\phi_1\rangle\|^2 \right) \\
&\qquad\qquad - 2\|\Pi_{\tilde{H}} \sum_{\substack{y,d,m,aux \\ D \ni \tilde{x}_0 \text{ or } \tilde{x}_1}} \beta_{y,d,m,aux,D} |y,d,m,aux\rangle |D\rangle \|^2 \Bigg) \\
&:= \frac{1}{2q} \cdot \Delta_{\tilde{H}} .
\end{aligned} \tag{62}$$

where the second inequality uses (ii) as well as the fact that, for any $\tilde{H} \in \mathcal{F}(\{0,1\}^n \setminus \{\tilde{x}_0, \tilde{x}_1\}, \{0,1\})$, we have

$$\|\Pi_{\tilde{H}} \sum_{\substack{y,d,m,aux \\ D \ni \tilde{x}_b}} \beta_{y,d,m,aux,D} |y,d,m,aux\rangle |D\rangle \|^2 \leq \|\Pi_{\tilde{H}} \sum_{\substack{y,d,m,aux \\ D \ni \tilde{x}_0 \text{ or } \tilde{x}_1}} \beta_{y,d,m,aux,D} |y,d,m,aux\rangle |D\rangle \|^2 .$$

Now, notice that

$$\begin{aligned}
\sum_{\tilde{H} \in \mathcal{H}_{good}} \Delta_{\tilde{H}} &\geq \left(1 - \xi_1(\varepsilon_1, \delta_2) - 2\xi_2(\varepsilon_1, \delta_2) + \xi_1(\varepsilon_1, \delta_2) \cdot \xi_2(\varepsilon_1, \delta_2) - 2\delta_1\right) \cdot \left(\| |\phi_0\rangle\|^2 + \| |\phi_1\rangle\|^2 \right) \\
&\geq \left(1 - \xi_1(\varepsilon_1, \delta_2) - 2\xi_2(\varepsilon_1, \delta_2) - 2\delta_1\right) \cdot \left(\| |\phi_0\rangle\|^2 + \| |\phi_1\rangle\|^2 \right) .
\end{aligned} \tag{63}$$

where the first inequality uses (i) and the definition of δ_1.

We can rewrite Equation (63) as

$$\sum_{\tilde{H} \in \mathcal{H}_{good}} \frac{\|\Pi_{\tilde{H}} |\phi_0\rangle\|^2 + \|\Pi_{\tilde{H}} |\phi_1\rangle\|^2}{\| |\phi_0\rangle\|^2 + \| |\phi_1\rangle\|^2} \cdot \Delta'_{\tilde{H}} \geq 1 - \xi_1(\varepsilon_1, \delta_2) - 2\xi_2(\varepsilon_1, \delta_2) - 2\delta_1 . \tag{64}$$

where

$$\Delta'_{\tilde{H}} := 1 - 2\xi_2(\varepsilon_1, \delta_2) - 2 \cdot \frac{\left\|\Pi_{\tilde{H}} \sum_{\substack{y,d,m,aux \\ D \ni \tilde{x}_0 \text{ or } \tilde{x}_1}} \beta_{y,d,m,aux,D} |y,d,m,aux\rangle |D\rangle \right\|^2}{\|\Pi_{\tilde{H}} |\phi_0\rangle\|^2 + \|\Pi_{\tilde{H}} |\phi_1\rangle\|^2} .$$

An averaging argument applied to Equation (64) implies that there exists a set $\mathcal{H}'_{good} \subseteq \mathcal{H}_{good}$ such that:

(i')

$$\sum_{\tilde{H} \in \mathcal{H}'_{good}} \frac{\|\Pi_{\tilde{H}} |\phi_0\rangle\|^2 + \|\Pi_{\tilde{H}} |\phi_1\rangle\|^2}{\| |\phi_0\rangle\|^2 + \| |\phi_1\rangle\|^2} \geq 1 - \sqrt{\xi_1(\varepsilon_1, \delta_2) + 2\xi_2(\varepsilon_1, \delta_2) + 2\delta_1} ,$$

(ii’) for all $\tilde{H} \in \mathcal{H}'_{good}$,

$$\Delta'_{\tilde{H}} \geq 1 - \sqrt{\xi_1(\varepsilon_1, \delta_2) + 2\xi_2(\varepsilon_1, \delta_2) + 2\delta_1} \tag{65}$$

Since $\mathcal{H}'_{good} \subseteq \mathcal{H}_{good}$, we can plug the latter bound on $\Delta'_{\tilde{H}}$ into Equation (62) to obtain that, for all $\tilde{H} \in \mathcal{H}'_{good}$,

$$\begin{aligned} &\mathbb{E}_{l \leftarrow [q]}[\| \left|x_b\right\rangle \left\langle x_b\right| (UO)^l \left|\tilde{H}\right\rangle \left\langle\tilde{H}\right| \mathsf{Decomp} \left|\Psi_0\right\rangle \|^2] \\ &\geq \frac{1}{2q} \cdot \big(\|\Pi_{\tilde{H}} \left|\phi_0\right\rangle\|^2 + \|\Pi_{\tilde{H}} \left|\phi_1\right\rangle\|^2 \big) \cdot \big(1 - \sqrt{\xi_1(\varepsilon_1, \delta_2) + 2\xi_2(\varepsilon_1, \delta_2) + 2\delta_1}\big) \end{aligned} \tag{66}$$

Using Equation (61), we can rewrite (i’) as

$$\begin{aligned} \sum_{\tilde{H} \in \mathcal{H}'_{good}} \frac{\|\Pi_{\tilde{H}} \left|\phi_0\right\rangle\|^2 + \|\Pi_{\tilde{H}} \left|\phi_1\right\rangle\|^2}{\| \left|y^*\right\rangle \left\langle y^*\right| \left|\Psi_{final}\right\rangle \|^2} &\geq (1 - \sqrt{\xi_1(\varepsilon_1, \delta_2) + 2\xi_2(\varepsilon_1, \delta_2) + 2\delta_1}) \cdot 2(\varepsilon_1 - \sqrt{\delta_2}) \\ &= (1 - \sqrt{\xi_1(\varepsilon_1, \delta_2) + 2\xi_2(\varepsilon_1, \delta_2) + 2\delta_1}) \cdot (1 - (1 - 2(\varepsilon_1 - \sqrt{\delta_2}))) \\ &= (1 - \sqrt{\xi_1(\varepsilon_1, \delta_2) + 2\xi_2(\varepsilon_1, \delta_2) + 2\delta_1}) \cdot (1 - \xi_1^2(\varepsilon_1, \delta_2)) \\ &\geq 1 - \sqrt{\xi_1(\varepsilon_1, \delta_2) + 2\xi_2(\varepsilon_1, \delta_2) + 2\delta_1} - \xi_1^2(\varepsilon_1, \delta_2) \,. \end{aligned} \tag{67}$$

We can further rewrite Equation (67) as

$$\begin{aligned} &\sum_{\tilde{H} \in \mathcal{H}'_{good}} \frac{\|\Pi_{\tilde{H}} \left|y^*\right\rangle \left\langle y^*\right| \left|\Psi_{final}\right\rangle \|^2}{\| \left|y^*\right\rangle \left\langle y^*\right| \left|\Psi_{final}\right\rangle \|^2} \cdot \frac{\|\Pi_{\tilde{H}} \left|\phi_0\right\rangle\|^2 + \|\Pi_{\tilde{H}} \left|\phi_1\right\rangle\|^2}{\|\Pi_{\tilde{H}} \left|y^*\right\rangle \left\langle y^*\right| \left|\Psi_{final}\right\rangle \|^2} \\ &\geq 1 - \sqrt{\xi_1(\varepsilon_1, \delta_2) + 2\xi_2(\varepsilon_1, \delta_2) + 2\delta_1} - \xi_1^2(\varepsilon_1, \delta_2) \,. \end{aligned} \tag{68}$$

By an averaging argument, there exists a set $\mathcal{H}''_{good} \subseteq \mathcal{H}'_{good}$ such that

(i”)

$$\sum_{\tilde{H} \in \mathcal{H}''_{good}} \frac{\|\Pi_{\tilde{H}} \left|y^*\right\rangle \left\langle y^*\right| \left|\Psi_{final}\right\rangle \|^2}{\| \left|y^*\right\rangle \left\langle y^*\right| \left|\Psi_{final}\right\rangle \|^2} \geq 1 - \sqrt{\sqrt{\xi_1(\varepsilon_1, \delta_2) + 2\xi_2(\varepsilon_1, \delta_2) + 2\delta_1} - \xi_1^2(\varepsilon_1, \delta_2)} \,,$$

(ii”) for all $\tilde{H} \in \mathcal{H}''_{good}$,

$$\frac{\|\Pi_{\tilde{H}} \left|\phi_0\right\rangle\|^2 + \|\Pi_{\tilde{H}} \left|\phi_1\right\rangle\|^2}{\|\Pi_{\tilde{H}} \left|y^*\right\rangle \left\langle y^*\right| \left|\Psi_{final}\right\rangle \|^2} \geq 1 - \sqrt{\sqrt{\xi_1(\varepsilon_1, \delta_2) + 2\xi_2(\varepsilon_1, \delta_2) + 2\delta_1} - \xi_1^2(\varepsilon_1, \delta_2)} \,. \tag{69}$$

Plugging (69) into (66), we obtain

(ii”’) For all $\tilde{H} \in \mathcal{H}''_{good}$,

$$\begin{aligned} &\frac{\mathbb{E}_{l \leftarrow [q]}[\| \left|x_b\right\rangle \left\langle x_b\right| (UO)^l \left|\tilde{H}\right\rangle \left\langle\tilde{H}\right| \mathsf{Decomp} \left|\Psi_0\right\rangle \|^2]}{\|\Pi_{\tilde{H}} \left|y^*\right\rangle \left\langle y^*\right| \left|\Psi_{final}\right\rangle \|^2} \\ &\geq \frac{1}{2q} \cdot \big(1 - \sqrt{\sqrt{\xi_1(\varepsilon_1, \delta_2) + 2\xi_2(\varepsilon_1, \delta_2) + 2\delta_1} - \xi_1^2(\varepsilon_1, \delta_2)}\big) \cdot \big(1 - \sqrt{\xi_1(\varepsilon_1, \delta_2) + 2\xi_2(\varepsilon_1, \delta_2) + 2\delta_1}\big) \qquad (70) \\ &\geq \frac{1}{2q} \cdot (1 - \Xi(\varepsilon_1, \delta_2, \delta_1)) \,, \qquad (71) \end{aligned}$$

where

$$\Xi(\varepsilon_1, \delta_2, \delta_1) := \sqrt{\sqrt{\xi_1(\varepsilon_1, \delta_2) + 2\xi_2(\varepsilon_1, \delta_2) + 2\delta_1} - \xi_1^2(\varepsilon_1, \delta_2)} + \sqrt{\xi_1(\varepsilon_1, \delta_2) + 2\xi_2(\varepsilon_1, \delta_2) + 2\delta_1} \,. \tag{72}$$

Finally, using the facts that:

- Decomp acts only on the oracle register and $\mathsf{Decomp} \circ O^{\mathsf{comp}} = O \circ \mathsf{Decomp}$,

- $|\tilde{H}\rangle\langle\tilde{H}|$ commutes with the local unitary evolution and any local measurement,

we have that, for any $\tilde{H}, y^*$,

$$\|\Pi_{\tilde{H}} |y^*\rangle\langle y^*| |\Psi_{final}\rangle\|^2 = \| |\tilde{H}\rangle\langle\tilde{H}| \mathsf{Decomp} |y^*\rangle\langle y^*| (UO^{\mathsf{comp}})^q |\Psi_0\rangle\|^2 \tag{73}$$

$$= \| |y^*\rangle\langle y^*| (UO)^q |\tilde{H}\rangle\langle\tilde{H}| \mathsf{Decomp} |\Psi_0\rangle\|^2 . \tag{74}$$

Thus, we can replace the denominator in the LHS of (71) with $\| |y^*\rangle\langle y^*| (UO)^q |\tilde{H}\rangle\langle\tilde{H}| \mathsf{Decomp} |\Psi_0\rangle\|^2$.

Then, $\mathcal{H}''_{good}$ is the desired set and (i") and (ii"') are the desired conditions. It is also straightforward to check that Ξ as defined above satisfies the desired properties. □

We are left with proving Lemma 114.

Proof of Lemma 114. For $b \in \{0,1\}$, let $|\phi_b\rangle$ be the un-normalized state

$$\begin{aligned} |\phi_b\rangle : &= \sum_{\substack{d,m,aux \\ D \ni x_b \wedge D \not\ni x_{\bar{b}}}} \alpha_{y^*,d,m,aux,D} |y^*, d, m, aux\rangle |D\rangle \\ &= \sum_{\substack{d,m,aux \\ D \not\ni \tilde{x}_0, \tilde{x}_1}} \alpha_{y^*,d,m,aux,D\cup\{x_b\}} |y^*, d, m, aux\rangle |D \cup x_b\rangle \ . \end{aligned}$$

First notice that, for any $\tilde{H} \in \mathcal{F}(\{0,1\}^n \setminus \{\tilde{x}_0, \tilde{x}_1\}, \{0,1\})$,

$$\begin{aligned} \|\Pi_{\tilde{H}} |\phi_1\rangle\|^2 &= \sum_{d,m,aux} \|\Pi_{\tilde{H}} \sum_{D \not\ni \tilde{x}_0, \tilde{x}_1} \alpha_{y^*,d,aux,D\cup\{\tilde{x}_1\}} |y^*, d, m, aux\rangle |D \cup \{\tilde{x}_1\}\rangle\|^2 \\ &= \sum_{d,aux} \|\Pi_{\tilde{H}} \sum_{D \not\ni \tilde{x}_0, \tilde{x}_1} (-1)^{m+d\cdot(x_0\oplus x_1)} \alpha_{y^*,d,aux,D\cup\{\tilde{x}_0\}} |y^*, d, m, aux\rangle |D \cup \{\tilde{x}_0\}\rangle\|^2 \\ &= \|\Pi_{\tilde{H}} |\phi_1'\rangle\|^2 \end{aligned} \tag{75}$$

where $|\phi_1'\rangle \coloneqq \sum_{\substack{d,m,aux \\ D \not\ni \tilde{x}_0, \tilde{x}_1}} (-1)^{m+d\cdot(x_0\oplus x_1)} \alpha_{y^*,d,m,aux,D\cup\{\tilde{x}_1\}} |y^*, d, m, aux\rangle |D \cup \tilde{x}_0\rangle$. The second equality in Equation (75) holds because the unitary $|D \cup \{\tilde{x}_1\}\rangle \mapsto |D \cup \{\tilde{x}_0\}\rangle$ commutes with $\Pi_{\tilde{H}}$ and thus does not affect the norm, and the phase $(-1)^{m+d\cdot(x_0\oplus x_1)}$ clearly also does not affect the norm. Hence, we have

$$\|\Pi_{\tilde{H}} |\phi_1\rangle\| = \|\Pi_{\tilde{H}} |\phi_1'\rangle\| . \tag{76}$$

In the following calculation, the sum is over $\tilde{H} \in \mathcal{F}(\{0,1\}^n \setminus \{\tilde{x}_0, \tilde{x}_1\}, \{0,1\})$. Notice that

$$\begin{aligned} &\sum_{\tilde{H}} (\|\Pi_{\tilde{H}} |\phi_0\rangle\| - \|\Pi_{\tilde{H}} |\phi_1\rangle\|)^2 \\ &= \sum_{\tilde{H}} (\|\Pi_{\tilde{H}} |\phi_0\rangle\| - \|\Pi_{\tilde{H}} |\phi_1'\rangle\|)^2 \qquad \text{by Equation (76)} \\ &\le \sum_{\tilde{H}} \|\Pi_{\tilde{H}} (|\phi_0\rangle - |\phi_1'\rangle)\|^2 \qquad \text{by the triangle inequality} \\ &= \| |\phi_0\rangle - |\phi_1'\rangle\|^2 \\ &= \sum_{\substack{d,m,aux, \\ D \not\ni \tilde{x}_0, \tilde{x}_1}} |\alpha_{y^*,d,m,aux,D\cup\{\tilde{x}_0\}} - (-1)^{m+d\cdot(x_0\oplus x_1)} \alpha_{y^*,d,m,aux,D\cup\{\tilde{x}_1\}}|^2 \\ &\le \sum_{\substack{d,m,aux, \\ D \not\ni \tilde{x}_0, \tilde{x}_1, b \in \{0,1\}}} |\alpha_{y^*,d,m,aux,D\cup\{x_b\}}|^2 \cdot (1 - \mu_2) \qquad \text{by Equation (59)} \\ &= (\| |\phi_0\rangle\|^2 + \| |\phi_1\rangle\|^2) \cdot (1 - \mu_2) \end{aligned} \tag{77}$$

We can equivalently rewrite Equation (77) as

$$\sum_{\tilde{H}} p_{\tilde{H}} \cdot \delta_{\tilde{H}} \le (1 - \mu_2) ,$$

where

$$p_{\tilde{H}} \coloneqq \frac{\|\Pi_{\tilde{H}} |\phi_0\rangle\|^2 + \|\Pi_{\tilde{H}} |\phi_1\rangle\|^2}{\| |\phi_0\rangle\|^2 + \| |\phi_1\rangle\|^2} ,$$

and

$$\delta_{\tilde{H}} := \frac{(\| \Pi_{\tilde{H}} |\phi_0\rangle \| - \| \Pi_{\tilde{H}} |\phi_1\rangle \|)^2}{\| \Pi_{\tilde{H}} |\phi_0\rangle \|^2 + \| \Pi_{\tilde{H}} |\phi_1\rangle \|^2} \,.$$

Note that $\sum_{\tilde{H}} p_{\tilde{H}} = 1$. Then, by an averaging argument, there must exist $\mathcal{H}_{good} \subseteq \mathcal{F}(\{0,1\}^n \setminus \{\tilde{x}_0, \tilde{x}_1\}$ such that

(a)

$$\sum_{\tilde{H} \in \mathcal{H}_{good}} p_{\tilde{H}} \geq 1 - \sqrt{1-\mu_2}\,, \tag{78}$$

(b) for all $\tilde{H} \in \mathcal{H}_{good}$,

$$\delta_{\tilde{H}} \leq \sqrt{1-\mu_2}\,. \tag{79}$$

We use the following lemma omitting the proof.

Lemma 115. *Let $0 \leq \gamma \leq 1$, and v, w vectors in a Hilbert space. If $\frac{(\|v\| - \|w\|)^2}{\|v\|^2 + \|w\|^2} \leq \gamma$, then*

$$\frac{\min(\|v\|^2, \|w\|^2)}{\|v\|^2 + \|w\|^2} \geq \frac{1}{2} - \frac{\sqrt{1-(1-\gamma)^2}}{2} \,.$$

Using Lemma 115 we have that (b) implies

(b') for all $H \in \mathcal{H}_{good}$, for $b \in \{0,1\}$,

$$\frac{\| \Pi_{\tilde{H}} |\phi_b\rangle \|^2}{\| \Pi_{\tilde{H}} |\phi_0\rangle \|^2 + \| \Pi_{\tilde{H}} |\phi_1\rangle \|^2} \geq \frac{1}{2} - \frac{\sqrt{1-(1-\sqrt{1-\mu_2})^2}}{2} \,. \tag{80}$$

(a) and (b') are the desired conditions. □

We now state Lemma 112 in a form which will be useful in our proof later on. Let Ξ be the same function as in Lemma 112.

Corollary 116. *Suppose the hypothesis of Lemma 112 holds. Then, there exists $\mathcal{H}_{good} \subseteq \mathcal{F}(\{0,1\}^n \setminus \{\tilde{x}_0, \tilde{x}_1\}, \{0,1\})$ such that*

(i)

$$\sum_{\tilde{H} \in \mathcal{H}_{good}} \| \Pi_{\tilde{H}} |\Psi_0\rangle \|^2 \geq 1 - \sqrt{\Xi(\varepsilon_1, \delta_1, \delta_2)} \cdot \| |y^*\rangle \langle y^*| |\Psi_{final}\rangle \|^2 \,.$$

(ii) for all $\tilde{H} \in \mathcal{H}_{good}$, $b \in \{0,1\}$,

$$\frac{\mathbb{E}_{l \leftarrow [q]}[\| |x_b\rangle \langle x_b| (UO)^l |\tilde{H}\rangle \langle \tilde{H}| \mathsf{Decomp} |\Psi_0\rangle \|^2]}{\| \Pi_{\tilde{H}} |\Psi_0\rangle \|^2} \geq \frac{1}{2q} \Big(1 - \sqrt{\Xi(\varepsilon_1, \delta_1, \delta_2)} - \Xi(\varepsilon_1, \delta_1, \delta_2)\Big) \cdot \| |y^*\rangle \langle y^*| |\Psi_{final}\rangle \|^2 \,. \tag{81}$$

Proof. From Lemma 112, we have that there exists $\mathcal{H}_{good} \subseteq \mathcal{F}(\{0,1\}^n \setminus \{\tilde{x}_0, \tilde{x}_1\}, \{0,1\})$ such that

(i)

$$\sum_{\tilde{H} \in \mathcal{H}_{good}} \frac{\| \Pi_{\tilde{H}} |y^*\rangle \langle y^*| |\Psi_{final}\rangle \|^2}{\| |y^*\rangle \langle y^*| |\Psi_{final}\rangle \|^2} \geq 1 - \Xi(\varepsilon_1, \delta_1, \delta_2) \,.$$

(ii) for all $\tilde{H} \in \mathcal{H}_{good}$, $b \in \{0,1\}$,

$$\frac{\mathbb{E}_{l \leftarrow [q]}[\| |x_b\rangle \langle x_b| (UO)^l |\tilde{H}\rangle \langle \tilde{H}| \mathsf{Decomp} |\Psi_0\rangle \|^2]}{\| |y^*\rangle \langle y^*| (UO)^q |\tilde{H}\rangle \langle \tilde{H}| \mathsf{Decomp} |\Psi_0\rangle \|^2} \geq \frac{1}{2q} \cdot \big(1 - \Xi(\varepsilon_1, \delta_1, \delta_2)\big)$$

We can rewrite (i) as

$$\sum_{\tilde{H}\in\mathcal{H}_{good}} p^{tot}_{\tilde{H}} \cdot \frac{\|\Pi_{\tilde{H}}\,|y^*\rangle\langle y^*|\,|\Psi_{final}\rangle\,\|^2}{p^{tot}_{\tilde{H}}} \geq 1 - \Xi(\varepsilon_1,\delta_1,\delta_2)\cdot \|\,|y^*\rangle\langle y^*|\,|\Psi_{final}\rangle\,\|^2\,,$$

where $p^{tot}_{\tilde{H}} := \|\Pi_{\tilde{H}}\,|\Psi_0\rangle\,\|^2$. Then, by an averaging argument, there exists $\mathcal{H}'_{good} \subseteq \mathcal{H}_{good}$ such that

(a)

$$\sum_{\tilde{H}\in\mathcal{H}'_{good}} p^{tot}_{\tilde{H}} \geq 1 - \sqrt{\Xi(\varepsilon_1,\delta_1,\delta_2)}\,, \tag{82}$$

(b) for all $H \in \mathcal{H}'_{good}$,

$$\frac{\|\Pi_{\tilde{H}}\,|y^*\rangle\langle y^*|\,|\Psi_{final}\rangle\,\|^2}{p^{tot}_{\tilde{H}}} \geq \frac{1}{2q}\Big(1-\sqrt{\Xi(\varepsilon_1,\delta_1,\delta_2)}\Big)\cdot\Big(1-\Xi(\varepsilon_1,\delta_1,\delta_2)\Big)\cdot \|\,|y^*\rangle\langle y^*|\,|\Psi_{final}\rangle\,\|^2 \tag{83}$$

Now, notice that

$$\|\,|y^*\rangle\langle y^*|\,(UO)^q\,|\tilde{H}\rangle\langle\tilde{H}|\,\mathsf{Decomp}\,|\Psi_0\rangle\,\|^2 = \|\Pi_{\tilde{H}}\,|y^*\rangle\langle y^*|\,|\Psi_{final}\rangle\,\|^2\,.$$

Then, since $\mathcal{H}'_{good} \subseteq \mathcal{H}_{good}$, (ii) and (b) together imply

(b') for all $H \in \mathcal{H}'_{good}$, $b \in \{0,1\}$,

$$\frac{\mathbb{E}_{l\leftarrow[q]}[\|\,|x_b\rangle\langle x_b|\,(UO)^l\,|\tilde{H}\rangle\langle\tilde{H}|\,\mathsf{Decomp}\,|\Psi_0\rangle\,\|^2]}{p^{tot}_{\tilde{H}}} \geq \frac{1}{2q}\Big(1-\sqrt{\Xi(\varepsilon_1,\delta_1,\delta_2)}-\Xi(\varepsilon_1,\delta_1,\delta_2)\Big)\cdot \|\,|y^*\rangle\langle y^*|\,|\Psi_{final}\rangle\,\|^2 \tag{84}$$

$\mathcal{H}'_{good}$ is the desired set, and (a) and (b') are the desired conditions. □

We state a simple consequence of Corollary 116, which we will use directly in our proof later on. Let $\tilde{x}_0 \neq \tilde{x}_1$ be in the domain of H, and let $c_0, c_1 \in \{0,1\}$. Let $O^H_{(\tilde{x}_0,c_0),(\tilde{x}_1,c_1)}$ be defined as:

$$O^H_{(\tilde{x}_0,c_0),(\tilde{x}_1,c_1)}\,|x\rangle\,|z\rangle = \begin{cases} O^H\,|x\rangle\,|z\rangle\,, & \text{if } x \neq \tilde{x}_0,\tilde{x}_1 \\ (-1)^{z\cdot c_0}\,|x\rangle\,|z\rangle\,, & \text{if } x = \tilde{x}_0 \\ (-1)^{z\cdot c_1}\,|x\rangle\,|z\rangle\,, & \text{if } x = \tilde{x}_1 \end{cases}$$

Corollary 117. *Suppose the conditions of Lemma 112 hold. Then, there exists* $\mathcal{H}_{good} \subseteq \mathcal{F}(\{0,1\}^n \setminus \{\tilde{x}_0,\tilde{x}_1\},\{0,1\})$ *such that*

(i) $\sum_{\tilde{H}\in\mathcal{H}_{good}} \|\Pi_{\tilde{H}}\,|\Psi_0\rangle\,\|^2 \geq \big(1-\sqrt{\Xi(\varepsilon_1,\delta_1,\delta_2)}\big)\cdot \|\,|y^*\rangle\langle y^*|\,|\Psi_{final}\rangle\,\|^2$,

(ii) for all $\tilde{H} \in \mathcal{H}_{good}$, $b \in \{0,1\}$,

$$\frac{\mathbb{E}_{c_0,c_1\leftarrow\{0,1\}}\mathbb{E}_{l\leftarrow[q]}[\|\,|\tilde{x}_b\rangle\langle\tilde{x}_b|\,(UO_{(\tilde{x}_0,c_0),(\tilde{x}_1,c_1)})^l\,|\tilde{H}\rangle\langle\tilde{H}|\,\mathsf{Decomp}\,|\Psi_0\rangle\,\|^2]}{\|\Pi_{\tilde{H}}\,|\Psi_0\rangle\,\|^2} \tag{85}$$

$$\geq \frac{1-\sqrt{\Xi(\varepsilon_1,\delta_1,\delta_2)}-\Xi(\varepsilon_1,\delta_1,\delta_2)}{8q}\cdot \|\,|y^*\rangle\langle y^*|\,|\Psi_{final}\rangle\,\|^2\,. \tag{86}$$

Proof. This is immediate since one of the four possible assignments of oracle outputs at $\tilde{x}_0,\tilde{x}_1$ agrees with $H(\tilde{x}_0), H(\tilde{x}_1)$. □

11.4.3 The structure of successful $\mathsf{CQ_d}$ strategies

By Lemma 104, recall that it suffices to restrict our analysis to $\mathcal{W}_d$ strategies that are successful at $\mathsf{subProblem}$, in the sense of Lemma 105. Let A be a $\mathcal{W}_d$ strategy for $\mathsf{subProblem}$. Since H_d is only queried by classical circuits, we assume that all queries to H_d are recorded by measuring the query register, without any disturbance to the state of the algorithm. We denote by Q the total number of queries to H_d made by the algorithm. We moreover assume that all points at which H_d is queried by the algorithm are distinct.

For a security parameter $\lambda \in \mathbb{N}$, we denote by $\mathcal{G}_\lambda$ the set of all possible functions G_b, by $\mathcal{Y}_\lambda$ the co-domain of such functions, and by $\mathcal{H}'_\lambda$ the set of all possible functions h. We denote by $\mathcal{V}_\lambda$ the set of all possible outcomes adv that one can obtain by measuring the registers work, query of A. We omit writing λ when this is clear from the context.

Throughout the section, $poly$ denotes a polynomial, not always the same one. Let Z^i be a random variable for the i-th (classical) query to H_d. In the following theorem, we denote by $\mathsf{h}^i_{\mathsf{data}}$ the random variable representing the set $h^{-1}(H_d(Z^i))$. Moreover, for simplicity and ease of notation, we assume that $\mathsf{h}^i_{\mathsf{data}}$ consists of a single element, and we identify the set with that element. The argument is virtually unchanged without this assumption since h is injective with overwhelming probability.

Lemma 118. *Let $\varepsilon : \mathbb{N} \to \mathbb{R}$. Suppose, for all λ, $\Pr[y \in \mathsf{TwoToOne}(G_0, G_1) : y, d, m \leftarrow A_\lambda] > \varepsilon$, and $\Pr[A \text{ wins}|y \in \mathsf{TwoToOne}(G_0, G_1), y, d, m \leftarrow A_\lambda] > \frac{1}{2} + \epsilon(\lambda)$. Consider a simulation $\tilde{A}$ of A, where calls to H are simulated via a compressed oracle.*

For $i \in [Q]$, let $\mathsf{h}^i_{\mathsf{data}}$ be the random variable for the i-th classical query to h made by $\tilde{A}$. Let D^i be a random variable obtained by measuring the compressed database (in the standard basis) just before the i-th query to h, and Y_{out} be a random variable for the y output of $\tilde{A}$. The random variables are implicitly functions of λ, but we omit writing this.

Then, there exists a negligible function negl such that, for all λ, there exists $i^* \in [Q]$, and $\mathcal{W} \subseteq \mathcal{V}_\lambda \times \mathcal{Y}_\lambda \times \mathcal{G}_\lambda \times \mathcal{G}_\lambda \times \mathcal{H}'_\lambda$ such that:

$$\Pr[(adv, \mathsf{h}^{i^*}_{\mathsf{data}}, G_0, G_1, h) \in \mathcal{W}] \geq poly(\varepsilon)\,.$$

Moreover, for all $(\tilde{adv}, \tilde{y}, \tilde{G}_0, \tilde{G}_1, \tilde{h}) \in \mathcal{W}$:

- $\tilde{y} \in \mathsf{TwoToOne}(\tilde{G}_0, \tilde{G}_1)$,
- $\Pr[(x, \tilde{h}(y)) \in D^{i^*} \text{ for some } x|(adv, G_0, G_1, h) = (\tilde{adv}, \tilde{G}_0, \tilde{G}_1, \tilde{h}) \wedge \mathsf{h}^{i^*}_{\mathsf{data}} = \tilde{y}] = \mathsf{negl}(\lambda)$,
- $\Pr[Y_{out} = \tilde{y}|(adv, G_0, G_1, h) = (\tilde{adv}, \tilde{G}_0, \tilde{G}_1, \tilde{h}) \wedge \mathsf{h}^{i^*}_{\mathsf{data}} = \tilde{y}] \geq poly(\varepsilon)$.
- $\Pr[\tilde{A} \text{ wins}|(adv, G_0, G_1, h) = (\tilde{adv}, \tilde{G}_0, \tilde{G}_1, \tilde{h}) \wedge Y_{out} = \mathsf{h}^{i^*}_{\mathsf{data}} = \tilde{y}] \geq \frac{1}{2} + poly(\varepsilon(\lambda))$.

We will prove the following Lemma first.

Lemma 119. *Suppose the hypothesis of Lemma 118 holds. Then, either there exists a negligible function negl such that*

$$\Pr[Y_{out} \notin \mathsf{h}_{\mathsf{data}} \wedge Y_{out} \in \mathsf{TwoToOne}(G_0, G_1)] = \mathsf{negl}(\lambda)\,,$$

or there exists a negligible function such that

$$\Pr[\tilde{A} \text{ wins} \,|\, Y_{out} \notin \mathsf{h}_{\mathsf{data}} \,\wedge\, Y_{out} \in \mathsf{TwoToOne}(G_0, G_1)] \leq \frac{1}{2} + \mathsf{negl}(\lambda)\,.$$

Proof. Suppose there exists a non-negligible function $\mathsf{non\text{-}negl}$ such that, for all λ,

$$\Pr[Y_{out} \notin \mathsf{h}_{\mathsf{data}} \,\wedge\, Y_{out} \in \mathsf{TwoToOne}(G_0, G_1)] = \mathsf{non\text{-}negl}(\lambda)\,.$$

Suppose for a contradiction that there exists a non-negligible function $\mathsf{non\text{-}negl}'$ such that, for all λ,

$$\Pr[\tilde{A} \text{ wins}|Y_{out} \notin \mathsf{h}_{\mathsf{data}} \wedge Y \in \mathsf{TwoToOne}(G_0, G_1)] > \frac{1}{2} + \mathsf{non\text{-}negl'}(\lambda)\,.$$

Fix λ. Let $\mathcal{Q}$ be the set of possible values that $\mathsf{h}_{\mathsf{data}}$ can take. Then, by an averaging argument, there exists $\mathcal{S} \subseteq \mathcal{Y} \times \mathcal{Q} \times \mathcal{G}_0 \times \mathcal{G}_1$ such that $\Pr[(Y_{out}, \mathsf{h}_{\mathsf{data}}, G_0, G_1) \in \mathcal{S}] = \mathsf{non\text{-}negl}(\lambda)$, and moreover, for all $(\tilde{y}, \tilde{\mathsf{h}}_{\mathsf{data}}, \tilde{G}_0, \tilde{G}_1) \in \mathcal{S}$,

- $\tilde{y} \notin \tilde{\mathsf{h}}_{\mathsf{data}}$, and $\tilde{y} \in \mathsf{TwoToOne}(\tilde{G}_0, \tilde{G}_1)$.

- $\Pr[\tilde{A} \text{ wins} \,|\, (Y_{out}, \mathsf{h}_{\mathsf{data}}, G_0, G_1) = (\tilde{y}, \tilde{\mathsf{h}}_{\mathsf{data}}, \tilde{G}_0, \tilde{G}_1)] \geq \frac{1}{2} + \mathsf{non\text{-}negl}(\lambda)\,.$

Let D^{final} be a random variable for the the outcome of measuring the final state of the compressed database (in the standard basis). Then, we can apply Lemma 111 to deduce that

$$\Pr[D^{final} \ni (x, h(Y_{out})) \,\wedge\, Y_{out} \notin \mathsf{h}_{\mathsf{data}} \,\wedge\, Y_{out} \in \mathsf{TwoToOne}(G_0, G_1) \,\wedge\, x \in \{G_0^{-1}(Y_{out}), G_1^{-1}(Y_{out})\}] \\ = \mathsf{non\text{-}negl}''(\lambda)\,,$$

for some non-negligible function $\mathsf{non\text{-}negl}''$. This straightforwardly implies that there exist an algorithm that only makes classical queries to h, and correctly predicts the value of h at an unqueried point with non-negligible probability. This is a contradiction. □

Proof of Lemma 118. Fix λ. For the rest of the proof, we omit writing any λ dependence. By hypothesis, $\Pr[Y_{out} \in \mathsf{TwoToOne}(G_0, G_1)] > poly(\varepsilon)$, and $\Pr[A \text{ wins} | Y_{out} \in \mathsf{TwoToOne}(G_0, G_1)] > \frac{1}{2} + \varepsilon$. Then, using Lemma 119, it is straightforward to see that there exists a negligible function negl such that:

(i) $\Pr[Y_{out} \in \mathsf{h}_{\mathsf{data}} | Y_{out} \in \mathsf{TwoToOne}(G_0, G_1)] > \min(poly(\varepsilon), 1 - \mathsf{negl})$, and

(ii) $\Pr[\tilde{A} \text{ wins} | Y_{out} \in \mathsf{h}_{\mathsf{data}} \wedge Y_{out} \in \mathsf{TwoToOne}(G_0, G_1)] \geq \frac{1}{2} + poly(\varepsilon)$

In what follows, for ease of notation, we denote the event "$Y_{out} \in \mathsf{TwoToOne}(G_0, G_1)$" as E. We can equivalently rewrite (ii) as

$$\sum_{i \in [Q]} \frac{\Pr[Y_{out} = \mathsf{h}^i_{\mathsf{data}} \,|\, E]}{\Pr[Y_{out} \in \mathsf{h}_{\mathsf{data}} \,|\, E]} \cdot \frac{\Pr[\tilde{A} \text{ wins} \wedge Y_{out} = \mathsf{h}^i_{\mathsf{data}} \,|\, E]}{\Pr[Y_{out} = \mathsf{h}^i_{\mathsf{data}} \,|\, E]} \geq \frac{1}{2} + poly(\varepsilon) \tag{87}$$

By an averaging argument, there exists $i^* \in [Q]$ such that

- $\Pr[Y_{out} = \mathsf{h}^{i^*}_{\mathsf{data}} \,|\, E] \geq \Pr[Y_{out} \in \mathsf{h}_{\mathsf{data}} \,|\, E] \cdot \frac{poly(\varepsilon)}{Q} \geq \frac{poly(\varepsilon)}{Q}$, and
- $\Pr[\tilde{A} \text{ wins} \,|\, Y_{out} = \mathsf{h}^{i^*}_{\mathsf{data}} \,\wedge\, E] > \frac{1}{2} + poly(\varepsilon)$

Since $\Pr[E] \geq \varepsilon$ by hypothesis, the former implies $\Pr[Y_{out} = \mathsf{h}^{i^*}_{\mathsf{data}} \,\wedge\, E] \geq \frac{poly(\varepsilon)}{Q} \cdot \Pr[E] \geq \frac{poly(\varepsilon)}{Q}$. By another averaging argument, there exists $\mathcal{W} \subseteq \mathcal{V} \times \mathcal{Y} \times \mathcal{G} \times \mathcal{G} \times \mathcal{H}'$ such that:

$$\Pr[Y_{out} = \mathsf{h}^{i^*}_{\mathsf{data}} \,\wedge\, (adv, \mathsf{h}^{i^*}_{\mathsf{data}}, G_0, G_1, h) \in \mathcal{W}] \geq \frac{poly(\varepsilon)}{Q}\,.$$

Moreover, for all $(\tilde{adv}, \tilde{y}, \tilde{G}_0, \tilde{G}_1, \tilde{h}) \in \mathcal{W}$:

- $\tilde{y} \in \mathsf{TwoToOne}(\tilde{G}_0, \tilde{G}_1)\,,$
- $\Pr[\tilde{A} \text{ wins} | Y_{out} = \mathsf{h}_{\mathsf{data}^{i^*}} = \tilde{y} \,\wedge\, (adv, G_0, G_1, h) = (\tilde{adv}, \tilde{G}_0, \tilde{G}_1, \tilde{h})] \geq \frac{1}{2} + poly(\varepsilon)\,.$

Notice that, trivially, for any $\tilde{y}$, the distribution of $h(\tilde{y})$ is uniform, conditioned on the values of H_d at any subset of points that does not contain $H_{d-1} \circ \cdots \circ H_0 \tilde{y}$. Thus, since we assumed without loss of generality that A never queries H_d at the same point twice, this clearly implies that, for all $(\tilde{adv}, \tilde{y}, \tilde{G}_0, \tilde{G}_1, \tilde{h})$,

- $\Pr[(x, h(\tilde{y})) \in D^{i^*} \text{ for some } x | (G_0, G_1) = (\tilde{G}_0, \tilde{G}_1) \,\wedge\, \mathsf{h}^{i^*}_{\mathsf{data}} = \tilde{y}] = \mathsf{negl}(\lambda)\,,$

This implies that there exists $\mathcal{W}' \subseteq \mathcal{W}$ such that:

$$\Pr[Y_{out} = \mathsf{h}^{i^*}_{\mathsf{data}} \,\wedge\, (adv, \mathsf{h}^{i^*}_{\mathsf{data}}, G_0, G_1, h) \in \mathcal{W}] \geq \frac{poly(\varepsilon)}{Q}\,.$$

Moreover, for all $(\tilde{y}, \tilde{G}_0, \tilde{G}_1, \tilde{h}) \in \mathcal{W}'$:

- $\tilde{y} \in \mathsf{TwoToOne}(\tilde{G}_0, \tilde{G}_1)\,,$
- $\Pr[\tilde{A} \text{ wins} | Y_{out} = \mathsf{h}_{\mathsf{data}^{i^*}} = \tilde{y} \,\wedge\, (adv, G_0, G_1, h) = (\tilde{adv}, \tilde{G}_0, \tilde{G}_1, \tilde{h})] \geq \frac{1}{2} + poly(\varepsilon)\,,$

and, for all $(\tilde{adv}, \tilde{y}, \tilde{G}_0, \tilde{G}_1, h) \in \mathcal{W}'$,

- $\Pr[(x, \tilde{h}(\tilde{y})) \in D^{i^*} \text{ for some } x | (adv, G_0, G_1, h) = (\tilde{adv}, \tilde{G}_0, \tilde{G}_1, \tilde{h}) \,\wedge\, \mathsf{h}^{i^*}_{\mathsf{data}} = \tilde{y}] = \mathsf{negl}(\lambda)\,,$

By one final averaging argument, there exists $\mathcal{W}'' \subseteq \mathcal{W}'$ such that

$$\Pr[(adv, \mathsf{h}^{i^*}_{\mathsf{data}}, G_0, G_1, h) \in \mathcal{W}''] \geq \frac{poly(\varepsilon)}{Q},$$

and for all $(\widetilde{adv}, \tilde{y}, \tilde{G}_0, \tilde{G}_1, \tilde{h}) \in \mathcal{W}''$,

- $\Pr[Y_{out} = \mathsf{h}^{i^*}_{\mathsf{data}} | (adv, \mathsf{h}^{i^*}_{\mathsf{data}}, G_0, G_1, h) = (\widetilde{adv}, \tilde{y}, \tilde{G}_0, \tilde{G}_1, \tilde{h})] \geq \frac{poly(\varepsilon)}{Q}$,
- $\tilde{y} \in \mathsf{TwoToOne}(\tilde{G}_0, \tilde{G}_1)$,
- $\Pr[\tilde{A} \text{ wins} | Y_{out} = \mathsf{h}^{i^*}_{\mathsf{data}} = \tilde{y} \land (adv, G_0, G_1, h) = (\widetilde{adv}, \tilde{G}_0, \tilde{G}_1, \tilde{h})] \geq \frac{1}{2} + poly(\varepsilon)$.
- $\Pr[(x, h(\tilde{y})) \in D^{i^*} \text{ for some } x | (adv, G_0, G_1, h) = (\widetilde{adv}, \tilde{G}_0, \tilde{G}_1, \tilde{h}) \land \mathsf{h}^{i^*}_{\mathsf{data}} = \tilde{y}] = \mathsf{negl}(\lambda)$.

This concludes the proof of Lemma 118.

□

11.4.4 Putting things together

In this subsection, we complete the proof of Lemma 107. Let $\varepsilon : \mathbb{N} \to [0,1]$. Suppose, for all λ, $\Pr[y \in \mathsf{TwoToOne}(G_0, G_1) : y, d, m \leftarrow A_\lambda] \geq \varepsilon(\lambda)$, and $\Pr[A \text{ wins} | y \in \mathsf{TwoToOne}(G_0, G_1), y, d, m \leftarrow A_\lambda] \geq \frac{1}{2} + \epsilon(\lambda)$. Let q be the total number of queries made by A.

We will show that Algorithm 1 extracts a collision with probability at least $poly(\varepsilon, 1/q)$.

We can apply Lemma 118. Using the notation of Lemma 118, we have that there exists a negligible function negl such that, for all λ, there exists $i^* \in [Q]$, and $\mathcal{W} \subseteq \mathcal{V} \times \mathcal{Y} \times \mathcal{G} \times \mathcal{G} \times \mathcal{H}'$ such that:

$$\Pr[(adv, \mathsf{h}^{i^*}_{\mathsf{data}}, G_0, G_1, h) \in \mathcal{W}] \geq poly(\varepsilon). \tag{88}$$

Moreover, for all $(\widetilde{adv}, \tilde{y}, \tilde{G}_0, \tilde{G}_1, \tilde{h}) \in \mathcal{W}$:

(i) $\tilde{y} \in \mathsf{TwoToOne}(\tilde{G}_0, \tilde{G}_1)$,

(ii) $\Pr[(x, h(y)) \in D^{i^*} \text{ for some } x | (adv, G_0, G_1, h) = (\widetilde{adv}, \tilde{G}_0, \tilde{G}_1, \tilde{h}) \land \mathsf{h}^{i^*}_{\mathsf{data}} = \tilde{y}] = \mathsf{negl}(\lambda)$,

(iii) $\Pr[Y_{out} = \tilde{y} | (adv, G_0, G_1, h) = (\widetilde{adv}, \tilde{G}_0, \tilde{G}_1, \tilde{h}) \land \mathsf{h}_{\mathsf{data}^{i^*}} = \tilde{y}] \geq poly(\varepsilon)$.

(iv) $\Pr[\tilde{A} \text{ wins} | (adv, G_0, G_1, h) = (\widetilde{adv}, \tilde{G}_0, \tilde{G}_1, \tilde{h}) \land Y_{out} = \mathsf{h}_{\mathsf{data}^{i^*}} = \tilde{y}] \geq \frac{1}{2} + poly(\varepsilon)$.

Notice then that, at step (ii) of Algorithm 1,

$$\Pr[i = i^* \land (adv, \mathsf{h}^{i^*}_{\mathsf{data}}, G_0, G_1, h) \in \mathcal{W}] \geq \frac{poly(\varepsilon)}{Q}.$$

Fix $(\widetilde{adv}, \tilde{y}, \tilde{G}_0, \tilde{G}_1, \tilde{h}) \in \mathcal{W}$. Let $|\Psi_0\rangle$ be the state of the compressed oracle simulation after step (ii), conditioned on $i = i^*$ and $\widetilde{adv}, \tilde{y}, \tilde{G}_0, \tilde{G}_1, \tilde{h}$. Let $x_0 = G_0^{-1}(\tilde{y})$, and $x_1 = G_1^{-1}(\tilde{y})$.

Let $|\Psi_{final}\rangle$ be the final state of the compressed oracle simulation (continuing from $|\Psi_0\rangle$), i.e. $|\Psi_{final}\rangle = (U' O^{\mathsf{comp}})^{q} |\Psi_0\rangle$, where $\tilde{q}$ denotes the number of remaining queries to H, and we are absorbing in U' all queries to G_0, G_1, h as well as the unitaries U_C and U_Q.

Condition (ii) implies that

$$\delta_1 := \| \Pi_{D \cap \{(x, h(\tilde{y})) : x \in \mathcal{X}\} \neq \varnothing} |\Psi_0\rangle \|^2 = \mathsf{negl}(\lambda).$$

Condition (iii) implies that

$$\| |\tilde{y}\rangle \langle \tilde{y}| |\Psi_{final}\rangle \|^2 \geq poly(\varepsilon), \tag{89}$$

and condition (iv) implies that

$$\varepsilon_1 := \Pr[\tilde{A} \text{ wins } | (adv, G_0, G_1, h) = (\widetilde{adv}, \tilde{G}_0, \tilde{G}_1, \tilde{h}) \land Y_{out} = \mathsf{h}_{\mathsf{data}^{i^*}} = \tilde{y}] - \frac{1}{2} \geq poly(\varepsilon).$$

Finally notice that there exists a negligible function negl', such that except with negl' probability over $(\widetilde{adv}, \tilde{y}, \tilde{G}_0, \tilde{G}_1, \tilde{h}) \in \mathcal{W}$, it must be that $\delta_2 := \| \Pi_{D \ni (x_0, h(\tilde{y})), (x_1, h(\tilde{y}))} \left| \Psi_{final} \right\rangle \|^2 \leq \mathsf{negl}'(\lambda)$. Otherwise, the algorithm that simply runs a compressed oracle simulation of A and measures the compressed database at the end, recovers a collision with non-negligible probability. We restrict to this "good" subset of $\mathcal{W}$ from here on.

We are now ready to apply Corollary 117 with ε_1, δ_1, and δ_2 as above. We deduce that there exists $\mathcal{H}_{good} \subseteq \mathcal{F}(\{0,1\}^n \smallsetminus \{\tilde{x}_0, \tilde{x}_1\}, \{0,1\})$ such that

(a) $\sum_{\tilde{H} \in \mathcal{H}_{good}} \| \Pi_{\tilde{H}} \left| \Psi_0 \right\rangle \|^2 \geq \left(1 - \sqrt{\Xi(\varepsilon_1, \delta_1, \delta_2)}\right) \cdot \| \left| \tilde{y} \right\rangle \left\langle \tilde{y} \right| \left| \Psi_{final} \right\rangle \|^2 \geq poly(\varepsilon) - \mathsf{negl}''(\lambda)$,

(b) for all $\tilde{H} \in \mathcal{H}_{good}$, $b \in \{0,1\}$,

$$\frac{\mathbb{E}_{c_0, c_1 \leftarrow \{0,1\}} \mathbb{E}_{l \leftarrow [q]} [\| \left| \tilde{x}_b \right\rangle \left\langle \tilde{x}_b \right| (U' O^{H}_{(x_0, c_0), (x_1, c_1)})^l \left| \tilde{H} \right\rangle \left\langle \tilde{H} \right| \mathsf{Decomp} \left| \Psi_0 \right\rangle \|^2]}{\| \Pi_{\tilde{H}} \left| \Psi_0 \right\rangle \|^2}$$
$$\geq \frac{1 - \sqrt{\Xi(\varepsilon_1, \delta_1, \delta_2)} - \Xi(\varepsilon_1, \delta_1, \delta_2)}{8 \tilde{q}} \cdot \| \left| \tilde{y} \right\rangle \left\langle \tilde{y} \right| \left| \Psi_{final} \right\rangle \|^2 \geq \frac{poly(\varepsilon)}{\tilde{q}} - \mathsf{negl}''(\lambda) .$$

where negl'' is a non-negligible function. To obtain the final inequalities in (a) and (b) we used the bounds on ε_1, δ_1, and δ_2 and Equation (89).

Now, notice that, at step (ii) of Algorithm 1, conditioned on $(\widetilde{adv}, \tilde{y}, \tilde{G}_0, \tilde{G}_1, \tilde{h})$, the state $\left| \Psi_0 \right\rangle$ takes the form $\left| \Psi_0 \right\rangle = \left| adv \right\rangle_{\mathsf{work,query}} \otimes \left| \Phi \right\rangle_{\mathsf{O}}$, where $\left| \Phi \right\rangle_{\mathsf{O}}$ is some state on the compressed database register for H that can depend on $\widetilde{adv}, \tilde{y}, \tilde{G}_0, \tilde{G}_1, \tilde{h}$. Now, let $\mathcal{H}_{good} \subseteq \mathcal{F}(\{0,1\}^n \smallsetminus \{\tilde{x}_0, \tilde{x}_1\}, \{0,1\})$ be the set that is guaranteed to exist from the argument above. In the following calculation, we abbreviate $\mathcal{F}(\{0,1\}^n \smallsetminus \{\tilde{x}_0, \tilde{x}_1\}, \{0,1\})$ as $\mathcal{F}_{\tilde{x}_0, \tilde{x}_1}$.

Then, for $b, b' \in \{0,1\}$,

$$\Pr[\text{Algorithm 1 outputs } x_b, x_{b'} \mid i = i^* \wedge (adv, \mathsf{h}^{i^*}_{\mathsf{data}}, G_0, G_1, h) = (\widetilde{adv}, \tilde{y}, \tilde{G}_0, \tilde{G}_1, \tilde{h})]$$

$$= \sum_{\tilde{H} \in \mathcal{F}_{x_0, x_1}} \Bigg(\| \Pi_{\tilde{H}} \left| adv \right\rangle \otimes \left| \Phi \right\rangle \|^2$$
$$\cdot \mathbb{E}_{\substack{c_0, c_1 \leftarrow \{0,1\} \\ c_0, c'_1 \leftarrow \{0,1\} \\ j, j' \leftarrow [q]}} \Big[\| \big(\left| \tilde{x}_b \right\rangle \left\langle \tilde{x}_b \right| \otimes \left| \tilde{x}_{b'} \right\rangle \left\langle \tilde{x}_{b'} \right| \big) \big((U' O^{\tilde{H}}_{(x_0, c_0), (x_1, c_1)})^j \otimes (U' O^{\tilde{H}}_{(x_0, c'_0), (x_1, c'_1)})^{j'} \big) \left| adv \right\rangle_{\mathsf{work,query}} \otimes \left| adv \right\rangle_{\mathsf{work', query'}} \|^2 \Big] \Bigg)$$

$$= \sum_{\tilde{H} \in \mathcal{F}_{x_0, x_1}} \Bigg(\| \Pi_{\tilde{H}} \left| adv \right\rangle \otimes \left| \Phi \right\rangle \|^2 \cdot \mathbb{E}_{c_0, c_1 \leftarrow \{0,1\}} \mathbb{E}_{j, j' \leftarrow [q]} \Big[\| \left| \tilde{x}_b \right\rangle \left\langle \tilde{x}_b \right| (U' O^{\tilde{H}}_{(x_0, c_0), (x_1, c_1)})^j \left| adv \right\rangle \|^2 \Big]$$
$$\cdot \mathbb{E}_{c'_0, c'_1 \leftarrow \{0,1\}} \mathbb{E}_{j' \leftarrow [q]} \Big[\| \left| \tilde{x}_{b'} \right\rangle \left\langle \tilde{x}_{b'} \right| (U' O^{\tilde{H}}_{(x_0, c_0), (x_1, c_1)})^{j'} \left| adv \right\rangle \|^2 \Big] \Bigg)$$

$$= \sum_{\tilde{H} \in \mathcal{F}_{x_0, x_1}} \Bigg(\| \Pi_{\tilde{H}} \left| \Psi_0 \right\rangle \|^2 \cdot \frac{\mathbb{E}_{c_0, c_1 \leftarrow \{0,1\}} \mathbb{E}_{j \leftarrow [q]} [\| \left| \tilde{x}_b \right\rangle \left\langle \tilde{x}_b \right| (U' O^{\tilde{H}}_{(x_0, c_0), (x_1, c_1)})^j \left| \tilde{H} \right\rangle \left\langle \tilde{H} \right| \mathsf{Decomp} \left| \Psi_0 \right\rangle \|^2]}{\| \Pi_{\tilde{H}} \left| \Psi_0 \right\rangle \|^2}$$
$$\cdot \frac{\mathbb{E}_{c'_0, c'_1 \leftarrow \{0,1\}} \mathbb{E}_{j' \leftarrow [q]} [\| \left| \tilde{x}_{b'} \right\rangle \left\langle \tilde{x}_{b'} \right| (U' O^{\tilde{H}}_{(x_0, c_0), (x_1, c_1)})^{j'} \left| \tilde{H} \right\rangle \left\langle \tilde{H} \right| \mathsf{Decomp} \left| \Psi_0 \right\rangle \|^2]}{\| \Pi_{\tilde{H}} \left| \Psi_0 \right\rangle \|^2} \Bigg)$$

$$\geq \sum_{\tilde{H} \in \mathcal{H}_{good}} \| \Pi_{\tilde{H}} \left| \Psi_0 \right\rangle \|^2 \cdot \left(\frac{poly(\varepsilon)}{\tilde{q}} \right)^2 - \mathsf{negl}'' \qquad \text{using (b)}$$

$$\geq \frac{poly(\varepsilon)}{\tilde{q}^2} - \mathsf{negl}'' \qquad \text{using (a)},$$

where the first equality implicitly uses the equivalence between compressed an uncompressed simulations. All in all, we have

$$\Pr[\text{Algorithm 1 outputs } x_b, x_{b'}]$$
$$\geq \Pr[\text{Algorithm 1 outputs } x_b, x_{b'} \mid i = i^* \wedge (adv, \mathsf{h}^{i^*}_{\mathsf{data}}, G_0, G_1, h) \in \mathcal{W}] \cdot \Pr[i = i^* \wedge (adv, \mathsf{h}^{i^*}_{\mathsf{data}}, G_0, G_1, h) \in \mathcal{W}]$$
$$\geq \frac{poly(\varepsilon)}{Q \cdot \tilde{q}^2} - \mathsf{negl}'',$$

where we used Equation (88).

When $b \neq b'$, we get that Algorithm 1 outputs a collision with probability $\frac{poly(\varepsilon)}{Q \cdot \tilde{q}^2} - \mathsf{negl}'' \geq \frac{poly(\varepsilon)}{q^3} - \mathsf{negl}''$, where q is the total number of queries made by A to G_0, G_1, H, and H_d.

References

[AA09] Scott Aaronson and Andris Ambainis. ‘The need for structure in quantum speedups’. In: *arXiv preprint arXiv:0911.0996* (2009).

[AA15] Scott Aaronson and Andris Ambainis. ‘Forrelation: A Problem That Optimally Separates Quantum from Classical Computing’. In: *Proceedings of the Forty-Seventh Annual ACM Symposium on Theory of Computing*. STOC ’15. Portland, Oregon, USA: Association for Computing Machinery, 2015, 307–316. ISBN: 9781450335362. DOI: 10.1145/2746539.2746547.

[AGS22] Atul Singh Arora, Alexandru Gheorghiu and Uttam Singh. ‘Oracle Separations of Hybrid Quantum-Classical Circuits’. 5th Jan. 2022. DOI: 10.48550/arXiv.2201.01904. arXiv: 2201.01904 [quant-ph].

[AHU19] Andris Ambainis, Mike Hamburg and Dominique Unruh. ‘Quantum Security Proofs Using Semi-classical Oracles’. In: *Advances in Cryptology – CRYPTO 2019*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Springer International Publishing, 2019, pp. 269–295. ISBN: 978-3-030-26951-7. DOI: https://doi.org/10.1007/978-3-030-26951-7_10.

[AIK06] Benny Applebaum, Yuval Ishai and Eyal Kushilevitz. ‘Cryptography in $NC0̂$’. In: *SIAM Journal on Computing* 36.4 (Jan. 2006), pp. 845–888. ISSN: 0097-5397, 1095-7111. DOI: 10.1137/S0097539705446950. URL: http://epubs.siam.org/doi/10.1137/S0097539705446950 (visited on 21/04/2021).

[AS04] Scott Aaronson and Yaoyun Shi. ‘Quantum lower bounds for the collision and the element distinctness problems’. In: *Journal of the ACM (JACM)* 51.4 (2004), pp. 595–605.

[Aar05] Scott Aaronson. *Ten Semi-Grand Challenges for Quantum Computing Theory*. 2005. URL: https://www.scottaaronson.com/writings/qchallenge.html.

[Aar10] Scott Aaronson. ‘BQP and the Polynomial Hierarchy’. In: *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*. STOC ’10. Cambridge, Massachusetts, USA: Association for Computing Machinery, 2010, 141–150. ISBN: 9781450300506. DOI: 10.1145/1806689.1806711.

[Aar13] Scott Aaronson. ‘The Equivalence of Sampling and Searching’. In: *Theory of Computing Systems* 55 (2013), pp. 281–298.

[Ajt96] Miklós Ajtai. ‘Generating hard instances of lattice problems’. In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. 1996, pp. 99–108.

[BBBF18] Dan Boneh, Joseph Bonneau, Benedikt Bünz and Ben Fisch. ‘Verifiable Delay Functions’. In: *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*. Ed. by Hovav Shacham and Alexandra Boldyreva. Vol. 10991. Lecture Notes in Computer Science. Springer, 2018, pp. 757–788. DOI: 10.1007/978-3-319-96884-1_25.

[BBD+09] Hans J Briegel, David E Browne, Wolfgang Dür, Robert Raussendorf and Maarten Van den Nest. ‘Measurement-based quantum computation’. In: *Nature Physics* 5.1 (2009), pp. 19–26.

[BGJ+16] Nir Bitansky, Shafi Goldwasser, Abhishek Jain, Omer Paneth, Vinod Vaikuntanathan and Brent Waters. ‘Time-lock puzzles from randomized encodings’. In: *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*. 2016, pp. 345–356.

[BGK18] Sergey Bravyi, David Gosset and Robert König. ‘Quantum advantage with shallow circuits’. In: *Science* 362.6412 (2018), pp. 308–311.

[BKVV20] Zvika Brakerski, Venkata Koppula, Umesh V. Vazirani and Thomas Vidick. ‘Simpler Proofs of Quantumness’. In: *15th Conference on the Theory of Quantum Computation, Communication and Cryptography, TQC 2020, June 9-12, 2020, Riga, Latvia*. Ed. by Steven T. Flammia. Vol. 158. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, 8:1–8:14. DOI: 10.4230/LIPIcs.TQC.2020.8. URL: https://doi.org/10.4230/LIPIcs.TQC.2020.8.

[BLZ21] Jeremiah Blocki, Seunghoon Lee and Samson Zhou. ‘On the Security of Proofs of Sequential Work in a Post-Quantum World’. 18th May 2021. arXiv: 2006.10972 [cs]. URL: http://arxiv.org/abs/2006.10972 (visited on 25/05/2022).

[CCD+03] Andrew M Childs, Richard Cleve, Enrico Deotto, Edward Farhi, Sam Gutmann and Daniel A Spielman. ‘Exponential algorithmic speedup by a quantum walk’. In: *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*. 2003, pp. 59–68.

[CCL20] Nai-Hui Chia, Kai-Min Chung and Ching-Yi Lai. ‘On the Need for Large Quantum Depth’. In: *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2020. New York, NY, USA: Association for Computing Machinery, 8th June 2020, pp. 902–915. ISBN: 978-1-4503-6979-4. DOI: 10.1145/3357713.3384291.

[CDGS18] Sandro Coretti, Yevgeniy Dodis, Siyao Guo and John P. Steinberger. ‘Random Oracles and Non-uniformity’. In: *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part I*. Ed. by Jesper Buus Nielsen and Vincent Rijmen. Vol. 10820. Lecture Notes in Computer Science. Springer, 2018, pp. 227–258. DOI: 10.1007/978-3-319-78381-9_9. URL: https://doi.org/10.1007/978-3-319-78381-9_9.

[CGLQ20] Kai-Min Chung, Siyao Guo, Qipeng Liu and Luowen Qian. ‘Tight Quantum Time-Space Tradeoffs for Function Inversion’. In: *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*. Ed. by Sandy Irani. IEEE, 2020, pp. 673–684. DOI: 10.1109/FOCS46700.2020.00068. URL: https://doi.org/10.1109/FOCS46700.2020.00068.

[CGV22] Andrea Coladangelo, Shafi Goldwasser and Umesh Vazirani. ‘Deniable encryption in a Quantum world’. In: *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*. 2022, pp. 1378–1391.

[CH22] Nai-Hui Chia and Shih-Han Hung. *Classical verification of quantum depth*. 2022. DOI: 10.48550/ARXIV.2205.04656. URL: https://arxiv.org/abs/2205.04656.

[CM20] Matthew Coudron and Sanketh Menda. ‘Computations with Greater Quantum Depth Are Strictly More Powerful (Relative to an Oracle)’. In: *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2020. 2020, pp. 889–901. ISBN: 978-1-4503-6979-4. DOI: 10.1145/3357713.3384269. URL: https://doi.org/10.1145/3357713.3384269.

[CW00] R. Cleve and J. Watrous. ‘Fast Parallel Circuits for the Quantum Fourier Transform’. In: *Proceedings 41st Annual Symposium on Foundations of Computer Science*. Vol. 1. 2000, p. 526. DOI: 10.1109/SFCS.2000.892140. URL: https://doi.ieeecomputersociety.org/10.1109/SFCS.2000.892140.

[FGG14] Edward Farhi, Jeffrey Goldstone and Sam Gutmann. ‘A quantum approximate optimization algorithm’. In: *arXiv preprint arXiv:1411.4028* (2014).

[FSS84] Merrick Furst, James B Saxe and Michael Sipser. ‘Parity, circuits, and the polynomial-time hierarchy’. In: *Mathematical systems theory* 17.1 (1984), pp. 13–27.

[GR08] Venkatesan Guruswami and Atri Rudra. ‘Explicit Codes Achieving List Decoding Capacity: Error-Correction With Optimal Redundancy’. In: *IEEE Transactions on Information Theory* 54.1 (2008), pp. 135–150. DOI: 10.1109/TIT.2007.911222.

[HG22] Atsuya Hasegawa and François Le Gall. *An optimal oracle separation of classical and quantum hybrid schemes*. 2022. DOI: 10.48550/ARXIV.2205.04633. URL: https://arxiv.org/abs/2205.04633.

[Has86] John Hastad. ‘Almost optimal lower bounds for small depth circuits’. In: *Proceedings of the eighteenth annual ACM symposium on Theory of computing*. 1986, pp. 6–20.

[Joz05] Richard Jozsa. ‘An introduction to measurement based quantum computation’. In: *Quantum Information Processing* 199 (Sept. 2005).

[Kra03] V.Y. Krachkovsky. ‘Reed-Solomon codes for correcting phased error bursts’. In: *IEEE Transactions on Information Theory* 49.11 (2003), pp. 2975–2984. DOI: 10.1109/TIT.2003.819333.

[Mil92] Peter Bro Miltersen. ‘Circuit Depth Relative to a Random Oracle’. In: *Inf. Process. Lett.* 42.6 (1992), pp. 295–298. DOI: 10.1016/0020-0190(92)90225-K. URL: https://doi.org/10.1016/0020-0190(92)90225-K.

[PGWPR06] David Pérez-García, Michael M. Wolf, Denes Petz and Mary Beth Ruskai. 'Contractivity of positive and trace-preserving maps under Lp norms'. In: *Journal of Mathematical Physics* 47.8 (Aug. 2006), p. 083506. DOI: 10.1063/1.2218675.

[PMS+14] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik and Jeremy L. O'Brien. 'A variational eigenvalue solver on a photonic quantum processor'. In: *Nature Communications* 5.1 (July 2014). DOI: 10.1038/ncomms5213.

[PS19] Chris Peikert and Sina Shiehian. 'Noninteractive zero knowledge for NP from (plain) learning with errors'. In: *Annual International Cryptology Conference*. Springer. 2019, pp. 89–114.

[RB01] Robert Raussendorf and Hans J Briegel. 'A one-way quantum computer'. In: *Physical review letters* 86.22 (2001), p. 5188.

[RR94] Alexander A Razborov and Steven Rudich. 'Natural proofs'. In: *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*. 1994, pp. 204–213.

[RSW96] Ronald L Rivest, Adi Shamir and David A Wagner. 'Time-lock puzzles and timed-release crypto'. In: (1996).

[Sim97] Daniel R. Simon. 'On the Power of Quantum Computation'. In: *SIAM Journal on Computing* 26.5 (1997), pp. 1474–1483. DOI: 10.1137/S0097539796298637. eprint: https://doi.org/10.1137/S0097539796298637. URL: https://doi.org/10.1137/S0097539796298637.

[WKST19] Adam Bene Watts, Robin Kothari, Luke Schaeffer and Avishay Tal. 'Exponential separation between shallow quantum circuits and unbounded fan-in shallow classical circuits'. In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. 2019, pp. 515–526.

[YZ22] Takashi Yamakawa and Mark Zhandry. 'Verifiable Quantum Advantage without Structure'. 5th Apr. 2022. arXiv: 2204.02063 [quant-ph]. URL: http://arxiv.org/abs/2204.02063 (visited on 12/05/2022).

[Zha19] Mark Zhandry. 'How to record quantum queries, and applications to quantum indifferentiability'. In: *Annual International Cryptology Conference*. Springer. 2019, pp. 239–268.

Appendix

A The O2H lemma

The following proofs of the O2H lemma (due originally to [AHU19; CCL20]) as used in our setting are taken almost verbatim from [AGS22].

A.1 Proof of Lemma 41

Proof of Lemma 41. We begin by assuming that $\mathcal{L}$ and S are fixed (and so is $\mathcal{G}$). In that case, we can assume ρ is pure. If not, we can purify it and absorb it in the work register. (The general case should follow from concavity). From Remark 40, we have

$$|\psi_L\rangle := \mathcal{L}U\,|\psi\rangle_{Q'} \overset{40}{=} |\phi_0\rangle_{Q'} + |\phi_1\rangle_{Q'}\,.$$
$$\mathcal{L}U_S U\,|\psi\rangle_{Q'}\,|0\rangle_B = |\phi_0\rangle_{Q'}\,|0\rangle_B + |\phi_1\rangle_{Q'}\,|1\rangle_B$$

where Q' is a shorthand for QRW. Similarly let

$$|\psi_G\rangle := \mathcal{G}U\,|\psi\rangle_{Q'} = |\phi_0\rangle_{Q'} + \left|\phi_1^\perp\right\rangle_{Q'}$$

where note that

$$\left\langle\phi_1|\phi_1^\perp\right\rangle_{QRW} = 0 \tag{90}$$

because $|\phi_1\rangle$ and $|\phi_1^\perp\rangle$ are the states where the queries were made on S, and on S $\mathcal{G}$ responds with $\perp$ while $\mathcal{L}$ does not. Further, we analogously have

$$\mathcal{G}U_S U\,|\psi\rangle_{Q'}\,|0\rangle_B = |\phi_0\rangle_{Q'}\,|0\rangle_B + \left|\phi_1^\perp\right\rangle_{Q'}\,|1\rangle_B\,.$$

We show that the difference between $|\psi_L\rangle$ and $|\psi_G\rangle$ is bounded by $P_{\text{find}}(\mathcal{L},S) := \Pr[\text{find} : U^{\mathcal{L}\backslash S}, \rho]$, which in turn can be used to bound the quantity in the statement of the lemma.

$$\begin{aligned}
\left\||\psi_L\rangle - |\psi_G\rangle\right\|^2 &= \left\||\phi_1\rangle - \left|\phi_1^\perp\right\rangle\right\|^2 \\
&\overset{(90)}{=} \left\||\phi_1\rangle\right\|^2 + \left\|\left|\phi_1^\perp\right\rangle\right\|^2 \\
&= 2\left\||\phi_1\rangle\right\|^2 && \because \left\||\phi_1\rangle\right\|^2 = \left\|\left|\phi_1^\perp\right\rangle\right\|^2 = 1 - \left\||\phi_0\rangle\right\|^2 \\
&= 2P_{\text{find}}(\mathcal{L},S).
\end{aligned}$$

If $\mathcal{L}$ and S are random variables drawn from a (possibly) joint distribution $\Pr(\mathcal{L},S)$, the analysis can be generalised as follows. Let

$$\rho_L := \sum_{\mathcal{L},S} \Pr(\mathcal{L},S)\,|\psi_L\rangle\,\langle\psi_L|$$
$$\rho_G := \sum_{\mathcal{L},S} \Pr(\mathcal{L},S)\,|\psi_G\rangle\,\langle\psi_G|$$

where $|\psi_G\rangle$ is fixed by $\mathcal{L}$ and S because G itself is fixed once $\mathcal{L}$ and S is fixed (by assumption). One can then use monotonicity of fidelity to obtain

$$\begin{aligned}
F(\rho_L,\rho_G) &\geq \sum_{L,S} \Pr(\mathcal{L},S) F(|\psi_L\rangle\,,|\psi_G\rangle) \\
&\geq 1 - \frac{1}{2}.\sum_{L,S} \Pr(\mathcal{L},S)\,\left\||\psi_L\rangle - |\psi_G\rangle\right\|^2 && \because 1 - \frac{1}{2}F(|a\rangle\,,|b\rangle) \geq \left\||a\rangle - |b\rangle\right\|^2 \\
&\geq 1 - \cancel{\frac{1}{2}} \sum_{L,S} \Pr(\mathcal{L},S)\cancel{2}P_{\text{find}}(\mathcal{L},S) \\
&= 1 - P_{\text{find}}
\end{aligned}$$

where P_{find} is the expectation of $P_{\text{find}}(\mathcal{L},S)$ over $\mathcal{L}$ and S. It is known that the trace distance bounds the LHS of the Lemma and the trace distance itself is bounded by $\sqrt{2-2F}$.

□

A.2 Proof of Lemma 42

Proof of Lemma 42. We resume the use of boldface for the query and response registers as they do play an active role in the discussion. Let us begin with the case where the oracle is applied only once, i.e. $\boldsymbol{Q}$ is a single query register Q. Since the RW registers don't play any significant role, we denote it by L. Let

$$\begin{aligned} U\left|\psi\right\rangle &= \sum_{q,l} \psi(q,l) \left|q,l,0\right\rangle_{QLB} \\ \implies U_S U \left|\psi\right\rangle &= \sum_{q\notin S}\left(\sum_{r,l}\psi(q,l)\left|q,l\right\rangle_{QL}\right)\left|0\right\rangle_B + \sum_{q\in S}\left(\sum_{r,l}\psi(q,l)\left|q,l\right\rangle_{QL}\right)\left|1\right\rangle_B . \end{aligned}$$

Since $\mathcal{L}$ leaves registers QB unchanged,

$$\begin{aligned} \mathrm{tr}[\mathbb{I}_{QL}\otimes\left|1\right\rangle\left\langle 1\right|_B(\mathcal{L}\circ U_S\circ U\circ\left|\psi\right\rangle\left\langle\psi\right|)] &= \mathrm{tr}[\mathbb{I}_{QL}\otimes\left|1\right\rangle\left\langle 1\right|_B(U_S\circ U\circ\left|\psi\right\rangle\left\langle\psi\right|)] \\ &= \sum_q \left|\psi(q)\right|^2 \chi_S(q) \end{aligned}$$

where $\psi(q) = \sum_l \psi(q,l)$ and χ_S is the characteristic function for S, i.e.

$$\chi_S(q) = \begin{cases} 1 & q\in S \\ 0 & q\notin S. \end{cases}$$

We are yet to average over the random variable S. Clearly, $\mathbb{E}(\chi_S(q)) = \Pr[q\in S] \le p$, yielding

$$\Pr[\mathrm{find}: U^{\mathcal{L}\backslash S},\rho] \le p.$$

In the general case, everything goes through unchanged except the string q is now a set of strings $\boldsymbol{q}$ and

$$\chi_S(\boldsymbol{q}) = \begin{cases} 1 & \boldsymbol{q}\cap S \neq \varnothing \\ 0 & \boldsymbol{q}\cap S = \varnothing. \end{cases}$$

Consequently, one evaluates $\mathbb{E}(\chi_S(\boldsymbol{q})) = \Pr[\boldsymbol{q}\cap S\neq\varnothing] \le |\boldsymbol{q}|\cdot p = \bar{q}\cdot p$, by the union bound, yielding

$$\Pr[\mathrm{find}: U^{\mathcal{L}\backslash S},\rho] \le \bar{q}\cdot p.$$

□

B Misc calculations

B.1 Proof of Claim 56 — Deferred steps

B.1.1 Proof of Equation (6)

First, note that $2^\delta \frac{|S_{i-1}|}{|S_{i-1,i}|} \le 2^\delta \frac{1}{|\Sigma|}$ because $|S_{i-1}| = \Sigma$ and $1/|S_{i-1,i}| \le 1/|\Sigma|^2$ by construction (see Algorithm 72 or Algorithm 50 for simplicity).

Second, observe that

$$\begin{aligned} \frac{|S_{ii}| - |S_i|}{|S_{i-1,i}| - |S_i|} &\le \frac{|S_{ii}|}{|S_{i-1,i}| - |S_i|} \\ &= \frac{\frac{|S_{ii}|}{|S_{i-1,i}|}}{1 - \frac{|S_i|}{|S_{i-1,i}|}} \\ &= \frac{1}{|\Sigma|}\left(1 - \frac{|S_i|}{|S_{i-1,i}|}\right)^{-1} \\ &\le \frac{1}{|\Sigma|}\left(1 + \frac{|S_i|}{|S_{i-1,i}|} + \epsilon\right) \\ &\le \frac{1}{|\Sigma|}\left(1 + \frac{1}{|\Sigma|^2} + \epsilon\right) \end{aligned}$$

where ϵ is a small fixed constant ϵ and we used the fact that the inequality $(1-x)^{-1} \leq 1 + x + \epsilon$ holds for a small enough $0 \leq x$. Combining these, and recalling $|\Sigma| = 2^{\lambda^{\Theta(1)}}$ and $n = \Theta(\lambda)$, one obtains Equation (6).

B.2 Proof of Claim 106

We prove the following.

Claim 106. Let E and F be random variables taking values in $[0,1]$. Let $\gamma \in [0,1]$.

$$\Pr[E \geq \gamma \cdot F] \leq 1 - \mathbb{E}(F)\left(1 - \frac{\mathbb{E}(E)}{\gamma \cdot \mathbb{E}(F)}\right)$$

Proof. Let $p := \Pr[E \geq \gamma \cdot F]$. Then,

$$\mathbb{E}[E] = p \cdot \mathbb{E}[E \,|\, E \geq \gamma \cdot F] + (1-p) \cdot \mathbb{E}[E \,|\, E < \gamma \cdot F] \tag{91}$$

Notice that $\mathbb{E}[E \,|\, E \geq \gamma F] \geq \gamma \cdot \mathbb{E}[F \,|\, E \geq \gamma F]$. Plugging this into (91), we get

$$\begin{aligned} \mathbb{E}[E] &= p \cdot \gamma \cdot \mathbb{E}[F \,|\, E \geq \gamma \cdot F] + (1-p) \cdot \mathbb{E}[E \,|\, E < \gamma \cdot F] \\ &\geq p \cdot \gamma \cdot \mathbb{E}[F \,|\, E \geq \gamma \cdot F]\,. \end{aligned} \tag{92}$$

Now, notice that

$$\mathbb{E}[F] = p \cdot \mathbb{E}[F \,|\, E \geq \gamma \cdot F] + (1-p) \cdot \mathbb{E}[F \,|\, E < \gamma \cdot F] \tag{93}$$

Rearranging the latter, and plugging this into (92) gives

$$\begin{aligned} \mathbb{E}[E] &\geq \gamma \cdot (\mathbb{E}[F] - (1-p) \cdot \mathbb{E}[F \,|\, E < \gamma \cdot F]) \\ &\geq \gamma \cdot (\mathbb{E}[F] - (1-p)) \end{aligned}$$

Solving for p gives the desired inequality. □

C Sampling argument for Permutations

To keep the proof self-contained, we include the proof of the sampling argument for permutations, taken almost verbatim[70] from [AGS22]. The key idea has been adapted from [CDGS18] and slightly generalised.

C.1 Sampling argument for Uniformly Distributed Permutations

C.1.1 Convex Combination of Random Variables

We first make the notion of "convex combination of random variables" precise. Consider a function f which acts on a random permutation, say t, to produce an output, i.e. $f(t) = r$ where r is an element in the range of f.[71] This range can be arbitrary. We say a *convex combination* $\sum_i p_i t_i$ of random variables t_i is *equivalent* to t if for all functions f, and all outputs s in its range, $\sum_i p_i \Pr[f(t_i) = s] = \Pr[f(t) = s]$. This relation is denoted by $\sum_i p_i t_i \equiv t$.

C.1.2 The "parts" notation

While permutations are readily defined as an ordered set of distinct elements, it would nonetheless be useful to introduce what we call the "parts" notation which allows one to specify parts of the permutation.

Notation 120. Consider a permutation t over N elements, labelled $\{0, 1 \ldots N-1\}$.

- *Parts:* Let $S = \{(x_i, y_i)\}_{i=1}^M$ denote the mapping of $M \leq N$ elements under some permutation, i.e. there is some permutation t, such that $t(x_i) = y_i$. Call any such set S a "part" and its constituents "paths".
  - Denote by $\Omega_{\text{parts}}(N)$ the set of all such "parts".

[70] We fixed some notation.

[71] The function will later be interpreted as an algorithm and the random permutation accessed via an oracle.

- Call two parts $S = \{(x_i, y_i)\}_i$ and $S' = \{(x'_{i'}, y'_{i'})\}_{i'}$ *distinct* if for all i, i' (a) $x_i \neq x_{i'}$, and (b) there is a permutation t such that $t(x_i) = y_i$ and $t(x_{i'}) = y_{i'}$.
- Denote by[72] $\Omega_{\text{parts}}(N, S)$ the set of all parts $S' \in \Omega_{\text{parts}}(N)$ such that S' is distinct from S.

- *Parts in t:* The probability that t maps the elements as described by S may be expressed as $\Pr[\wedge_{i=1}^{M}(t(x_i) = y_i)] = \Pr[S \subseteq \text{paths}(t)]$ where $\text{paths}(t) := \{(x, t(x))\}_{x=0}^{N-1}$.
- *Conditioning t based on parts:* Finally, use the notation t_S to denote the random variable t conditioned on $S \subseteq \text{paths}(t)$.

To clarify the notation, consider the following simple example.

Example 121. Let $N = 2$. Then $\Omega_{\text{parts}}(N) = \{\{(0,0)\}, \{(1,1)\}, \{(0,0),(1,1)\}, \{(0,1)\}, \{(1,0)\}, \{(0,1),(1,0)\}\}$ and there are only two permutations, $t(x) = x$ and $t'(x) = x \oplus 1$ for all $x \in \{0,1\}$. An example of a part S is $S = \{(0,0)\}$. A part (in fact the only part) distinct from S is $(1,1)$, i.e. $\Omega_{\text{parts}}(N, S) = \{(1,1)\}$.

C.1.3 δ non-uniform distributions

Using the "parts" notation (see Notation 120), we define uniform distributions over permutations and a notion of being δ non-uniform—distributions which are at most δ "far from" being being uniform.[73]

Definition 122 (uniform and δ non-uniform distributions)**.** Consider the set, $\Omega(N)$, of all possible permutations of N objects labelled $\{0, 1, 2 \ldots N-1\}$. Let $\mathbb{F}$ be a distribution over Ω. Call $\mathbb{F}$ a *uniform distribution* if for $u \sim \mathbb{F}$, $\Pr[S \subseteq \text{paths}(u)] = \frac{(N-M)!}{N!}$ for all parts $S \in \Omega_{\text{parts}}(N)$ where we are using Notation 120.

An arbitrary distribution $\mathbb{F}^{\delta}$ over Ω is δ non-uniform if it satisfies for $t \sim \mathbb{F}^{\delta}$

$$\Pr[S \subseteq \text{paths}(t)] \leq 2^{\delta|S|} \cdot \Pr[S \subseteq \text{paths}(u)]$$

for all parts $S \in \Omega_{\text{parts}}(N)$.

Finally, $\mathbb{F}^{p,\delta}$ over Ω is (p, δ) non-uniform if there is a subset of parts S of size $|S| \leq p$ such that the distribution conditioned on S specifying a part of the permutation, becomes δ non-uniform over parts distinct from S. Formally, let $t' \sim \mathbb{F}^{p,\delta}$. Then t' is (p, δ) non-uniformly distributed if t'_S is δ non-uniformly distributed over all $S' \in \Omega_{\text{parts}}(N, S)$ (see Notation 120), i.e.

$$\Pr[S' \subseteq \text{paths}(t') | S \subseteq \text{paths}(t')] \leq 2^{\delta|S'|} \cdot \Pr[S' \subseteq \text{paths}(u) | S \subseteq \text{paths}(u)]. \tag{94}$$

In Equation (94), we are conditioning a uniform distribution using the "paths/parts" notation which may be confusing. The following should serve as a clarification.

Note 123. Let $u \sim \mathbb{F}$ as above. Then, we have $\Pr[S' \subseteq \text{paths}(u) | S \subseteq \text{paths}(u)] = \frac{(N-|S|-|S'|)!}{(N-|S|)!}$ where $S' \in \Omega_{\text{parts}}(N, S)$ and $S \in \Omega_{\text{parts}}(N)$. Let $S = \{(x_i, y_i)\}_{i=1}^{|S|}$. Then, the conditioning essentially specifies that the $|S|$ elements in $X = (x_i)_{i=1}^{|S|}$ must be mapped to $Y = (y_i)_{i=1}^{|S|}$ by u, i.e. $u(x_i) = y_i$, but the remaining elements $\{0, 1 \ldots N-1\} \backslash X$ are mapped uniformly at random to $\{0, 1 \ldots N-1\} \backslash Y$.

Clearly, for $\delta = 0$, the δ non-uniform distribution becomes a uniform distribution. However, this can be achieved by relaxing the uniformity condition in many ways. The δ non-uniform distribution is defined the way it is to have the following property. Notice that $|S|$ appears in a form such that the product of two probabilities, $\Pr[S_1 \subseteq \text{parts}(t)]$ and $\Pr[S_2 \subseteq \text{parts}(t)]$ yields $|S_1| + |S_2|$, e.g. $(1+\delta)^{|S|}$ instead of $2^{\delta|S|}$ would also have worked.[74] This property plays a key role in establishing that in the main decomposition (as described informally in Subsection C.1), the number of "paths" (in the informal discussion it was bits) fixed is small. We chose the pre-factor $2^{|S|}$ for convenience—unlikely events in our analysis are those which are exponentially suppressed, and we therefore take the threshold parameter to be $\gamma = 2^{-m}$. These choices result in a simple relation between $|S|$ and m.

Notation 124. To avoid double negation, we use the phrase "t is more than δ non-uniform" to mean that t is not δ non-uniform. Similarly, we use the phrase "t is at most δ non-uniform" to mean that t is δ non-uniform.

[72] We use Ω_{parts} because the symbol Ω is often used for the sample space; for parts, Ω_{parts} plays an analogous role.

[73] Clarification to a possible conflict in terms: We use the word uniform in the sense of probabilities—a uniformly distributed random variable—and not quite in the complexity theoretic sense—produced by some Turing Machine without advice.

[74] The former was chosen by Chia, Chung and Lai [CCL20] while the latter by Coretti, Dodis, Guo and Steinberger [CDGS18] and possibly others.

As shall become evident, the only property of a uniform distribution we use in proving the main proposition of this section, is the following. It not only holds for all distributions over permutations, but also for d-Shuffler. We revisit this later.

Note 125. Let t be a permutation sampled from an arbitrary distribution $\mathbb{F}'$ over $\Omega(N)$. Let $S, S' \subseteq \Omega_{\text{parts}}(N)$ be *distinct* parts (see Notation 120). Then,

$$\Pr[S \subseteq \text{paths}(t) \wedge S' \subseteq \text{paths}(t)] = \Pr[S \cup S' \subseteq \text{paths}(t)].$$

If $S \cap S' = \varnothing$ and the parts are not distinct, then both expressions vanish.

C.1.4 Advice on uniform yields δ non-uniform

We are now ready to state and prove the simplest variant of the main proposition of this section.

Proposition 126 ($\mathbb{F}|r' \equiv \text{conv}(\mathbb{F}^{p,\delta})$). *Premise:*

- *Let $u \sim \mathbb{F}$ where $\mathbb{F}$ is a uniform distribution over all permutations, Ω, on $\{0, 1 \dots N-1\}$, as in Definition 122 with $N = 2^n$.*
- *Let r be a random variable which is arbitrarily correlated to u, i.e. let $r = g(u)$ where g is an arbitrary function.*
- *Fix any $\delta > 0$, $\gamma = 2^{-m} > 0$ (m may be a function of n) and some string r'.*
- *Suppose*
$$\Pr[r = r'] \geq \gamma. \tag{95}$$
- *Let t denote the variable u conditioned on $r = r'$, i.e. let $t = u|(g(u) = r')$.*

Then, t is "γ-close" to a convex combination of finitely many (p, δ) non-uniform distributions, i.e.

$$t \equiv \sum_i \alpha_i t_i + \gamma' t'$$

where $t_i \sim \mathbb{F}_i^{p,\delta}$ and $\mathbb{F}_i^{p,\delta}$ is (p, δ) non-uniform with $p = \frac{2m}{\delta}$. The permutation t' is sampled from an arbitrary (but normalised) distribution over Ω and $\gamma' \leq \gamma$.

Proof. Suppose that t is more than δ non-uniformly distributed (see Definition 122 and Notation 124), otherwise then there is nothing to prove (set α_1 to 1, and t_i to t, remaining α_is and γ' to zero). Recall $\Omega_{\text{parts}}(N)$ is the set of all parts (see Notation 120). Let the subset $S \in \Omega_{\text{parts}}(N)$ be the maximal subset of paths (i.e. subset with the largest size) such that

$$\Pr[S \subseteq \text{paths}(t)] > 2^{\delta \cdot |S|} \cdot \Pr[S \subseteq \text{paths}(u)]. \tag{96}$$

Claim 127. Let S and t be as described above. The random variable t conditioned on being consistent with the paths in $S \in \Omega_{\text{parts}}(N)$, i.e. t_S, is δ non-uniformly distributed over $S' \subseteq \Omega_{\text{parts}}(N, S)$, is δ non-uniformly distributed.

We prove Claim 127 by contradiction. Suppose that t_S is "more than" δ non-uniform. Then, there exists some $S' \in \Omega_{\text{parts}}(N, S)$ such that

$$\Pr[S' \subseteq \text{paths}(t_S)] = \Pr[S' \subseteq \text{paths}(t) | S \subseteq \text{paths}(t)] > 2^{\delta \cdot |S'|} \cdot \Pr[S' \subseteq \text{paths}(u) | S \subseteq \text{paths}(u)]. \tag{97}$$

Since S' violates the δ non-uniformity condition for t_S, the idea is to see if the union $S \cup S'$ violates the δ non-uniformity condition for t. If it does, we have a contradiction because S was by assumption the maximal subset satisfying this property. Indeed,

$$\begin{aligned}
\Pr[S \cup S' \subseteq \text{paths}(t)] &= \Pr[S \subseteq \text{paths}(t) \wedge S' \subseteq \text{paths}(t)] && \because S \text{ and } S' \text{ are distinct} \\
&= \Pr[S \subseteq \text{paths}(t)] \Pr[S' \subseteq \text{paths}(t) | S \subseteq \text{paths}(t)] && \text{conditional probability} \\
&> 2^{\delta \cdot (|S| + |S'|)} \cdot \Pr[S \subseteq \text{paths}(u)] \Pr[S' \subseteq \text{paths}(u) | S \subseteq \text{paths}(u)] && \text{using (96) and (97)} \\
&= 2^{\delta \cdot |S \cup S'|} \cdot \Pr[S \cup S' \subseteq \text{paths}(u)] && \because S \text{ and } S' \text{ are disjoint}
\end{aligned}$$

which completes the proof.

Claim 127 shows how to construct a δ non-uniform distribution after conditioning but we must also bound $|S|$. This is related to how likely is the r' we are conditioning upon, i.e. the probability of $g(u)$ being r'.

Claim 128. One has

$$|S| < \frac{m}{\delta}.$$

While Equation (96) lower bounds $\Pr[S \subseteq \text{paths}(t)]$, the upper bound is given by

$$\begin{aligned} \Pr[S \subseteq \text{paths}(t)] &= \Pr[S \subseteq \text{paths}(u)|(g(u) = r')] \\ &= \Pr[S \subseteq \text{paths}(u) \wedge g(u) = r'] / \Pr[g(u) = r'] \\ &\leq \Pr[S \subseteq \text{paths}(u) \wedge g(u) = r'] \cdot \gamma^{-1} \\ &\leq \Pr[S \subseteq \text{paths}(u)] \cdot \gamma^{-1} \end{aligned} \tag{98}$$

where recall that $\gamma = 2^{-m}$. Combining these, we have $2^{\delta \cdot |S|} < 2^m$, i.e., $|S| < \frac{m}{\delta}$.

Using Bayes rule on the event that $S \subseteq \text{paths}(t)$ we conclude that

$$t \equiv \alpha_1 t_1 + \alpha_1' t_1'$$

where $\alpha_1 = \Pr[S \subseteq \text{paths}(t)]$, $t_1 = t_S$, i.e. t conditioned on $S \subseteq \text{paths}(t)$, $\alpha_1' = 1 - \alpha_1$ and t_1' is t conditioned on $S \nsubseteq \text{paths}(t)$. Further, while t_1 is (p, δ) non-uniform (from Claim 127 and Claim 128), t_1' may not be. Proceeding as we did for t, if t_1' is itself δ non-uniform, there is nothing left to prove (we set $\alpha_2 = \alpha_1'$ and $t_2 = t_1'$ and the remaining α_is and γ' to zero). Also assume that $\alpha_1' > \gamma$ because otherwise, again, there is nothing to prove.

Therefore, suppose that t_1' is not δ non-uniform. Note that the proof of Claim 127 goes through for any permutation which is not δ non-uniform. Thus, the claim also applies to t_1' where we denote the maximal set of parts by S_1. Let t_2 be t_1' conditioned on $S_1 \subseteq \text{paths}(t_1')$ and t_2' be t_1' conditioned on $S_1 \nsubseteq \text{paths}(t_1')$. Using Bayes rule as before, we have

$$t \equiv \alpha_1 t_1 + \alpha_2 t_2 + \alpha_2' t_2'.$$

Adapting the statement of Claim 127 (with t_1' playing the role of t and S_1 playing the role of S) to this case, we conclude that t_2 is δ non-uniform but we still need to show that $|S_1| \leq p$. We need the analogue of Claim 128 which we assert is essentially unchanged.

Claim 129. One has

$$|S_i| < \frac{2m}{\delta}. \tag{99}$$

The proof is deferred to Subsection C.2. The factor of two appears because for the general case, we use both $\alpha_i' > \gamma$ and $\Pr[g(u) = r'] > \gamma$. One can iterate the argument above. Suppose

$$t \equiv \alpha_1 t_1 + \dots \alpha_j t_j + \alpha_j' t_j' \tag{100}$$

where $t_1, \dots t_j$ are (p, δ) non uniformly distributed while t_j' is not and for $\alpha_j' := \Pr[S \nsubseteq \text{paths}(t) \wedge \dots \wedge S_{j-1} \nsubseteq \text{paths}(t)]$ it holds that $\alpha_j' > \gamma$ (else one need not iterate). Let S_j be the maximal set such that $t_{j+1} := t_j'|S_j \subseteq \text{paths}(t_j')$ is δ non-uniform (which must exist from Claim 127) and let $t_{j+1}' := t_j'|S_j \nsubseteq \text{paths}(t_j')$. Let $\alpha_{j+1}' := \Pr[S_j \nsubseteq \text{paths}(t_j')]$ which equals $\Pr[S \nsubseteq \text{paths}(t) \wedge \dots \wedge S_j \nsubseteq \text{paths}(t)]$. From Claim 129, $|S_j| < 2m/\delta \leq p$ therefore t_{j+1} is (p, δ) non-uniform.

We now argue that the sum in Claim 129 contains finitely many terms. At every iteration, α_i' strictly decreases because at each step, more constraints are added; $S_i \neq S_j$ for all $i \neq j$ (otherwise conditioning on S_j (if $j \geq i$) as in Claim 127 could not have any effect). Since $\Omega_{\text{parts}}(N)$ is finite, the decreasing sequence $\alpha_1' \dots \alpha_i'$ must, for some integer i, satisfy $\alpha_i \leq \gamma$ after finitely many iterations. □

C.1.5 Iterating advice and conditioning on uniform distributions — δ non-β-uniform distributions

Once generalised to the d-Shuffler (which, as we shall, see is surprisingly simple), recall that the way we intend to use the above result is to repeatedly get advice from a quantum circuit, a role played by g in the previous discussion. However, the way it is currently stated, one starts with a uniformly distributed permutation u for which some advice $g(u)$ is given but one ends up with (p,δ) non-uniform distributions. We want the result to apply even when we start with a (p,δ) non-uniform distribution.

As should become evident shortly, the right generalisation of Proposition 126 for our purposes is as follows. Assume that the advice being conditioned occurs with probability at least $\gamma = 2^{-m}$ and think of m as being polynomial in n; $\delta > 0$ is some constant and $p = 2m/\delta$.

- Step 1: Let $t \sim \mathbb{F}^{\delta'}$ be δ' non-uniform[75] and $s \sim \mathbb{F}^{\delta'}|r$ be $t|(g(t) = r)$. Then it is straightforward to show that $s \equiv \sum_i \alpha_i s_i$ where s_i are $(p,\delta+\delta')$ non-uniform, which we succinctly write as
$$\mathbb{F}^{\delta'}|r \equiv \mathrm{conv}(\mathbb{F}^{p,\delta+\delta'}).$$

Observation: If $t \sim \mathbb{F}^{p,\delta}$ is (p,δ) non-uniform, then there is some S of size at most p such that $t \sim \mathbb{F}^{\delta|\beta}$ is δ non-β-uniform where[76] $\beta := (S)$. A β-uniform distribution is simply a uniform distribution conditioned on having S as parts. This amounts to basically making the conditioning explicit. Having this control will be of benefit later.

- Step 2: It is not hard to show that Step 1 goes through unchanged if non-uniform is replaced with non-β-uniform for an arbitrary β.

These combine to yield the following. Let $t \sim \mathbb{F}^{\delta'|\beta}$ be a δ' non-β-uniform distribution and $s \sim \mathbb{F}^{\delta'|\beta}|r$ be $t|(g(t) = r)$. Then $t \equiv \sum_i \alpha_i s_i$ where $s_i \sim \mathbb{F}^{p,\delta+\delta'|\beta}$ are $(p,\delta+\delta')$ non-β-uniform,[77] which we briefly express as
$$\mathbb{F}^{\delta'|\beta}|r \equiv \mathrm{conv}(\mathbb{F}^{p,\delta+\delta'|\beta}).$$

Observe that this composes well,
$$\mathbb{F}^{p,\delta+\delta'|\beta}|r \equiv \mathrm{conv}(\mathbb{F}^{2p,2\delta+\delta'|\beta}). \tag{101}$$

To see this, consider the following:

- For some S_i, s_i (as defined in the statement above) is $\delta'' := \delta + \delta'$ non-β'-uniform where $\beta' := (S \cup S_i)$ if $\beta = (S)$.
- With t set to s_i, β set to β', one can apply the above to get $s_i|(h(s_i) = r') \equiv \sum_i \alpha'_i q_i$ where q_i are $(p,\delta+\delta'')$ non-β'-uniform.
- Note that q_i are also $(2p, 2\delta+\delta')$ non-β-uniform; which we succinctly denoted as $\mathbb{F}^{2p,2\delta+\delta'|\beta}$.

Clearly, if this procedure is repeated $\tilde{n} \le \mathrm{poly}(n)$ times, starting from $\delta = 0$ and $\beta = (\varnothing)$, then the final convex combination would be over $\mathbb{F}^{\tilde{n}p,\tilde{n}\delta}$. As we shall see, for our use, it suffices to ensure that $\tilde{n}\delta$ is a small constant and that $\tilde{n}p = \frac{\tilde{n}m}{\delta} \le \mathrm{poly}(n)$. Choosing $\delta = \Delta/\tilde{n}$ for some small fixed $\Delta > 0$ yields $\tilde{n}\delta = \Delta$ and $\tilde{n}p = \frac{\tilde{n}^2 m}{\Delta}$ which is indeed bounded by $\mathrm{poly}(n)$ (recall m and $\tilde{n}$ are bounded by $\mathrm{poly}(n)$).

One can define a notion of closeness to any arbitrary distribution, as we did for closeness to uniform. To this end, first consider the following.

Definition 130 (δ non-$\mathbb{G}$ distributions—$\mathbb{G}^{\delta}$)**.** Let s be sampled from an arbitrary distribution, $\mathbb{G}$, over the set of all permutations $\Omega(N)$ of N objects and fix any $\delta > 0$.

Then, a distribution $\mathbb{G}^{\delta}$ is δ *non-*$\mathbb{G}$ if $s' \sim \mathbb{G}^{\delta}$ satisfies
$$\Pr[S \subseteq \mathrm{paths}(s')] \le 2^{\delta|S|} \cdot \Pr[S \subseteq \mathrm{paths}(s)]$$
for all $S \in \Omega_{\mathrm{parts}}(N)$.

Similarly, a distribution $\mathbb{G}^{p,\delta}$ is (p,δ) non-$\mathbb{G}$ if there is a subset $S' \in \Omega_{\mathrm{parts}}(N)$ of size at most $|S'| \le p$ such that conditioned on $S' \subseteq \mathrm{parts}(s)$, $s'' \sim \mathbb{G}^{p,\delta}$ satisfies
$$\Pr[S \subseteq \mathrm{paths}(s'')|S' \subseteq \mathrm{paths}(s'')] \le 2^{\delta|S'|} \cdot [S \subseteq \mathrm{paths}(s)|S' \subseteq \mathrm{paths}(s)]$$
for all $S \in \Omega_{\mathrm{parts}}(N, S')$, i.e. conditioned on S' is a part of both s and s'', s'' is δ non-$\mathbb{G}$.

[75] Notation: When I say t is δ non-uniform, it is implied that t is sampled from a δ non-uniform distribution.

[76] The conditioning is in superscript because it is non-standard; standard would be $S \subseteq \mathrm{parts}(t)$ which is too long.

[77] The last term with $\alpha_k < \gamma$ is suppressed for clarity in this informal discussion.

We now define β-uniform as motivated above and using the previous definition, define δ non-β-uniform.

Definition 131 (β-uniform and δ non-β-uniform distributions—$\mathbb{F}^{|\beta}$ and $\mathbb{F}^{\delta|\beta}$)**.** Let $u \sim \mathbb{F}(N)$ be sampled from a uniform distribution over all permutations, $\Omega(N)$, of $\{0, 1 \dots N-1\}$ as in Notation 124. A permutation $s \sim \mathbb{F}^{|\beta}(N)$ sampled from a β-uniform distribution is $s = u|(S \subseteq \text{paths}(u))$ where[78] $\beta =: (S)$ and $S \in \Omega_{\text{parts}}(N)$.

A distribution $\mathbb{F}^{\delta|\beta}$ is δ non-β-uniform if it is δ non-$\mathbb{G}$ with $\mathbb{G}$ set to a β-uniform distribution (see Definition 130, above). Similarly, a distribution $\mathbb{F}^{p,\delta|\beta}$ is (p, δ) non-β-uniform if it is (p, δ) non-$\mathbb{G}$ with $\mathbb{G}$, again, set to a β-uniform distribution.

We now state the general version of Proposition 126.

Proposition 132 ($\mathbb{F}^{\delta'|\beta}|r' = \text{conv}(\mathbb{F}^{(p,\delta+\delta')|\beta})$)**.** *Let $t \sim \mathbb{F}^{\delta'|\beta}(N)$ be sampled from a δ' non-β-uniform distribution with $N = 2^n$. Fix any $\delta > 0$ and let $\gamma = 2^{-m}$ be some function of n. Let $s \sim \mathbb{F}^{\delta'|\beta}|r$, i.e. $s = t|(h(t) = r)$ and suppose $\Pr[h(t) = r] \geq \gamma$ where h is an arbitrary function and r some string in its range. Then s is "γ-close" to a convex combination of finitely many $(p, \delta + \delta')$ non-β-uniform distributions, i.e.*

$$s \equiv \sum_i \alpha_i s_i + \gamma' s'$$

where $s_i \sim \mathbb{F}_i^{p,\delta+\delta'|\beta}$ with $p = 2m/\delta$. The permutation s' may have an arbitrary distribution (over $\Omega(2^n)$) but $\gamma' \leq \gamma$.

The proof follows from minor modifications to that of Proposition 126 (see below).

C.2 Technical results for δ non-uniform distributions

Proof of Claim 129. To see this for S_1, we proceed as before and recall the lower bound $\Pr[S_1 \subseteq \text{paths}(t'_1)] > 2^{\delta|S_1|} \Pr[S_1 \subseteq \text{paths}(u)]$. The upper bound may be evaluated as

$$\begin{aligned}
\Pr[S_1 \subseteq \text{paths}(t'_1)] &= \Pr[S_1 \subseteq \text{paths}(t) | S \nsubseteq \text{paths}(t)] \\
&= \frac{\Pr[S_1 \subseteq \text{paths}(t) \wedge S \nsubseteq \text{paths}(t)]}{\Pr[S \nsubseteq \text{paths}(t)]} \\
&= \frac{\Pr[S_1 \subseteq \text{paths}(u) \wedge S \nsubseteq \text{paths}(u) \wedge g(u) = r']}{\Pr[S \nsubseteq \text{paths}(t)] \Pr[g(u) = r']} \\
&\leq \Pr[S_1 \subseteq \text{paths}(u)] \cdot \gamma^{-2}
\end{aligned}$$

where we used $\alpha'_1 = 1 - \Pr[S \subseteq \text{paths}(t)] = \Pr[S \nsubseteq \text{paths}(t)] \geq \gamma$, and $\Pr[g(u) = r'] \geq \gamma$. In the general case, suppose t'_is, t_is and S_is are as described in the proof of Proposition 126. Then, one would have

$$\begin{aligned}
\Pr[S_i \subseteq \text{paths}(t'_i)] &= \frac{\Pr[S_i \subseteq \text{paths}(u) \wedge S_{i-1} \nsubseteq \text{paths}(u) \wedge \dots S \nsubseteq \text{paths}(u) \wedge g(u) = r']}{\Pr[S_{i-1} \nsubseteq \text{paths}(t) \wedge \dots S \nsubseteq \text{paths}(t)] \Pr[g(u) = r']} \qquad (102) \\
&\leq \Pr[S_i \subseteq \text{paths}(u)] \cdot \gamma^{-2}
\end{aligned}$$

where $\alpha'_i = \Pr[S_{i-1} \nsubseteq \text{paths}(t) \wedge \dots S \nsubseteq \text{paths}(t)] > \gamma$ is assumed (else there is nothing to prove). □

Proposition (Proposition 59 restated with slightly different parameters)**.** *Let $t \sim \mathbb{F}^{\delta'|\beta}(N)$ be sampled from a δ' non-β-uniform distribution with $N = 2^n$. Fix any $\delta > \delta'$ and let $\gamma = 2^{-m}$ be some function of n. Let $s = t|(h(t) = r')$ and suppose $\Pr[h(t) = r'] \geq \gamma$ where h is an arbitrary function and r' some string in its range. Then s is "γ-close" to a convex combination of finitely many (p, δ) non-β-uniform distributions, i.e.*

$$s \equiv \sum_i \alpha_i s_i + \gamma' s'$$

where $s_i \sim \mathbb{F}_i^{p,\delta|\beta}$ with $p = 2m/(\delta - \delta')$. The permutation s' may have an arbitrary distribution (over $\Omega(2^n)$) but $\gamma' \leq \gamma$.

[78] As alluded to earlier, we define β to be a redundant-looking "one-tuple" (S) here but this is because later when we generalise to d-Shufflers, we set $\beta = (S, T)$ where T encodes paths not in u.

Proof. While redundant, we follow the proof of Proposition 126 adapting it to this general setting and omitting full details this time.

(For comparison: We replace t with s and u with b)

Step A: Lower bound on $\Pr[S \subseteq \text{paths}(s)]$.

Let $b \sim \mathbb{F}^{|\beta}(N)$. Suppose s is not δ non-β-uniform. Then consider the largest $S \in \Omega_{\text{parts}}(N)$ such that

$$\Pr[S \subseteq \text{paths}(s)] > 2^{\delta \cdot |S|} \cdot \Pr[S \subseteq \text{paths}(b)]. \tag{103}$$

Claim 133. Let S and s be as described. The random variable s conditioned on being consistent with the paths in $S \in \Omega_{\text{parts}}(N)$, i.e. $s_S := s|(S \subseteq \text{parts}(s))$, is δ non-β-uniformly distributed.

We give a proof by contradiction. Suppose s_S is "more than" δ non-β-uniform. Then there exist some $S' \in \Omega_{\text{parts}}(N, S)$ such that

$$\Pr[S' \subseteq \text{paths}(s) | S \subseteq \text{paths}(s)] > 2^{\delta \cdot |S'|} \Pr[S' \subseteq \text{paths}(b) | S \subseteq \text{paths}(b)].$$

Then

$$\begin{aligned}\Pr[S \cup S' \subseteq \text{paths}(s)] &= \Pr[S \subseteq \text{paths}(s)] \Pr[S' \subseteq \text{paths}(s) | S \subseteq \text{paths}(s)] \\ &> 2^{\delta \cdot |S \cup S'|} \cdot \Pr[S \cup S' \subseteq \text{paths}(b)]\end{aligned}$$

using Equation (103) and Equation (97). That's a contradiction to S being maximal.

Step B: Upper bound on $\Pr[S \subseteq \text{paths}(s)]$.

Claim 134. One has $|S| < m/(\delta - \delta')$.

To see this, observe that

$$\begin{aligned}\Pr[S \subseteq \text{paths}(s)] &= \Pr[S \subseteq \text{paths}(t) \wedge h(t) = r'] \cdot \Pr[h(t) = r'] \\ &\leq \Pr[S \subseteq \text{paths}(t)] \cdot \gamma^{-1} \\ &\leq 2^{\delta' |S|} \Pr[S \subseteq \text{paths}(b)] \cdot \gamma^{-1}\end{aligned}$$

and comparing this with the lower bound, one obtains $|S| < m/(\delta - \delta')$.

The remaining proof Proposition 126 similarly generalises by proceeding in the same vein. More concretely, suppose S_i, s_i, s_i' are defined analogously. Then the lower bound goes through almost unchanged while for the upper bound, the analogue of Equation (102) becomes

$$\begin{aligned}\Pr[S_i \subseteq \text{paths}(s_i')] &= \frac{\Pr[S_i \subseteq \text{paths}(s) \wedge S_{i-1} \nsubseteq \text{paths}(s) \wedge \ldots S \nsubseteq \text{paths}(s)]}{\Pr[S_{i-1} \nsubseteq \text{paths}(s) \wedge \ldots S \nsubseteq \text{paths}(s)]} \\ &\leq \Pr[S_i \subseteq \text{paths}(t) \wedge S_{i-1} \nsubseteq \text{paths}(t) \wedge \ldots S \nsubseteq \text{paths}(t) | h(t) = r'] \cdot \gamma^{-1} \\ &\leq \frac{\Pr[S_i \subseteq \text{paths}(t)]}{\Pr[h(t) = r']} \cdot \gamma^{-1} \leq 2^{\delta' |S_i|} \Pr[S_i \subseteq \text{paths}(b)] \cdot \gamma^{-2}.\end{aligned}$$

□