Constructive Post-Quantum Reductions

Nir Bitansky[*] Zvika Brakerski[†] Yael Tauman Kalai[‡]

Abstract

Is it possible to convert classical reductions into post-quantum ones? It is customary to argue that while this is problematic in the interactive setting, non-interactive reductions do carry over. However, when considering quantum auxiliary input, this conversion results in a *non-constructive* post-quantum reduction that requires duplicating the quantum auxiliary input, which is in general inefficient or even impossible. This violates the win-win premise of provable cryptography: an attack against a cryptographic primitive should lead to an algorithmic advantage.

We initiate the study of constructive quantum reductions and present positive and negative results for converting large classes of classical reductions to the post-quantum setting in a constructive manner. We show that any non-interactive non-adaptive reduction from assumptions with a polynomial solution space (such as decision assumptions) can be made post-quantum constructive. In contrast, assumptions with super-polynomial solution space (such as general search assumptions) cannot be generally converted.

Along the way, we make several additional contributions:

1. We put forth a framework for reductions (or general interaction) with *stateful* solvers for a computational problem, that may change their internal state between consecutive calls. We show that such solvers can still be utilized. This framework and our results are meaningful even in the classical setting.

2. A consequence of our negative result is that quantum auxiliary input that is useful against a problem with a super-polynomial solution space cannot be generically "restored" post-measurement. This shows that the novel rewinding technique of Chiesa et al. (FOCS 2021) is tight in the sense that it cannot be extended beyond a polynomial measurement space.

[*] Tel Aviv University, Israel, nbitansky@gmail.com. Member of the checkpoint institute of information security. Supported by ISF grants 18/484 and 19/2137, by Len Blavatnik and the Blavatnik Family Foundation, by the Blavatnik Interdisciplinary Cyber Research Center at Tel Aviv University, and by the European Union Horizon 2020 Research and Innovation Program via ERC Project REACT (Grant 756482).

[†] Weizmann Institute of Science, Israel, zvika.brakerski@weizmann.ac.il. Supported by the Israel Science Foundation (Grant No. 3426/21), and by the European Union Horizon 2020 Research and Innovation Program via ERC Project REACT (Grant 756482) and via Project PROMETHEUS (Grant 780701).

[‡] Microsoft Research and Massachusetts Institute of Technology, MA, USA, yael@microsoft.com. This material is based upon work supported by DARPA under Agreement No. HR001120200. 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.

Contents

1 Introduction

The notion of provable security in cryptography has had a great impact on the field and has become a defacto gold standard in evaluating the security of cryptographic primitives. A provably secure cryptographic primitive is stated in the form of a computational problem P, whose hardness is related by means of *reduction* to that of another problem Q which is either by itself considered intractable or in turn can be further reduced down the line. The reduction is an algorithm that solves the problem Q provided that it is given access to an algorithm that solves the problem P.

This gives rise to the "win-win principle" which stands as one of the main motivations for using provably secure cryptography. The logic is the following. Either an algorithmic solution for P cannot be found, i.e. the cryptographic primitive P is secure for all intents and purposes, or one can find an algorithmic solution for P which would imply an algorithmic solution for Q, thus contributing to the state of the art in algorithms design. Indeed, cryptographic reductions are the main working tool for the theoretical cryptographer. Numerous reductions between cryptographic primitives are known and hundreds of such reductions are published in the cryptographic literature every year.

The emergence of the quantum era in computing poses a new challenge to provable security and the win-win principle. Many existing reductions in the "pre-quantum" world implicitly or explicitly relied on the P-algorithm being classical. These reductions are thus a-priori invalid when considering quantum algorithms. A central line of investigation in the domain of post-quantum security is thus dedicated to the following question.

To what extent can pre-quantum reductions be ported to the post-quantum setting?

Such conversion may not always be possible. This is particularly a concern when considering *interactive* problems, i.e. ones where the solution to P involves multiple messages being exchanged with the solver algorithm. Indeed, one of the most prominent techniques for proving security in the interactive setting, namely the notion of *rewinding*, does not directly translate to the quantum setting and moreover one can explicitly show cases where pre-quantum reductions exist but post-quantum ones do not. In fact, this property was actually used to construct *proofs of computational quantumness* [BCM+18] in which a party proves that it is quantum by succeeding in a task for which there is a classical impossibility result (under computational assumptions). In a nutshell, the reason is that a quantum algorithm may keep a quantum state between rounds of interaction, and this quantum state is measured and thus potentially destroyed in order to produce the next message of interaction. It is therefore not possible to naively "rewind" the interaction back to a previous step as is customary in many classical proofs.

The focus of this work, therefore, is on *non-interactive cryptographic assumptions.* These are problems P whose syntax contains a (randomized) instance generator which generates some instance x, and a verifier that checks whether solutions y are valid (with respect to x or more generally the randomness that was used to generate x). The role of the solver algorithm in this case is simply to take x as input and produce a y that "verifies well" (we avoid getting into the exact formalism at this point).

Contrary to the interactive case, it is customary to postulate (often without proof) that classical reductions to non-interactive cryptographic assumptions carry over straightforwardly to the post-quantum setting since there is no rewinding. There is a simple challenge-response interface that on the face of it "does not care" whether the underlying P-solver is implemented classically or quantumly. This viewpoint, however, is overly simplistic, since the P solver may use *quantum auxiliary input*: a quantum state $|s\rangle$ that is used as a resource for solving P. The state $|s\rangle$ can be the result of some natural process upon which we have no control, or a result of some exhaustive preprocessing, or generated in the course of execution of some protocol. At any rate, the means to produce $|s\rangle$ are often not at our disposal, we just get a copy of the state.

In this case, similarly to the interactive setting, the quantum state is measured whenever the P-solver is called, and therefore, it potentially precludes us from calling the P solver more than once. This issue is often addressed in the literature by noticing that providing many copies of $|s\rangle$ would allow to call the P solver multiple times – namely there exists a quantum state $|s\rangle^{\otimes t}$ that allows to solve Q given access to the P solver. Therefore, the existence of a classical reduction still implies that if Q is intractable *even given arbitrary auxiliary input*, then the same holds for P.

We argue that the aforementioned common "solution" for post-quantum reductions in the presence of quantum auxiliary input is unsatisfactory. First and foremost, this solution strictly violates the win-win principle. While the argument above indeed implies that (some form of) intractability for P follows from (some form of) intractability for Q, it *does not* allow to convert an auxiliary-input algorithm for P into an auxiliary-input algorithm for Q in a constructive manner, since the transformation $|s\rangle \to |s\rangle^{\otimes t}$ is not an efficient one. An additional related concern is the *durability* of such reductions. Namely, that if we wish to execute the reduction more than once (i.e. solve multiple instances of Q) then we need to duplicate the state $|s\rangle$ an a-priori unbounded number of times.

Given this state of affairs, the question we are facing is the following.

To what extent can pre-quantum reductions to non-interactive assumptions be ported to the post-quantum setting constructively and durably?

Naturally, we do not wish to redo decades of cryptographic work in re-proving each result individually. Instead, we would like to identify the broadest class of pre-quantum reductions that can be generically converted into the post-quantum regime, and at the same time characterize the limitations where such generic conversion is not possible. This is the focus of this work, and indeed we show a generic transformation for a very broad class of reductions. Along the way we develop an adversarial model for *stateful* adversaries that may be of interest in its own right, even in the *classical* setting.

1.1 Our Main Results

We prove a general positive result for converting classical reductions into post-quantum ones. In particular we consider *non-adaptive reductions.* In such reductions, the set of queries to the oracle is determined before any query is made. It turns out that an important parameter in our positive as well as our negative result is the size of the *solution space* of the computational problem P ("the cryptographic primitive"). Our positive results apply to cases where the solution space is polynomial. One notable example the case where P is a "decision assumption", namely the P solver is a distinguisher that returns a single bit as output. Another notable example is the case where P is an NP search problem, with unique solutions (e.g., injective one-way functions or unique signatures). An informal result statement follows.

Theorem 1.1 (Positive result, informal)**.** *There exists an efficient transformation for converting any classical non-adaptive black-box reduction from assumption Q to assumption P, where P is a non-interactive assumption with a polynomial solution space, into a* constructive and durable *post-quantum reduction from Q to P.*

We prove a complementary negative result, for the case where P has a large solution space. The negative result relies on the existence of classical indistinguishability obfuscation which is secure against quantum adversaries.

Theorem 1.2 (Negative result, informal)**.** *Assume the existence of post-quantum secure indistinguishability obfuscation. Then there exist non-interactive assumptions P, Q, where P has a super-polynomial solution space and the following hold. There exists a classical non-adaptive black-box reduction from assumption Q to assumption P, but there is no such constructive post-quantum reduction.*

As explained above, in order to address the question of constructiveness, we need to develop a new adversarial model and a host of tools to address this question. An account of these intermediate contributions appears in the technical overview below.

1.2 Our Techniques and Additional Contributions

Known approaches fall short of achieving constructiveness and durability since they regard quantum auxiliary input similarly to its classical counterpart, despite the inherent difference of the inability to duplicate or reuse quantum information. We assert that the process of making multiple calls to an algorithm with quantum side information is inherently *stateful.* Namely, the internal state of the "oracle" changes and evolves over time. In this work we put forth a framework for stateful solvers, namely algorithms that change their internal state and thus their behavior over time.

In the post-quantum setting, reductions start from *one-shot* solvers. That is, ones that have an initial state that allows them to provide an answer for a single instance of P successfully, but afterwards all bets are off. It seems natural (and, as we show, turns out to be useful) to consider stateful solvers that propagate their P-solving property throughout an execution, we call this property *persistence.* Persistent solvers evolve their state in an arbitrary way subject to being able, at any point in their evolution, to successfully answer a P-query (with some noticeable advantage).

A Framework for Stateful Solvers. Section 3 is dedicated to formally defining the notion of a (potentially stateful) solver and quantifying its success probability in solving a problem P. We accordingly provide definitions for a post-quantum reduction in this setting, and more specifically the notion of a post-quantum black-box reduction. The standard notion of a classical black-box reduction is recovered as a special case of our definition, when specializing to so-called *stateless* P-solvers.

Using our new formalism, the task at hand is to convert a reduction that expects to be interacting with a *stateless* solver, into one that is successful even when given a *one-shot stateful* solver.

One-Shot Solvers Imply Persistent Solvers. One-shot solvers may seem quite useless, since on the face of it they may only successfully respond to a single query. However, our first technical result, in Section 4, is that they can in fact be converted generically (but in a non-black-box manner) into persistent solvers. Namely, ones that can answer an *a-priori unbounded* number of queries and maintain roughly the same success probability. The persistent solver has a state of length that is polynomially related to that of the one-shot solver. The running time of the persistent solver increases with each query it is being asked. That is, the time complexity of answering the t-th query scales with $\mathrm{poly}(t)$ for a fixed polynomial. This still ensures that for any polynomial-length sequence of queries, the total time to answer all queries is bounded by a fixed polynomial. The persistent value of the resulting solver (i.e. the value that is maintained for an a-priori unbounded number of times) is itself a random variable that is determined during the conversion process. The expectation of the persistent value is equal to the one-shot value of the solver we start from. (We note that it is inherently impossible to achieve a non-probabilistic behavior, i.e. to ensure a persistent value that is always above some threshold.[1])

Our transformation is an extension of the techniques in the recent work of Chiesa, Ma, Spooner and Zhandry [CMSZ21], that can be interpreted as showing such a transformation for "public-coin" cryptographic assumptions (ones where the instances are uniformly distributed and the verification requires only the instance and the solution, and not the randomness that was used to generate the instance). It is only in this step that we have the restriction that the solution space of the problem needs to be polynomial, due to limitations of the [CMSZ21] technique. Our negative result (further discussed below) proves that these limitations are inherent.

The conversion from one-shot to persistent is the only transformation that uses the solver in a non-black-box manner. In the rest of our (positive) results we take a persistent P-solver and a bound on the length of its auxiliary quantum state and only make black-box use of this solver, i.e., provide instances as input and receive solutions as output. We do not further intervene with the evolution of the state between consecutive calls to the solver.

Once we transformed our solver to being persistent, we are guaranteed that we can make multiple P queries, and each one will be answered by a "successful" solver. It may seem that our mission is complete. However, this is far from being the case. While all queries are answered by a successful solver, these solvers may be arbitrarily correlated. For example, thinking about a simple linearity test where a reduction queries

[1] To see this, consider the case where the one-shot auxiliary input $|s\rangle$ is a superposition giving weight $\sqrt{1-\varepsilon}$ to a value $|\bot\rangle$ that always makes the P-solver fail, and giving weight $\sqrt{\varepsilon}$ to a state that makes the P-solver perfectly successful. Then, by trace-distance considerations, any processing of $|s\rangle$ must be ε-statistically-indistinguishable from a case where $|s\rangle = |\bot\rangle$. Therefore, with probability at least $1-\varepsilon$ the persistent value will be trivial. Nevertheless, using a Markov argument, if we start from a one-shot solver with a non-negligible advantage, we recover, with a non-negligible probability, a many-shot solver with a non-negligible persistent advantage.

$x_1, x_2, x_3 = x_1 \oplus x_2$ and checks whether a linear relation holds. It may be the case that for each query x_i we get a response y_i from an approximately-linear function, and yet the solver "remembers" that x_1, x_2 were previously made as queries, and deliberately fails on $x_1 \oplus x_2$ in the next query. Another example, that will be quite useful to illustrate our transformation is that of the Goldreich-Levin (GL) hardcore bit [GL89], where queries take the form $(f(x), r_i)$, always with the same $f(x)$, and with additional correlations between the r_i values across different queries. In particular, it may be the case that once a query with some value $f(x)$ has been made, the solver refuses to meaningfully answer any additional queries with the same $f(x)$.[2]

We note that attributing adversarial behavior to the solver is done for purposes of analysis. Our transformation from one-shot to persistent appears quite "innocent" and we do not know whether it can actually generate such pathological behavior that will prevent reductions from running. However, we cannot rule it out and therefore we consider a worst-case adversarial model.

When described in this way, it seems that only very specialized reductions can be carried over to the post quantum setting. For example, ones that employ a strong form of random self reduction when making solver queries. One such case is the search-to-decision reduction for the learning with errors problem [Reg05]. However, as the GL hardcore bit example demonstrates, this doesn't even extend to all search to decision reductions. We must therefore find a new way to utilize stateful solvers. Indeed, the handle that we use is that while the solver may change its behavior adversarially, its adversarial behavior is constrained by the length of the auxiliary state $|s\rangle$ that it uses. We will indeed leverage the fact that this state is polynomailly bounded to limit the adversarial powers of the solver and handle more general reductions.

Before moving on to describe our techniques in this context, we notice that while this adversarial model (of black-box access to a persistent solver) emerged as a by-product of our work on quantum reductions, it is nevertheless a valid model in its own right in both the quantum and classical setting. We may consider interacting with an adversary/solver that is *only* guaranteed to be noticeably successful at every point in time but, unlike the standard notion of an "oracle", may change its behavior over time. In our case, we allow the behavior to change arbitrarily, so long that the amount of information carried over between executions is bounded (in our case, by the length of the state, which is polynomially bounded).

Memoryless Persistent Solvers. Our next step, in Section 5, is to show that a persistent solver, even with adversarial behavior, can be effectively converted into a more predictable form of solver that we call *memoryless* (note that this is different from our final goal which is to achieve a *stateless* solver). A memoryless solver keeps track of the sequence number of the question it is asked (e.g. it knows that it is now answering query number 4) but it is not allowed to remember any information about the actual content of the previous queries that were made.

We show that a combination of a non-adaptive reduction and a persistent solver induce a memoryless (persistent) solver (more accurately a distribution over memoryless solvers). These memoryless solvers are accessible using a simulator that, given access to the reduction and the original solver, efficiently simulates the interaction of the reduction with the induced memoryless solver, up to inverse-polynomial statistical distance. Note that we require that the reduction is non-adaptive. Namely, its queries to the solver can be arbitrarily correlated (as in the GL case), but the identity of the queries must not depend on the answers to previous queries.

The transformation relies on the fact that the solver has a bounded amount of memory, say ℓ qubit of state that is propagated through the execution. Our strategy is to dazzle the solver with an abundance of i.i.d dummy queries, that are sampled from the marginal distribution of the "real" queries (for example, in the GL case, each dummy query will have the form $(f(x_i), r_i)$ where x_i, r_i are both random). In between the dummy queries, in random locations, we plant our real queries, in random order. We prove that the solver, having only ℓ qubits of state, must answer our real queries as if they were dummy queries. This requires us to develop a proper formalism and to prove a new lemma (Plug-In Lemma) using tools from quantum information theory. See Section 9 for the full details.

Stateless Solvers at Last. Finally, we show in Section 6 that memoryless solvers imply stateless solvers.

[2] We note that while the *classical* GL reduction, falls under our umbrella of non-adaptive reductions, in this specific case, it is in fact known how to devise a single-query *quantum* reduction [AC02]. This, however, does not resolve the question of durability, and more importantly does not provide a general framework for all non-adaptive reductions.

This is again shown by means of simulation via a similar formalism to the previous result. Recall that a stateless solver must answer all queries according to the same distribution. This transformation again relies on the non-adaptive nature of the reduction, namely on the ability to generate all solver-queries ahead of time. To do this, we notice that we can think of a memoryless solver simply as a sequence of stateless solvers that can be queried one at a time. Therefore, we can consider the induced stateless solver that at every query picks a random solver from this collection and executes it on the query. This indeed will result in a stateless solver. The solving probability of the induced stateless solver is simply the average success probability of solvers in the collection, which is concentrated due to persistence. Moreover, this behavior can be simulated by randomly permuting the *queries*, while still calling the solvers according to their order in the sequence.[3]

This way, asking the queries in a permuted order to the memoryless solver will (almost) mimic the action of sampling a solver from the collection independently for each query. The only reason why this mimic is not perfect is that permuted queries are sampling "without repetition", i.e. none of the solvers in the sequence defined by the memoryless solver will be queried twice, whereas in the ideal strategy we described above, it is possible that the same solver from the sequence will be sampled more than once. We deal with this by making the number of solvers in the sequence so big, that the probability of hitting the same solver twice becomes very small (inversely polynomial for a polynomial of our choice). We simply add to our queries of interest a large number of dummy "0 queries", and perform a random permutation on this extended set of queries.

Putting Things Together. In Section 7 we put all of the components together and prove our main positive result, that any classical non-adaptive reduction which relies on a non-interactive polynomial-solution-space assumption can be made post quantum. This requires putting together the components in a careful manner.

The fact that the first step in our transformation was to produce a persistent P, allows us to continue using it even after having solved a Q instance. In particular, this means that we can solve additional instances of Q, or use it to solve additional instances of P or any other problem Q' for which a non-adaptive reduction to P exists. In particular, this property implies that our reduction is *durable*.

A Negative Result for Search Assumptions. We show in Section 8, that a generic conversion from classical to constructive quantum reductions is not always possible, even for the case of non-adaptive reductions to non-interactive assumptions. In particular, if P is an assumption with a large solution space (intuitively, a search assumption) this may not be possible.

We show our negative result by relying on a recently introduced primitive known as tokenized signatures [BS16]. These are signature schemes with the standard classical syntax, but for which it is possible to produce a quantum *signature token*. The signature token allows to generate a single classical signature for a message of the signer's choice, but only one such signature can be created. Tokenized signatures have been constructed relative to a classical oracle [BS16] or based on cryptographic assumptions [CLLZ21].

We can define an "assumption" which is essentially the task of signing a random message using a tokenized signature scheme.[4] In the classical world, there is a trivial reduction between the task of signing one random message and the task of signing two random messages. However, if we consider a quantum solver that holds the token as auxiliary input, then by definition it should not be possible to use it to obtain two signatures for two different messages. Our negative result holds for any conversion process that is constructive, and in particular does not obtain any implicit non-uniform advice about the assumption.

1.3 Other Related Work

The question of which reductions can be translated from the classical to the post-quantum setting also received significant attention in the context of the random-oracle model (ROM), starting from the work of Boneh et al. [BDF+11]. The question asked in these works is whether it is possible to convert reductions in the classical ROM into ones the quantum ROM (QROM, where the adversary is allowed to make quantum queries to the oracle). There are several results proving that specific schemes that are secure in the ROM are

[3] Remember that we have access to the memoryless solver which only allows to make queries in order.

[4] The assumption is instantiated by a verification key which we can think of as non-uniformity of the assumption, see discussion in Section 8.

also secure in the QROM [Zha12, TU16, JZC$^+$18, KYY18, Zha19, DFMS19, LZ19, DFM20, KS20]. Recently, a more general "lifting theorem" was given in [YZ21], showing how to convert a proof in the ROM to one in the QROM for any "search-type game" where a challenger makes only a *constant* number of queries to the random oracle. This work also presented a negative result, showing that there are schemes that are secure in the ROM yet are insecure in the QROM. While the general motivation in these works is similar to ours, the question they ask is quite different from ours. In the ROM/QROM, the solver is allowed to make queries to the oracle (which is simulated by the reduction), which is more similar to the setting where interactive-assumptions are used.

Our memoryless transformation (Section 5) relies heavily on the state of the solver being bounded in length. The idea that bounded quantum memory can be used to restrict an otherwise all powerful adversary is at the core of the bounded quantum storage model. It can be shown (see, e.g., [DFSS08]) that it is possible to achieve cryptographic abilities against strong adversaries while relying only on a limit on the amount of quantum storage they can use. This setting is quite different from ours, though, since the quantum bounded storage model allows an unbounded amount of classical memory, which in our setting would make it impossible to achieve any result. Indeed, the bounded storage model requires quantum communication (whereas our reduction-solver communication is completely classical), and thus the set of tools and techniques that are used in both settings are completely different.

2 Preliminaries and Tools

We say that a given function $f(x_1, \ldots, x_k)$ is $\text{poly}(x_1, \ldots, x_k)$, if there exist constants c, C such that $(x_1 \cdot x_2 \cdot \ldots \cdot x_k)^c \leq f \leq (x_1 \cdot x_2 \cdot \ldots \cdot x_k)^C$.

We denote by TD the trace distance between two matrices.

Algorithms. By default, when referring to an *algorithm* we mean a classical probabilistic (resp. quantum) algorithm. Algorithms may be uniform or non-uniform, meaning that they have *classical* advice related to the input size (we specify when uniformity matters). An *efficient* algorithm is also polynomial time.

Quantum Notation. We use standard quantum information in Dirac notation. We denote quantum variables in boldface $\boldsymbol{x}$ and classical variables in lowercase x. The density matrix of $\boldsymbol{x}$ is denoted $\rho_{\boldsymbol{x}}$. Classical variables may also have (diagonal) density matrices. Quantum variables $\boldsymbol{x}, \boldsymbol{y}$ have a joint density matrix $\rho_{\boldsymbol{x},\boldsymbol{y}}$ if they can be jointly produced by an experiment. As usual, $\boldsymbol{x}, \boldsymbol{y}$ are independent if $\rho_{\boldsymbol{x},\boldsymbol{y}} = \rho_{\boldsymbol{x}} \otimes \rho_{\boldsymbol{y}}$. We never assume that quantum variables are independent unless we explicitly say so. Quantum registers are denoted in capital letters. We also sometimes use capital letters to denote distributions, where it is clear from the context. For a finite Hilbert space $\mathcal{H}$ we denote by $\mathbf{S}(\mathcal{H})$ the set of density matrices over quantum states in $\mathcal{H}$.

A quantum procedure is a general quantum algorithm that can apply unitaries, append ancilla registers in 0 state, perform measurements in the computational basis and trace out registers. The complexity of F is the number of *local* operations it performs (say, operations on up to 3 qubits are considered local). If F is a quantum procedure then we denote by $F(\boldsymbol{x})$ the application of F on $\boldsymbol{x}$. Any unitary induces a quantum procedure that implements this unitary, which does not perform measurements or trace out registers, we call this procedure "a unitary quantum circuit".

Purification of Quantum Procedures and States. A quantum procedure may introduce new ancilla qubits, perform intermediate measurement throughout its computation and discard registers or parts thereof. However, any quantum procedure can be *purified* into unitary form without much loss in complexity [NC16]. This is formally stated below.

Proposition 2.1. *Let C be a general quantum procedure of complexity s. Then it is possible to efficiently generate a* unitary *quantum circuit $\widehat{C}$ of size $O(s)$, such that for any quantum state $(\boldsymbol{x}, \boldsymbol{a})$, setting $(\boldsymbol{y}, \boldsymbol{z}) = \widehat{C}(\boldsymbol{x}, \mathbf{0})$, it holds that $(\boldsymbol{y}, \boldsymbol{a})$ has identical density matrix to $(C(\boldsymbol{x}), \boldsymbol{a})$.*

Likewise, any quantum state can be viewed as a reduced density matrix of the output of a unitary (which may be inefficient to implement) .

Proposition 2.2. *Let $\boldsymbol{x}$ be a variable with density matrix $\rho_{\boldsymbol{x}}$. Then there exists a unitary U over registers XY such that applying $U(\mathbf{0},\mathbf{0})$, the reduced density matrix of the value in the X register has density matrix $\rho_{\boldsymbol{x}}$.*

2.1 The Plug-In Lemma

The following lemma is another manifestation of information incompressibility in the quantum setting. Specifically, we are interested in an experiment in which an all powerful compressing procedure attempts to compress t samples which are arbitrarily distributed into ℓ quantum bits. We show that this is infeasible even in the weak sense in which a decoder receives the compressed value, and a $(j-1)$-prefix of the sequence, and is required to identify the j-th element. We show that as t increases, the probability of succeeding in the experiment drops. A formal statement follows.

Lemma 2.3 (Plug-In Lemma)**.** *Let $\vec{Y} = (Y_1, \ldots, Y_t)$ be a vector of arbitrarily jointly distributed classical random variables. Let $\vec{y}$ be distributed according to $\vec{Y}$. Let $\boldsymbol{s}$ be an ℓ-qubit random variable that has arbitrary dependence on $\vec{y}$. We let $\vec{y}_i$ denote the prefix $\vec{y}_i = (y_1, \ldots, y_i)$ for $1 \le i \le t$, and $\vec{y}_0$ is the empty vector (and likewise for $\vec{Y}$). Let J be the uniform distribution over $[t]$ and let $j \leftarrow J$. Define $y' \leftarrow Y_J|(\vec{Y}_{j-1} = \vec{y}_{j-1})$. Then it holds that*

$$\mathrm{TD}((j, \vec{y}_{j-1}, y_j, \boldsymbol{s}), (j, \vec{y}_{j-1}, y', \boldsymbol{s})) \le \sqrt{\ell/(2t)}\ . \tag{1}$$

Note that the above two distributions are *not* identical even though $(j, \vec{y}_{j-1}, y_j)$ and $(j, \vec{y}_{j-1}, y')$ are identically distributed. The reason is that in both cases, $\boldsymbol{s}$ is always generated as a function of $\vec{y}$, i.e. using y_j and not y'_j.

The lemma is proven in Section 9.

3 Assumptions, Stateful Solvers, and Reductions

In this section, we formally define the concepts of non-interactive cryptographic assumptions, stateful solvers, and their value and advantage in breaking an assumption.

3.1 Non-Interactive Assumptions

We define the notion of a non-interactive (falsifiable) cryptographic assumption as in [Nao03, HH09]. While we frame the notion as "cryptographic", it can be viewed more generally as a notion for average-case problems where the solution can be verified.

Definition 3.1 (Non-Interactive Assumption)**.** *A non-interactive assumption is associated with polynomials $d(\lambda), n(\lambda), m(\lambda)$ and a tuple $P = (G, V, c)$ with the following syntax. The* generator *G takes as input 1^λ and $r \in \{0,1\}^d$, it returns $x \in \{0,1\}^n$. The* verifier *V takes as input 1^λ and $(r, y) \in \{0,1\}^d \times \{0,1\}^m$ and returns a single bit output. (Both G and V are deterministic.) $c(\lambda)$ is the assumption's* threshold.

We say that P is falsifiable *if G, V are uniform polynomial-time algorithms (in their input size).*

We also define a property called *verifiably-polynomial image* that roughly speaking requires that any instance has at most polynomial many solutions and that this can be verified in some weak sense. The property in particular captures problems where the solution space $\{0,1\}^m$ is of polynomial size such as decision problems (where $m = 1$), and problems in **NP** where there are a few solutions per instance (such as injective one-way functions).

Definition 3.2 (Verifiably-Polynomial Image)**.** *A non-interactive assumption P has a* verifiably-polynomial image *if there exists an efficient verifier K and a polynomial $k = \mathrm{poly}(\lambda)$, such that for any instance $x \in \{0,1\}^n$, the set $Y_x := \{y : K(1^\lambda, x, y) = 1\}$ of K-valid solutions is of size at most k and for any valid instance $x = G(1^\lambda, r)$ and solution y such that $V(1^\lambda, r, y) = 1$, it holds that $y \in Y_x$.*

The traditional notion of the advantage in solving an assumption P is measured in terms of the distance between the solving probability (which we term the value) and the threshold c.

Definition 3.3 (Value and Advantage of Classical Functions)**.** *Let $P = (G, V, C)$ be a non-interactive assumption and let $f = \{ f_\lambda : \{0,1\}^n \to \{0,1\}^m \}_\lambda$ be a family of (possibly randomized) functions. For every $\lambda \in \mathbb{N}$, we define the corresponding value and advantage:*

$$\mathsf{val}_P[f](\lambda) := \Pr\left[V(1^\lambda, r, y) = 1 \;\middle|\; \begin{array}{c} r \leftarrow \{0,1\}^d \\ x = G(1^\lambda, r) \\ y \leftarrow f_\lambda(x) \end{array} \right] \qquad \mathsf{a}_P[f](\lambda) := |\mathsf{val}_P[f](\lambda) - c(\lambda)| \;\;,$$

where the probability is also above the randomness of f_λ in case it is randomized.

3.2 Stateful Solvers

The premise of our work is that in the quantum setting, one ought to think about *stateful* solvers, which generalizes the standard treatment of a solver as a one-shot algorithm. We now define this formally.

Definition 3.4 (Stateful Solvers: Syntax)**.** *Let P be a non-interactive assumption.*

*Let $\ell = \ell(\lambda)$ be a function. A classical (resp. quantum) ℓ-*stateful solver *$\mathcal{B} = (B, \mathsf{state}_0 = \{\mathsf{state}_{\lambda,0}\}_\lambda)$ is defined as follows.*

- *B is a classical (resp. quantum) algorithm that takes as input 1^λ, 1^t, $x \in \{0,1\}^n$ and state which is an ℓ-bit (resp. qubit) string, and outputs a value $y \in \{0,1\}^m$ and state' which is an ℓ-bit (resp. qubit) next-state. We let $B(\cdots)_{\mathrm{y}}$ denote the y output and $B(\cdots)_{\mathrm{st}}$ denote the state' output.*

- *$\mathsf{state}_0 = \{\mathsf{state}_{\lambda,0}\}_\lambda$ is a sequence of classical (resp. quantum) states consisting of $\ell = \ell(\lambda)$ bits (resp. qubits).*

We say that $\mathcal{B}$ is efficient if B runs in time $\mathrm{poly}(\lambda, t, n)$*; i.e., in polynomial time in the lengths of its inputs.*

Remark 3.1 (Non-uniformity). The algorithm B may have a non-uniform classical advice. It does not have any additional quantum advice.

Remark 3.2 (Dependence on Runtime). Our definition allows the running time of efficient stateful solvers to depend polynomially on the "iteration" t. In particular, for any polynomial number of solving attempts $t = \mathrm{poly}(\lambda)$, the overall running of the solver is polynomial. One could also consider a more stringent definition that requires that each call runs in fixed polynomial time independently of the iteration number t. Jumping forward, we will show how a solver can preserve its solving ability through time, but at the cost of running for longer in each step. Doing this according to the more stringent time-independent definition remains an open question.

It will be useful to define some properties of solvers with respect to an extension of the sovler's execution transcript. The extension corresponds to the would-be transcript of a purified version of the solver, running on a purified version of the initial state. This will allow us to get a precise well-defined handle on the evolution of quantum states throughout the lifetime of the solver. The extended transcript will only be used for purposes of definition and analysis and will never be required algorithmically.

Definition 3.5 (Stateful Solvers: Purifying Values)**.** *Consider a solver $\mathcal{B} = (B, \mathsf{state}_0 = \{\mathsf{state}_{\lambda,0}\}_\lambda)$. Let $B_{\lambda,t,x}$ denote the quantum procedure that takes $\boldsymbol{s}$ as input and produces $B(1^\lambda, 1^t, x, \boldsymbol{s})$ over registers SY. By Proposition 2.1, we can consider its purification $\widehat{B}_{\lambda,t,x}$ which acts on registers $SY\hat{Y}$ and takes as input $(\boldsymbol{s}, \mathbf{0}, \mathbf{0})$. Then define $\widehat{B}(1^\lambda, 1^t, x, \boldsymbol{s})$ as the algorithm that computes $(\boldsymbol{s}', \boldsymbol{y}, \hat{\boldsymbol{y}}) = \widehat{B}_{\lambda,t,x}(\boldsymbol{s}, \mathbf{0}, \mathbf{0})$, measures $(\boldsymbol{y}, \hat{\boldsymbol{y}})$ in the computational basis to obtain $(y, \hat{y})$, and then outputs $\boldsymbol{s}'$ as the state output, y as the solution output, and $\hat{y}$ as the* purifying output.

In addition, by Proposition 2.2, there exists a (possibly inefficient) unitary $\widehat{B}_{0,\lambda}$ that operates on two registers $S\hat{Y}$ such that when applying $(\boldsymbol{s}_0, \hat{\boldsymbol{y}}_0) \leftarrow \widehat{B}_{0,\lambda}(\mathbf{0}, \mathbf{0})$, the reduced density matrix of $\boldsymbol{s}_0$ is identical to *that of* state_0*. Then define* $\widehat{B}_0(1^\lambda)$ *as the quantum procedure that computes* $(\boldsymbol{s}_0, \hat{\boldsymbol{y}}_0) \leftarrow \widehat{B}_{0,\lambda}(\mathbf{0}, \mathbf{0})$*, measures* $\hat{\boldsymbol{y}}_\mathbf{0}$ *in the computational basis, and then outputs* $\boldsymbol{s}_0$ *as* state_0 *and* $\hat{y}_0$ *as the* purifying initial value.

We refer to the collection $\widehat{\mathcal{B}} = \{\, \widehat{B}_{i,\lambda,x} \,\}$ *as* a purification of $\mathcal{B}$ *(it is not unique).*

Remark 3.3. We note that the purifying values can be arbitrarily long. These values will only be used for analysis purposes and are never produced in an actual execution, and hence we do not require any bound whatsoever on the length of the purifying values or the complexity of producing them.

We now define the concept of a *solver interaction,* which captures the process of repeatedly invoking a stateful solver by a given algorithm.

Definition 3.6 (Solver Interaction)**.** *Let* $P = (G, V, c)$ *be a non-interactive assumption. For any stateful solver* $\mathcal{B} = (B, \mathsf{state}_0)$ *and corresponding purification* $\widehat{\mathcal{B}}$*, and any algorithm* A *with input* $z \in \{0,1\}^*$*, we consider the process* $A^{\mathcal{B}}(1^\lambda, z)$ *of the algorithm interacting with the solver. We define this process in two different yet equivalent manners: one which is efficient given the ability to execute* B*, and one which may be inefficient but implies an identical output distribution. The latter will include a production of all purifying values (Definition 3.5) which will be useful for definitions and analysis.*

- *We let* state_0 *be as defined in* $\mathcal{B}$*.*
  Equivalently: We let $(\mathsf{state}_0, \hat{y}) \leftarrow \widehat{B}_0(1^\lambda)$*.*
- A *is invoked on input* $(1^\lambda, z)$ *and at every step* $i \geq 1$*:*
  1. A *submits a query* $x_i \in \{0,1\}^n$*.*
  2. $(y_i, \mathsf{state}_i) \leftarrow B(1^\lambda, 1^i, x_i, \mathsf{state}_{i-1})$ *is invoked.*
     Equivalently: $(\hat{y}_i, y_i, \mathsf{state}_i) \leftarrow \widehat{B}(1^\lambda, 1^i, x_i, \mathsf{state}_{i-1})$ *is invoked.*
  3. A *obtains* y_i*, and proceeds to the next step.*
- *At the end of the interaction* A *may produce an output* w*.*

We sometimes refer to A *as a* solver-aided *algorithm and use the shorthand* $A_z^{\mathcal{B}}$ *for the solver interaction and* $A_z^{\widehat{\mathcal{B}}}$ *for the purified solver interaction. We refer to the random variables* $\mathsf{state}_0, \mathsf{state}_1, \mathsf{state}_2, \ldots$ *as the state random variables of the interaction. We refer to the list of pairs of generated instances and solutions* (x_i, y_i) *as the transcript of the interaction and denote it by* ts*. We also define the* extended transcript $\widehat{\mathsf{ts}}$ *of the execution as consisting of the value* $\hat{y}_0$ *followed by a list to triples* $(x_i, y_i, \hat{y}_i)$*. Given an extended transcript* $\widehat{\mathsf{ts}}$*, we can produce the standard transcript* ts *by removing all purifying values. We call this action* redaction *and say that* ts *is the redacted transcript induced by* $\widehat{\mathsf{ts}}$*. Generating an extended transcript according to the purified solver interaction* $A_z^{\widehat{\mathcal{B}}}$ *and redacting it produces an identical distribution to the generation of the redacted transcript by direct interaction* $A_z^{\mathcal{B}}$*. The length of a transcript/extended-transcript is the number of pairs/triples it contains (this means that an extended transcript of length* 0 *is not empty since it still contains* $\hat{y}_0$ *. The* i*-prefix of a transcript/extended-transcript is denoted* $\mathsf{ts}_i/\widehat{\mathsf{ts}}_i$ *and contains the first* i *pairs/triples (and also* $\hat{y}_0$ *in the extended case).*

We show that the purifying values indeed purify the entire solver interaction, in the sense that they determine all states state_i as pure states for any solver interaction.

Proposition 3.7. *Let* $\mathcal{B} = (B, \mathsf{state}_0)$ *be a solver with purification* $\widehat{\mathcal{B}}$ *and consider the extended transcript* $\widehat{\mathsf{ts}}$ *of the solver interaction* $A_z^{\widehat{\mathcal{B}}}$ *and let* t *be its length. Then for all* $i \leq t$*, the state* state_i *is pure conditioned on* $\widehat{\mathsf{ts}}_i$*. Specifically, it has density matrix* $|s_{\widehat{\mathsf{ts}}_i}\rangle\langle s_{\widehat{\mathsf{ts}}_i}|$ *that is completely determined by* $\widehat{\mathsf{ts}}_i$ *(and therefore by the classical string* $\widehat{\mathsf{ts}}$*) and does not depend on any other parameter of the execution.*

Proof. We consider the purifying description of the solver interaction $A_z^{\widehat{\mathcal{B}}}$ and prove by induction. For $t = 0$, we recall that the pair $(\mathsf{state}_0, \hat{y})$ is generated by applying $\widehat{B}_{0,\lambda}$ on the zero state, followed by measuring the

$\hat{Y}$ register. The pre-measurement state over registers $S\hat{Y}$ is therefore pure, and can always be written as

$$\sum_{\hat{y}} \alpha_{\hat{y}} \left|s_{\hat{y}}\right\rangle_S \otimes \left|\hat{y}\right\rangle_{\hat{Y}} \ , \tag{2}$$

where $\alpha_{\hat{y}}$ are non-negative real values with $\sum_{\hat{y}} \alpha_{\hat{y}}^2 = 1$, and $\left|s_{\hat{y}}\right\rangle$ are fully specified unit vectors. Therefore, post-selecting on having measured the value $\hat{y}_0$ in register $\hat{Y}$, we have that the state in register S is exactly $\mathsf{state}_0 = \left|s_{\hat{y}_0}\right\rangle \left\langle s_{\hat{y}_0}\right|$, which completes the base step of the proof.

Now assume that the above holds for all $i < t$. Consider a transcript $\widehat{\mathsf{ts}}$ of length t s.t. $\widehat{\mathsf{ts}} = \widehat{\mathsf{ts}}_{t-1}\|(x, y, \hat{y})$ for some $\widehat{\mathsf{ts}}_{t-1}, x, y, \hat{y}$.

Let us consider the state of the system right before the t-th query to the solver. At this point, $\widehat{\mathsf{ts}}_{t-1}$ was already determined, and thus by induction we know that $\mathsf{state}_{t-1} = \left|s_{\widehat{\mathsf{ts}}_{t-1}}\right\rangle \left\langle s_{\widehat{\mathsf{ts}}_{t-1}}\right|$ is a pure state. At this point x has also been determined.

By definition, state_t is produced by executing a unitary $\widehat{B}_{\lambda,t,x}$ (that acts on registers $SY\hat{Y}$) on $(\mathsf{state}_{t-1}, \mathbf{0}, \mathbf{0})$, which is pure by the induction hypothesis, and measuring the $Y\hat{Y}$ registers. The analysis here is similar to the base case. The pre-measurement state is pure (since it is induced by applying a unitary on a pure state) and thus can always be written as

$$\sum_{y,\hat{y}} \alpha_{y,\hat{y}} \left|s_{y,\hat{y}}\right\rangle_S \otimes \left|y, \hat{y}\right\rangle_{Y\hat{Y}} \ , \tag{3}$$

and as above α_y are non-negative real values with $\sum_y \alpha_y^2 = 1$, and $\left|s_{y,\hat{y}}\right\rangle$ are fully specified unit vectors. Post selecting on $y, \hat{y}$ leaves us with register S containing $\mathsf{state}_t = \left|s_{y,\hat{y}}\right\rangle \left\langle s_{y,\hat{y}}\right|$, which completes the proof. □

We are now ready to define the concepts of value and advantage of stateful solvers. Traditionally, when thinking about stateless solvers, we consider their *one shot value*, namely the probability that they solve the problem on a random instance. Since they are stateless this probability does not change over time. In the case of stateful solvers, this probability may change over time. Our definition of the *many shot values* aims to capture exactly this. For any solver interaction $A_z^{\mathcal{B}}$, the value at time t, captures the probability that the solver $\mathcal{B}$ successfully solves a random instance at this time, after a given t-round interaction with A_z. This value is, in fact, a random variable that depends on the history of the interaction. To make this precise, we consider any purification $\widehat{\mathcal{B}}$, and define these values as a function of the extended transcript.

Definition 3.8 (Stateful Solvers: Value and Advantage)**.** *Let P be a non-interactive assumption, $\mathcal{B} = (B, \mathsf{state}_0)$ be a corresponding stateful solver, $\widehat{\mathcal{B}}$ a corresponding purification, and A a solver-aided algorithm with input z. For every $\lambda, i \in \mathbb{N}$, let state_i be the i-th pure state random variable of the solver interaction $A_z^{\widehat{\mathcal{B}}}$ (determined by $\widehat{\mathsf{ts}}_i$). The corresponding value random variables are:*

$$\mathsf{val}_P[i, A_z^{\widehat{\mathcal{B}}}](\lambda) := \Pr\left[V(1^\lambda, r, y) = 1 \;\middle|\; \begin{array}{c} r \leftarrow \{0,1\}^d \\ x = G(1^\lambda, r) \\ (\hat{y}_{i+1}, y, \mathsf{state}_{i+1}) \leftarrow \widehat{B}(1^\lambda, 1^i, x, \mathsf{state}_i) \end{array}\right] \ ,$$

where the probability is over the choice of r and the measurement of $\hat{y}_{i+1}, y$.
The one-shot value of $\mathcal{B}$ is

$$\mathsf{val}_P[0, \mathcal{B}](\lambda) := \Pr\left[V(1^\lambda, r, y) = 1 \;\middle|\; \begin{array}{c} r \leftarrow \{0,1\}^d \\ x = G(1^\lambda, r) \\ (y, \mathsf{state}_1) \leftarrow B(1^\lambda, x, \mathsf{state}_0) \end{array}\right] \ ,$$

where the probability is over the choice of r, measurements of B, and (the possibly mixed) state_0. Note that this is in fact a number, independent of any A or the choice of purification $\widehat{\mathcal{B}}$.

The corresponding advantage random variables are:

$$\mathsf{a}_P[i, A_z^{\widehat{\mathcal{B}}}](\lambda) := \left|\mathsf{val}_P[i, A_z^{\widehat{\mathcal{B}}}](\lambda) - c(\lambda)\right| \qquad \mathsf{a}_P[0, \mathcal{B}](\lambda) := \left|\mathsf{val}_P[0, \mathcal{B}](\lambda) - c(\lambda)\right| \ .$$

For a distribution $\mathbb{B}$ on solvers $\{\mathcal{B}_\alpha\}_\alpha$, we define the one-shot value of the distribution as:

$$\mathsf{val}_P[0,\mathbb{B}](\lambda) = \mathbb{E}_{\alpha\leftarrow\mathbb{B}}[\mathsf{val}_P[0,\mathcal{B}_\alpha](\lambda)] \ .$$

The corresponding advantage is $\mathsf{a}_P[0,\mathbb{B}](\lambda) = |c(\lambda) - \mathsf{val}_P[0,\mathbb{B}](\lambda)|$.

As the solver's state evolves over time, its advantage in solving an assumption may reduce or disappear altogether. This is in particular relevant to the quantum setting, where when a solver is invoked its internal state is disturbed. Aiming to capture solvers that remain useful over time, we next define the notion of solvers with *persistent value,* namely, solvers whose value in solving a given assumption is preserved through time. We define it more generally for distributions over solvers; single solvers are a special case.

Definition 3.9 (Persistent Value)**.** *Let P be a non-interactive assumption. A distribution $\mathbb{B}$ on solvers $\{\mathcal{B}_\alpha\}_\alpha$ is η-persistent if there exist purifications $\{\widehat{\mathcal{B}}_\alpha\}_\alpha$ such that for any algorithm A with input z, with probability $1-\eta$ over the choice of solver $\alpha \leftarrow \mathbb{B}$ and over an extended transcript $\widehat{\mathsf{ts}}$ in the solver interaction process $A_z^{\widehat{\mathcal{B}}_\alpha}$, there exists a value p such that:*

$$\max_i \left|\mathsf{val}_P[i, A_z^{\widehat{\mathcal{B}}_\alpha}] - p\right| \leq \eta \ . \tag{4}$$

We call p a persistent value*. Given a random variable $p^*(\alpha) \subseteq [0,1]$, we say that a solver is (p^*,η)-persistent if the condition holds for $p^*(\alpha)$.*

We next define the notion of a persistent advantage. This aims to capture the case that solvers maintain a lower bound on their advantage through time.

Definition 3.10 (Persistent Advantage)**.** *Let P be a non-interactive assumption with threshold c. A distribution $\mathbb{B}$ on solvers $\{\mathcal{B}_\alpha\}_\alpha$ has ε-persistent advantage if there exist purifications $\{\widehat{\mathcal{B}}_\alpha\}_\alpha$ such that for any algorithm A with input z:*

$$\mathbb{E}\left[\min_i \mathsf{val}_P[i, A_z^{\widehat{\mathcal{B}}_\alpha}]\right] \geq c + \varepsilon \ , \tag{5}$$

where the expectation is over the choice of solver $\alpha \leftarrow \mathbb{B}$ and over an extended transcript $\widehat{\mathsf{ts}}$ in the solver interaction process $A_z^{\widehat{\mathcal{B}}_\alpha}$.

In the above, We require that the advantage has a consistent sign (for simplicity, positive). Intuitively, the reason we focus on persistence of the positive advantage $v_t - c$ at time t, rather than the absolute advantage $|v_t - c|$, is that if the sign of $v_t - c$ arbitrarily changes after each solver invocation, then the solver may not be as useful. (As a simple example, take a deterministic distinguisher and turn it into a stateful distinguisher that flips the output of the original distinguisher at random with each invocation, deeming it useless.) We note that η persistent solvers in particular preserve the sign of their advantage (up to η).

Memoryless and Stateless Solvers. A special case of the above definitions is that of *memoryless and stateless solvers.*

Definition 3.11. *A solver $\mathcal{B} = (B, \mathsf{state}_0)$ is* memoryless *if the size of its state is $\ell = 0$. The solver is* stateless *if in addition (to being memoryless), the algorithm B does not depend on 1^t (in functionality or runtime).*

Remark 3.4 (Persistent Value for Stateless and Memoryless Solvers)*.* Note that in the case of stateless solvers, successive invocations of the solver will always result in the same output distribution. Here the one-shot (and many-shot) advantage coincide with the standard notion of advantage for functions (Definition 3.3) and values are persistent (Definition 3.9). Accordingly, stateless solvers exactly capture the traditional notion of classical solvers, given by a randomized function.

Moreover, even for memoryless solvers, when considering the definition of persistent solvers the value $\mathsf{val}_P[i, A_z^{\widehat{\mathcal{B}}}]$ does not depend on A_z at all (only on i), and therefore it is a fixed number rather than a random variable. It follows that for (p,η)-persistent memoryless solvers, Eq. (5) holds with probability 1.

3.3 Reductions

We now define the notion of a reduction. A reduction is a way to prove a claim of the form "if there exists a successful solver for assumption P then there exists a successful solver for an assumption Q". We consider *constructive reductions* in the sense that they are an explicit uniform algorithm that takes as input a successful solver for P and efficiently solves the problem Q.

The default notion of a reduction in the literature is *one shot*. In pparticular, a given quantum P-solver is only assumed to have a meaningful one-shot advantage in solving P, and there is no a priori guarantee on its advantage in any many shot solving process, in particular there may not be any value persistence. Likewise, the produced solver for the assumption Q is only required to have a meaningful one-shot advantage. Below we define both the default notion of one-shot reductions as well as the stronger notion of *durable reductions* requiring that the resulting Q-solver also has persistent advantage, meaning that with noticeable probability, the reduction can go on solving for an arbitrary polynomial number of times.

Definition 3.12 (Reduction)**.** *A reduction from classically (resp. quantumly) solving a non-interactive assumption Q to classically (resp. quantumly) solving a non-interactive assumption P is an efficient classical (resp. quantum) uniform algorithm $\mathcal{R}$ with the following guarantee. For any solver $\mathcal{B}_P = (B_P, \mathsf{state}_0)$ for P with one-shot advantage ε and running time T, let $\mathsf{state}'_0 = (\mathsf{state}_0, B_P, 1^{1/\varepsilon}, 1^T)$. Then $\mathcal{B}_Q = (\mathcal{R}, \mathsf{state}'_0)$ is a solver for Q with one-shot advantage $\varepsilon' = \mathrm{poly}(\varepsilon, T^{-1}, \lambda^{-1})$ and running-time $\mathrm{poly}(T, \varepsilon^{-1}, \lambda)$. We say that the reduction is durable if $\mathcal{B}_Q$ has $\mathrm{poly}(\varepsilon, T^{-1}, \lambda^{-1})$-persistent advantage.*

We refer to a reduction from solving Q to classically (resp. quantumly) solving P as a classical-solver (resp. quantum-solver) reduction.

Remark 3.5 (Many Shot Reductions)*.* There could be several conceivable extensions of the above definition that also account for the *many-shot advantage*. One such natural extension is requiring that the reduction works only given a solver with a persistent value (as in Definition 3.9). Jumping ahead, in section 4, we show that under certain conditions, persistent solving can in fact be reduced to one-shot solving, even in the quantum setting.

Remark 3.6 (The Loss)*.* We allow for a (fixed) polynomial loss in the advantage and running time. One could naturally extend it to more general relations.

Classical Black-Box Reductions. In this work, we prove that several general classes of *classical reductions* that a priori are only guaranteed to work for classical solvers, can be enhanced *efficiently* to also work for quantum solvers. Our focus is on black-box reductions; that is, reductions that are oblivious of the representation and inner workings of the solver that they use (in contrast to the above Definition 3.12, where the reduction obtains the full description of the solver $\mathcal{B}_P$).

We next formally define such black box reductions, using the terminology we have already developed. Specifically, we capture the notion of a classical solver for a given problem P as a stateless (classical) solver.

Definition 3.13 (Classical Black-Box Reduction)**.** *A classical black-box reduction, from solving a non-interactive assumption Q to solving a non-interactive assumption P, is an efficient classical solver-aided uniform algorithm $\mathcal{R}$ with the following syntax and guarantee. $\mathcal{R}$ takes as input a security parameter 1^λ, parameter $1^{1/\varepsilon}$, and instance $x \in \{0,1\}^{n_Q}$ of Q. It interacts with a solver $\mathcal{B}$ for P (per Definition 3.6) and produces an output $y \in \{0,1\}^{m_Q}$. We require that for any distribution $\mathbb{B}$ over* **stateless classical** *solvers $\{\,\mathcal{B}_\alpha\,\}_\alpha$ such that $\mathbb{B}$ has advantage at least ε in solving P, the corresponding solver distribution $\mathbb{R}$ over solvers $\{\,\mathcal{R}^{\mathcal{B}_\alpha}(1^\lambda, 1^{1/\varepsilon}, \cdot)\,\}_\alpha$ has advantage at least $\mathrm{poly}(\varepsilon, \lambda^{-1})$ in solving Q. The advantage of $\mathcal{R}$ is positive if its value is always at least c_Q (above the assumption Q's threshold), regardless of any P-solver.*

We further say that the reduction $\mathcal{R}$ is non-adaptive *if $\mathcal{R}$ produces all of its oracle queries $x_1, \dots, x_k \in \{0,1\}^{n_P}$ to $\mathcal{B}$ in one shot, obtains all answers $y_1, \dots, y_k$, and then produces its output y.*

Remark 3.7*.* In our definition of solver interaction, a given solver $\mathcal{B}$ is only ever invoked for the instance size $n_P(\lambda)$. Accordingly, the above definition restricts attention to classical reductions that in order to solve problem Q for instance size $n_Q(\lambda)$ make queries to a P-solver on a specific related input size $n_p(\lambda)$. While this is not without loss of generality, it does capture natural reductions. (In fact, we are not aware of important reductions that do not adhere to this.)

Remark 3.8 (Deterministic Solver Reductions, Positive Advantage, and Repeated Queries). We consider classical reductions that ought to work when given a stateless solver from a distribution $\mathbb{B}$ over solvers $\{\mathcal{B}_\alpha\}$. (As a matter of fact in our model, even once a stateless solver $\mathcal{B}_\alpha$ is fixed, the process of answering any given query is randomized, but this can be modeled as sampling a deterministic stateless solver from another distribution $\mathbb{B}$ with the same advantage.) A weaker notion of classical reductions only requires that the reduction works for deterministic solvers. In the classical setting, this is typically not an issue, as long as the reduction has the power to fix the solver's randomness and repeatedly replace it as needed. Jumping forward, when considering quantum reductions, the randomness of a given solver may arise from the quantum nature of the solving process, and the reduction may not be able to control it. Accordingly, in our transformations from classical-solver reductions to quantum-solver reductions, we will naturally need the classical reduction we start from to also be able to deal with distributions over solvers.

We note that for typical assumptions Q such as search problems (with trivial threshold $c = 0$) or decision problems (with solution length $m = 1$ and trivial threshold $c = 1/2$), a classical reduction $\mathcal{R}$ from Q-solving to deterministic P-solving implies a classical reduction $\mathcal{R}$' from Q-solving to distributional P-solving. Here two subtleties should be addressed. The first issue that could prevent $\mathcal{R}$ from working for distributional P-solvers is that the sign of the advantage of $\mathcal{R}^{\mathcal{B}}$ as a Q-solver may depend on the randomness of $\mathcal{B}$ and may cancel out in expectation. For search assumptions Q, where $c = 0$, this cannot happen as any advantage is positive. For decision problems, this can be avoided by slightly augmenting $\mathcal{R}$ to make sure that the advantage is always positive using standard black-box techniques [BG11]. This incurs only a polynomial overhead in solving queries, or even just a single query, at the cost of quadratically decreasing the advantage. The second issue concerns the running time of the reduction. Specifically a reduction that works for deterministic oracles, excepts to get their advantage $1^{1/\varepsilon}$ as input, where ε is the P-solver's advantage. When executing such a reduction with a solver distribution, we are given $1^{1/\varepsilon}$, where ε is the average advantage. Nevertheless, we can run the original reduction with input $1^{2/\varepsilon}$. Note that the probability that that the advantage of a sampled oracle is at least $\varepsilon/2$ is at least $\varepsilon/2$, and since the reduction has positive advantage, we are overall guaranteed to maintain a noticeable advantage.

Following the above, for typical assumptions Q, we can in particular assume w.l.o.g positive advantage. For simplicity, we also assume throughout that classical reductions We do not repeat queries. This is w.l.o.g as given a deterministic oracle, the reduction can simply store previous answers and answer consistently by itself.

4 Persistent Solvers in the Quantum Setting

In this section, invoking state restoration techniques from [CMSZ21], we prove that any one-shot solver for an assumption P with a verifiably-polynomial image (in particular, decision problems) can be converted into a persistent solver for P.

Theorem 4.1 (Persistence Theorem). *Let P be a non-interactive falsifiable assumption with a verifiably-polynomial image. For any inverse polynomial function η, there exist efficient quantum algorithms S, R with the following syntax and guarantee. $S_B(\mathsf{state}_0)$ takes as input a quantum algorithm B and state state_0 and outputs a state state_0^* and a value $p^* \in [0,1]$. $R_B(1^\lambda, 1^i, x, \mathsf{state}_{i-1}^*)$ takes as input B, a security parameter 1^λ, step 1^i, input $x \in \{0,1\}^n$, and state state_{i-1}^* and outputs a solution $y \in \{0,1\}^m$ and state state_i^*.*

For any solver $\mathcal{B} = (B, \mathsf{state}_0)$ with one-shot value $p = \mathsf{val}_P[0, \mathcal{B}]$, considering the random variable $(\mathsf{state}_0^*, p^*) \leftarrow S_B(\mathsf{state}_0)$, it holds that:

1. $\mathbb{E}[p^*] = p$.
2. *$\mathcal{R}^* = (R_B, \mathsf{state}_0^*)$ sampled in this process is a distribution over efficient stateful solvers that is (p^*, η)-persistent.*

Remark 4.1. The efficiency of the algorithms S, R is also polynomial in the running time of B. We avoid passing explicitly the running time bound as input to simplify notation.

The proof is based on two lemmas from [CMSZ21], adapted to our notation and simplified for our needs.

Lemma 4.2 (Lemmas 4.9 and 4.10 in [CMSZ21], adapted)**.** *Let $\mathcal{H}$ be a Hilbert space. There exist efficient quantum algorithms:*

1. $(\boldsymbol{\rho}^*, p^*) \leftarrow \mathsf{ValEst}_{V,A}(\boldsymbol{\rho}, 1^{1/\varepsilon})$ *that given as input any verifier circuit* $V : \{0,1\}^d \times \{0,1\}^n \to \{0,1\}$*, any quantum circuit* A*, a quantum state* $\boldsymbol{\rho} \in \mathbf{S}(\mathcal{H})$ *and accuracy parameter* $1^{1/\varepsilon}$*, outputs a quantum state* $\boldsymbol{\rho}^* \in \mathbf{S}(\mathcal{H})$ *and value* $p^* \in [0,1]$*;*
2. $\boldsymbol{\sigma}^* \leftarrow \mathsf{Repair}_{V,A,\Pi}(\boldsymbol{\sigma}, y, p, 1^{1/\varepsilon}, 1^k)$ *that given* V*,* A*, and a projective measurement* $\Pi = (\Pi_y)_{y\in Y}$ *on* $\mathcal{H}$ *with outcomes* $Y = \{\, y_1, \ldots, y_k \,\}$*, and given a quantum state* $\boldsymbol{\sigma} \in \mathbf{S}(\mathcal{H})$*, outcome* $y \in Y$*, probability* $p \in [0,1]$*, and parameters* $1^{1/\varepsilon}$*,* 1^k*, outputs a quantum state* $\boldsymbol{\sigma}^* \in \mathbf{S}(\mathcal{H})$*;*

such that the following guarantees hold:

1. **Value Estimation:**

$$\mathbb{E}\left[p^* \;\middle|\; (\boldsymbol{\rho}^*, p^*) \leftarrow \mathsf{ValEst}_{V,A}(\boldsymbol{\rho}, 1^{1/\varepsilon})\right] = \Pr\left[V(r,y) = 1 \;\middle|\; \begin{array}{l} r \leftarrow \{0,1\}^d \\ y \leftarrow A(\boldsymbol{\rho}, r) \end{array}\right] .$$

2. **Estimation is Almost Projective:**

$$\textit{For } \varepsilon \geq \varepsilon' > 0, \qquad \Pr\left[|p^* - p^{**}| \geq \varepsilon \;\middle|\; \begin{array}{l} (\boldsymbol{\rho}^*, p^*) \leftarrow \mathsf{ValEst}_{V,A}(\boldsymbol{\rho}, 1^{1/\varepsilon}) \\ (\boldsymbol{\rho}^{**}, p^{**}) \leftarrow \mathsf{ValEst}_{V,A}(\boldsymbol{\rho}^*, 1^{1/\varepsilon'}) \end{array}\right] \leq \varepsilon \ .$$

3. **Repairing:**

$$\textit{For } \varepsilon > 0, \qquad \Pr\left[|p^* - p^{**}| \geq \varepsilon \;\middle|\; \begin{array}{l} (\boldsymbol{\rho}^*, p^*) \leftarrow \mathsf{ValEst}_{V,A}(\boldsymbol{\rho}, 1^{1/\varepsilon}) \\ (\boldsymbol{\sigma}, y) \leftarrow \Pi(\boldsymbol{\rho}^*) \\ \boldsymbol{\sigma^*} \leftarrow \mathsf{Repair}_{V,A,\Pi}(\boldsymbol{\sigma}, y, p^*, 1^{1/\varepsilon}, 1^k) \\ (\boldsymbol{\rho}^{**}, p^{**}) \leftarrow \mathsf{ValEst}_{V,A}(\boldsymbol{\sigma}^*, 1^{1/\varepsilon}) \end{array}\right] \leq \varepsilon \ .$$

Remark 4.2 (On the Restriction to Verifiably-Polynomial Image Assumptions)*.* We note that the reason that the Persistence Theorem 4.1 is restricted to assumptions P with a verifiably-polynomial image stems from the fact that the running time of the repairing procedure Repair given by Lemma 4.2 scales with the number of outcomes k of the corresponding projection Π. It is tempting to search for a better repairing procedure that does not scale with k, since (as we show) it would imply a stronger version of Theorem 4.1, without the restriction of a verifiably-polynomial image. However, as we show in Section 8 (Theorem 8.3), such a stronger version of Theorem 4.1 does not exist (provided that R, S use the assumption P as a black box,[5] which is indeed the case in our proof).

We now proceed to prove the Persistence Theorem 4.1 relying on Lemma 4.2.

Proof of Theorem 4.1. Throughout, fix the assumption P, with generator G, verifier V, and polynomial-image verifier K with corresponding polynomial image bound k. Also fix an inverse polynomial function η and the security parameter λ.

Notation and Conventions. In what follows, we consider a one-shot quantum solver $y \leftarrow B(1^\lambda, x, \mathsf{state})$ that operates on quantum states $\mathsf{state} \in \mathbf{S}(\mathcal{H})$. We assume that $\mathcal{H} = \mathcal{Z}\mathcal{Y}$, where for every $x \in \{0,1\}^n$, there is an efficiently computable unitary purification $\widehat{B}_{\lambda,x}$ on $\mathcal{H}$, describing the action of $B(1^\lambda, x, \mathsf{state})$ on state, where $\mathcal{Y}$ is the output register (this assumption regarding $\mathcal{H}$ is w.l.o.g by polynomially increasing the size of the state if needed). We also consider a *wrapper solver* $\widetilde{B}(1^\lambda, r, \mathsf{state}^*)$ that given $\mathsf{state}^* \in \mathbf{S}(\mathcal{H})$, computes $x = G(1^\lambda, r)$, applies $\widehat{B}_{\lambda,x}$, and outputs a measurement of $\mathcal{Y}$ in the computational basis.

[5] Or in some more general explicit fashion that we define in Section 8.

For any $x \in \{0,1\}^n$, we denote by $\Pi_x = (\Pi_{x,y})_{y \in Y_x \cup \perp}$ the projective measurement on $\mathcal{H}$ where:

$$Y_x := \{\, y \in \{0,1\}^m : K(1^\lambda, x, y) = 1 \,\} \quad ,$$

$$\Pi_{x,y} := \widehat{B}^\dagger_{\lambda,x}(I_{\mathcal{Z}} \otimes |y\rangle\langle y|_{\mathcal{Y}})\widehat{B}_{\lambda,x} \qquad \text{for } y \in Y_x \quad ,$$

$$\Pi_{x,\perp} := \sum_{y \in \{0,1\}^m \setminus Y_x} \widehat{B}^\dagger_{\lambda,x}(I_{\mathcal{Z}} \otimes |y\rangle\langle y|_{\mathcal{Y}})\widehat{B}_{\lambda,x} \quad .$$

We define the algorithms S and R. Throughout, the algorithms ValEst and Repair operate on corresponding states state^* in $\mathbf{S}(\mathcal{H})$.

$S_B(\mathsf{state}_0)$:

- Output $(\mathsf{state}^*_0, p^*_0) \leftarrow \mathsf{ValEst}_{V,\widetilde{B}}(\mathsf{state}_0, 1^{2/\eta})$.

$R_B(1^\lambda, 1^i, x_i, \mathsf{state}^*_{i-1})$:

- Let $\varepsilon_i = \eta/(i\pi)^2$.
- Apply $(\boldsymbol{\sigma}_{i-1}, p_{i-1}) \leftarrow \mathsf{ValEst}_{V,\widetilde{B}}(\mathsf{state}^*_{i-1}, 1^{1/\varepsilon_i})$.
- Apply $(\boldsymbol{\sigma}^*_i, y_i) \leftarrow \Pi_{x_i}(\boldsymbol{\sigma}_{i-1})$.
- Apply $\boldsymbol{\rho}_i \leftarrow \mathsf{Repair}_{V,\widetilde{B},\Pi_{x_i}}(\boldsymbol{\sigma}^*_i, y_i, p_{i-1}, 1^{1/\varepsilon_i}, 1^{k+1})$.
- Apply $(\mathsf{state}^*_i, p^*_i) \leftarrow \mathsf{ValEst}_{V,\widetilde{B}}(\rho_i, 1^{1/\varepsilon_i})$.
- Output $(y_i, \mathsf{state}^*_i)$

We now prove that the above algorithms satisfy the requirements of Theorem 4.1.

First note that if B has one-shot value p, then by the estimation guarantee of Lemma 4.2,

$$\begin{aligned}\mathbb{E}\left[p^*_0 \;\middle|\; (\mathsf{state}^*_0, p^*_0) \leftarrow \mathsf{ValEst}_{V,\widetilde{B}}(\mathsf{state}_0, 1^{2/\eta})\right] \\ = \Pr\left[V(r,y) = 1 \;\middle|\; \begin{array}{l} r \leftarrow \{0,1\}^d \\ y \leftarrow \widetilde{B}(1^\lambda, r, \mathsf{state}_0) \end{array}\right] \\ = \Pr\left[V(r,y) = 1 \;\middle|\; \begin{array}{l} r \leftarrow \{0,1\}^d \\ x = G(1^\lambda, r) \\ y \leftarrow B(1^\lambda, x, \mathsf{state}_0) \end{array}\right] = p \quad .\end{aligned}$$

We now turn to prove the persistence property.

The Purified $\widehat{R}$. We consider the unitary purifications $\{\, \widehat{R}_{\lambda,i,x} \,\}_{\lambda,i,x}$ of $R_B(1^\lambda, 1^i, x, \cdot)$ that act on registers $SY\hat{Y}$. Each such unitary $\widehat{R}_{\lambda,i,x}$ is composed of the unitary purifications of each of the steps $R_B(1^\lambda, 1^i, x, \cdot)$, which are performed coherently. We assume w.l.o.g that $Y\hat{Y}$ consist of the purifying registers for each one of the steps. In particular, $\hat{Y}$ includes registers $P_{i-1}P^*_i$ corresponding to the measurements p_{i-1}, p^*_i done by ValEst, and Y corresponds to the measurement of y_i by Π_{x_i}. Throughout we rely on the fact that the distribution of p_{i-1}, p^*_i measured in any purified interaction $A_z^{\widehat{\mathcal{R}}^*}$ is identical to their distribution in the non-purified interaction $A_z^{\mathcal{R}^*}$.

Lemma 4.3. *Fix any* $\mathsf{state}^* \in \mathbf{S}(\mathcal{H})$ *with purification* $\widehat{R}_{\lambda,0}(\mathbf{0},\mathbf{0})$*. Then* $\mathcal{R}^* = (R_B, \mathsf{state}^*_0)$ *is* $\frac{\eta}{2}$*-persistent, with respect to the purifications* $\{\, \widehat{R}_{\lambda,i,x} \,\}$*, with persistent value* p_0*, where* p_0 *the purifying measurement corresponding to the first application of (the purified)* ValEst*.*

Lemma 4.4. *Except with probability $\eta/2$, over $(\mathsf{state}_0^*, p_0^*) \leftarrow S_B(\mathsf{state}_0)$ and $(\boldsymbol{\sigma}_0, p_0) \leftarrow \mathsf{ValEst}_{V,\widetilde{B}}(\mathsf{state}_0^*, 1^{1/\varepsilon_1})$,*

$$|p_0 - p_0^*| \leq \eta/2 \ .$$

Indeed, combining the two lemmas it follows the distribution on $\mathcal{R}^* = (R_B, \mathsf{state}_0^*)$ induced by $(\mathsf{state}_0^*, p_0^*) \leftarrow S(\mathsf{state}_0)$ is (p_0^*, η)-consistent.

We now prove the above lemmas. Toward proving Lemma 4.3, we first prove two useful claims. In what follows let $\widehat{\mathsf{ValEst}}_\varepsilon$ be a unitary purification of $\mathsf{ValEst}_{V,\widetilde{B}}(\cdot, 1^{1/\varepsilon})$ that operates on registers SPZ, where given $(\boldsymbol{\rho}, \mathbf{0}, \mathbf{0})$, it outputs $(\boldsymbol{\sigma}, \boldsymbol{p}, \boldsymbol{z})$ where $(\boldsymbol{\sigma}, \boldsymbol{p})$ have the same density matrix as $\mathsf{ValEst}(\boldsymbol{\rho}, 1^{1/\varepsilon})$. Then given any pure input state $\boldsymbol{\rho}$, measuring $(\boldsymbol{p}, \boldsymbol{z})$ in the computational basis, purifies $\boldsymbol{\sigma}$. We denote by $(\boldsymbol{\sigma}, p, z) \leftarrow \widehat{\mathsf{ValEst}}(\cdot, 1^{1/\varepsilon})$ the process that applies $\mathsf{ValEst}_\varepsilon$ and measures $(\boldsymbol{p}, \boldsymbol{z})$.

Claim 4.5 (Value of Post-Estimation State)**.** *Let $\boldsymbol{\rho} \in \mathbf{S}(\mathcal{H})$ be a pure state. Then with probability $1 - \varepsilon$ over the measurement of (p, z) in $(\boldsymbol{\sigma}, p, z) \leftarrow \widehat{\mathsf{ValEst}}(\boldsymbol{\rho}, 1^{1/\varepsilon})$, it holds that*

$$\left| p - \Pr\left[V(r,y) = 1 \;\middle|\; \begin{array}{l} r \leftarrow \{0,1\}^d \\ x = G(1^\lambda, r) \\ y \leftarrow \widetilde{B}(1^\lambda, r, \boldsymbol{\sigma}) \end{array} \right] \right| \leq \varepsilon \ .$$

Proof. Consider applying $(\boldsymbol{\sigma}, p, z) \leftarrow \widehat{\mathsf{ValEst}}(\boldsymbol{\rho}, 1^{1/\varepsilon})$ and then applying $(\boldsymbol{\sigma}^*, p^*) \leftarrow \mathsf{ValEst}_{V,\widetilde{B}}(\boldsymbol{\sigma}, 1^{1/\varepsilon})$. Then by the fact that estimation is almost projective (Lemma 4.2), it holds with probability $1 - \varepsilon$ that $|p^* - p| \leq \varepsilon$. Also,

$$\Pr\left[V(r,y) = 1 \;\middle|\; \begin{array}{l} r \leftarrow \{0,1\}^d \\ x = G(1^\lambda, r) \\ y \leftarrow \widetilde{B}(1^\lambda, r, \boldsymbol{\sigma}) \end{array} \right] = \mathbb{E}[p^*] = p + \mathbb{E}(p^* - p) \ .$$

The claim follows. □

Claim 4.6 (Π_x vs $\widetilde{B}$)**.** *For any state $\boldsymbol{\rho} \in \mathbf{S}(\mathcal{H})$,*

$$\Pr\left[V(r,y) = 1 \;\middle|\; \begin{array}{l} r \leftarrow \{0,1\}^d \\ x = G(1^\lambda, r) \\ (\boldsymbol{\rho}^*, y) \leftarrow \Pi_x(\boldsymbol{\rho}) \end{array} \right] = \Pr\left[V(r,y) = 1 \;\middle|\; \begin{array}{l} r \leftarrow \{0,1\}^d \\ x = G(1^\lambda, r) \\ y \leftarrow \widetilde{B}(1^\lambda, r, \boldsymbol{\rho}) \end{array} \right] \ .$$

Proof. By the definition of Π_x it acts exactly as $\widetilde{B}(1^\lambda, r, \cdot)$ with the exception that Π_x replaces with $\perp$ any $y \notin Y_x = \{\, y : K(1^\lambda, x, y) = 1 \,\}$. Recall, however, that Y_x contains all valid solutions $\{\, y : V(1^\lambda, r, y) = 1 \,\}$, and hence (assuming w.l.o.g that $V(1^\lambda, r, \perp) \neq 1$), this difference does not affect whether $V(1^\lambda, r, y) = 1$. □

We now prove Lemma 4.3.

Proof of Lemma 4.3. Fix any solver-aided algorithm A with input z. We consider the random variables $\mathsf{state}_0^*, p_0, p_1^*, \mathsf{state}_1^*, p_1, p_2^*, \mathsf{state}_2^*, p_2, \ldots$ given by extended the solver interaction $A_z^{\widehat{\mathcal{R}}^*}$, where state_i^* are the corresponding pure state and p_{i-1}, p_i^* are the purifying measurements of registers $P_{i-1} P_i^*$.

Claim 4.7 (The Many Shot Value)**.** *For all $i \geq 0$, except with probability ε_{i+1} over $A_z^{\widehat{\mathcal{R}}^*}$,*

$$\left| p_i - \mathsf{val}_P[i, A_z^{\widehat{\mathcal{R}}^*}] \right| \leq \varepsilon_{i+1} \ ,$$

where each random variable $\mathsf{val}_P[i, A_z^{\widehat{\mathcal{R}}^*}]$ is determined by the random variable state_i^*.

Proof. For every i,

$$\mathsf{val}_P[i, A_z^{\widehat{\mathcal{R}}^*}] = \Pr\left[V(r,y) = 1 \;\middle|\; \begin{array}{l} (\boldsymbol{\sigma}_i, p_i, z) \leftarrow \widehat{\mathsf{ValEst}}(\mathsf{state}_i^*, 1^{1/\varepsilon_{i+1}}) \\ r \leftarrow \{0,1\}^d \\ x = G(1^\lambda, r) \\ (\boldsymbol{\sigma}_{i+1}^*, y) \leftarrow \Pi_x(\boldsymbol{\sigma}_i) \end{array} \right] =$$

$$\Pr\left[V(r,y) = 1 \;\middle|\; \begin{array}{l} (\boldsymbol{\sigma}_i, p_i, z) \leftarrow \widehat{\mathsf{ValEst}}(\mathsf{state}_i^*, 1^{1/\varepsilon_{i+1}}) \\ r \leftarrow \{0,1\}^d \\ x = G(1^\lambda, r) \\ y \leftarrow \widetilde{B}(1^\lambda, r, \boldsymbol{\sigma}_i) \end{array} \right] ,$$

where the first equality follows by the definition of $\mathsf{val}_P[i, A_z^{\widehat{\mathcal{R}}^*}]$ and the second by Claim 4.6. The claim now follows from Claim 4.5. □

Claim 4.8 (Persistence)**.** *For all $i \geq 1$, except with probability $2\varepsilon_i$ over $A_z^{\widehat{\mathcal{R}}^*}$,*

$$|p_i - p_i^*| \leq \varepsilon_i \ ,$$
$$|p_i^* - p_{i-1}| \leq \varepsilon_i \ .$$

Proof. The first inequality follows from the estimation is almost projective guarantee and second from the repairing guarantee (both given by Lemma 4.2). □

Combining Claims 4.7 and 4.8, and applying a union bound, we deduce that $3\sum_i \varepsilon_i$-persistence holds, with persistent value p_0. The Lemma now follows by our choice of ε_i:

$$3\sum_i \varepsilon_i = \frac{3\eta}{\pi^2} \sum_i i^{-2} = \eta/2 \ .$$

□

Proof of Lemma 4.4. The lemma follows from the estimation is almost projective guarantee (Lemma 4.2). □

This concludes the proof of Theorem 4.1. □

5 Stateful Solvers To Memoryless Solvers

The following theorem shows that it is possible to convert stateful solvers into memoryless solvers with the same value, albeit with a few caveats. First, the distribution of queries that is to be made to the memoryless solver needs to be known ahead of time (i.e. it needs to be decided upfront in a non-adaptive manner). Second, the resulting memoryless solver might not be efficiently executable. Instead, we provide a simulator that can emulate its behavior, but only once, and only on an input that comes from the prescribed distribution. The simulator only manages to simulate the execution up to some statistical error, and its running time is polynomial in the inverse of this error. A formal theorem statement follows.

Theorem 5.1. *There exists a polynomial time oracle-aided simulator* $\mathsf{SimMemless}$ *with the following properties. Let $\mathcal{B}$ be a (p, η)-persistent ℓ-stateful solver for a falsifiable non-interactive assumption P and let $D = \{D_\lambda\}_\lambda$ be an efficiently samplable distribution ensemble over k-tuples of P instances. Finally, let δ be some parameter. Then there exists a (p, η)-persistent (but possibly inefficient) distribution over memoryless solvers $\mathcal{B}' = \mathcal{B}'_{\ell,D,\delta} = (B', \emptyset)$ for P such that the following holds.*

Consider sampling $\vec{x} \leftarrow D_\lambda$, and let $\mathcal{B}'(1^\lambda, \vec{x})$ be the transcript of the process that feeds the elements of $\vec{x}$ into $\mathcal{B}'$ one-by-one in order (i.e. executes $B'(1^\lambda, 1^i, x_i, \emptyset)$ in order). Then $\mathsf{SimMemless}^{\mathcal{B},D}(1^\lambda, 1^\ell, 1^{1/\delta}, \vec{x})$ *makes non-adaptive black-box access to $\mathcal{B}$ and produces a distribution that is within at most δ statistical distance from $\mathcal{B}'(1^\lambda, \vec{x})$.*

We note that our simulator is "almost" a black-box algorithm in $\mathcal{B}$ in the sense that it takes the size of the state 1^ℓ as input, but otherwise it only makes black-box queries to $\mathcal{B}$. We also emphasize that the simulator does not depend at all on p, η or any other property of $\mathcal{B}$ (other than ℓ).

5.1 The Simulator $\mathsf{SimMemless}$

We start by describing the simulator that will be used to prove Theorem 5.1. The simulator $\mathsf{SimMemless}$ simply "floods" the solver $\mathcal{B}$ with queries from a fixed distribution, and plants the elements of $\vec{x}$ in random positions.

Specifically, $\mathsf{SimMemless}^{\mathcal{B},D}(1^\lambda, 1^\ell, 1^{1/\delta}, \vec{x})$ works as follows. Let t be such that $k\sqrt{\ell/2t} \leq \delta$, i.e. $t = O(\ell(k/\delta)^2)$. The simulator is also going to generate a non-adaptive sequence of queries. We start by defining our "flooding" distribution.

Definition 5.2 (Random Marginal)**.** *Let D be a distribution over X^k, i.e. k-tuples over a domain X. Then the random marginal distribution D_U over X is a distribution obtained by sampling $(x_1, \dots, x_k)$ according to D, sampling a random i in $[k]$, and outputting x_i as the final sample.*

The simulator starts by sampling the following values.

1. A vector $\vec{z}$ of $k \cdot t$ samples $z_{j,i} \leftarrow D_U$, where $j \in [k]$, $i \in [t]$.
2. k uniform samples $i_j \leftarrow [t]$, where $j \in [k]$.
3. A uniform permutation π over $[k]$.

It then generates a sequence of queries $\vec{z}^*$ by taking the vector $\vec{z}$ and, for all $j \in [k]$, replacing z_{j,i_j} with $x_{\pi(j)}$. Namely, thinking of $\vec{z}$ as containing k sequences of queries of length t each, we plug in a random element from $\vec{x}$ in a random location in each sequence.

The simulator then calls $\mathcal{B}$ on the queries in $\vec{z}^*$ in order, to obtain a sequence of responses $\vec{y}$. Let $y_{j,i}$ be the (j,i) element in this sequence. We define $y^*_j = y_{j,i_j}$. The simulator returns the transcript $((x_1, y^*_{\pi^{-1}(1)}), \dots, (x_k, y^*_{\pi^{-1}(k)}))$. Namely, we output a transcript that pairs each x_i with the response that $\mathcal{B}$ produces when introduced to the query $z_{j,i_j} = x_i$, namely $\pi(j) = i$.

5.2 Proving Theorem 5.1

We now turn to prove the theorem. We start by defining a hybrid distribution which is defined with respect to purifying executions of $\mathcal{B}$. This will allow us to make claims about extended transcripts, and finally to redact to standard transcript and derive the proof of the theorem.

A Hybrid Distribution. To prove the theorem, we define the hybrid distribution $\mathcal{S}_h$, defined for every $h \in \{0, 1, \dots, k\}$.

1. Sample a uniform permutation π over $[k]$.
2. For all $j \in [k]$, sample a random index $i_j \in [t]$.
3. Sample $\vec{x}$ from D.
4. Generate a sequence of queries $z_{j,i}$ for all $j \in [k]$, $i \in [t]$ as follows.
   (a) For all $j > h$, set $z_{j,i_j} = x_{\pi(j)}$.
   (b) Otherwise sample z_{j,i_j} from D_U.
5. Generate the extended transcript $\widehat{\mathsf{ts}}$ of executing $\mathcal{B}$ (in a purifying manner) on the entries $z_{j,i}$ in lexicographic order (i.e. starting with $(1,1), \dots, (1,t)$ and concluding with $(k,1), \dots, (k,t)$). We let $\widehat{\mathsf{ts}}_{j,i}$ denote the prefix of the transcript prior to making the (j,i) query. We let $|s_{j,i}\rangle$ denote the solver state respective to $\widehat{\mathsf{ts}}_{j,i}$, as guaranteed by Proposition 3.7. Notice that $|s_{1,1}\rangle$ is the initial state state_0 of $\mathcal{B}$ conditioned on $\widehat{\mathsf{ts}}_0 = \hat{y}_0$.

6. The output of the hybrid $\mathcal{S}_h$ then consists the following values, for all $j \in [k]$:

   (a) The values $i_j, \pi(j)$.

   (b) The quantum state in the beginning of the j-th run: $|s_{j,1}\rangle$.

   (c) The quantum state right before the i_j-th query in the j-th sequence is made: $|s_{j,i_j}\rangle$.

   (d) The value $x_{\pi(j)}$, which is the i_j-th query in the j-th sequence if $j > h$.

   (e) An answer $(y_{\pi(j)}, \hat{y}_{\pi(j)})$ computed as follows.

   - If $j > h$ then set $(y_{\pi(j)}, \hat{y}_{\pi(j)}) = (y_{j,i_j}, \hat{y}_{j,i_j})$ (i.e. the $(y, \hat{y})$-part of the (j, i_j)-th triple in $\widehat{\mathsf{ts}}$).
   - Otherwise generate $(y_{\pi(j)}, \hat{y}_{\pi(j)})$ as $\widehat{B}(1^\lambda, 1^{t(j-1)+i_j}, x_{\pi(j)}, |s_{j,i_j-1}\rangle)_{\mathrm{y},\hat{\mathrm{y}}}$.

In what follows, we will prove that the distributions induced by the first and last hybrids are close in trace distance, as formalized below.

Lemma 5.3. *It holds that* $\mathrm{TD}(\mathcal{S}_0, \mathcal{S}_k) \le k\sqrt{\ell/(2t)}$.

Before proving Lemma 5.3, we argue that it implies the validity of Theorem 5.1. Indeed, we observe that the output of the simulator $\mathsf{SimMemless}$ can be extracted from $\mathcal{S}_0$ by simply outputting all of the pairs $((x_1, y_1), \ldots, (x_k, y_k))$. Applying the same extraction procedure on the last hybrid $\mathcal{S}_k$ will lead to a sequence $((x_1, y_1), \ldots, (x_k, y_k))$ in which $y_{\pi(j)} = B(1^\lambda, 1^{t(j-1)+i_j}, x_{\pi(j)}, |s_{j,i_j-1}\rangle)_{\mathrm{y}}$. However, in the hybrid $\mathcal{S}_k$, the transcript $\widehat{\mathsf{ts}}$, and therefore all states $|s_{j,i}\rangle$, are generated independently of $\vec{x}$. Therefore, for every values of $\pi, \widehat{\mathsf{ts}}$ one could define a memoryless adversary $\mathcal{B}' = (B'_{\pi,\widehat{\mathsf{ts}}}, \emptyset)$, defined by

$$B'_{\pi,\widehat{\mathsf{ts}}}(1^\lambda, 1^j, x, \emptyset) = B(1^\lambda, 1^{t(j'-1)+i_{j'}}, x, |s_{j',i_{j'}-1}\rangle)_{\mathrm{y}} \ , \tag{6}$$

with $j' = \pi^{-1}(j)$. Note that the sequence of states is hard-wired into B' and it does not require to propagate a state throughout the execution.

We therefore indeed have that the solver $\mathcal{B}'$ is a distribution over memoryless solvers indicated by sampling $\pi, \widehat{\mathsf{ts}}$ from their respective distributions and executing $B'_{\pi,\widehat{\mathsf{ts}}}$. Since $\mathcal{B}$ is (p, η)-persistent, we have that with probability $1 - \eta$ over $\widehat{\mathsf{ts}}$, all invocations of $B(1^\lambda, 1^{t(j'-1)+i_{j'}}, x, |s_{j',i_{j'}-1}\rangle)$ have value $p \pm \eta$, which would imply that $(B'_{\pi,\widehat{\mathsf{ts}}}, \emptyset)$ is (p, η)-persistent. Therefore, the distribution $\mathcal{B}'$ is also, by definition, (p, η)-persistent.

The proof of Lemma 5.3 will follow from a standard hybrid argument, given by the following lemma.

Lemma 5.4. *For all* $h \in \{0, 1, \ldots, k-1\}$ *it holds that*

$$\mathrm{TD}(\mathcal{S}_h, \mathcal{S}_{h+1}) \le \sqrt{\ell/(2t)} \ . \tag{7}$$

Proof. We will show that the lemma holds true even when conditioning both $\mathcal{S}_h, \mathcal{S}_{h+1}$ on any value for $\widehat{\mathsf{ts}}_{h,1}$ (the $(h \cdot t)$-prefix of the transcript $\widehat{\mathsf{ts}}$).

We will show that the lemma follows from the following claim.

Claim 5.5. *Conditioning on any value of* $\widehat{\mathsf{ts}}_{h,1}$ *for both* $\mathcal{S}_h, \mathcal{S}_{h+1}$*, the joint distribution of:*

$$(i_h, \widehat{\mathsf{ts}}_{h,i_h}, |s_{h+1,1}\rangle, (x_{\pi(h)}, y_{\pi(h)}, \hat{y}_{\pi(h)})) \tag{8}$$

is within trace distance $\sqrt{\ell/(2t)}$ *between* $\mathcal{S}_h, \mathcal{S}_{h+1}$.

Given Claim 5.5, Lemma 5.4 follows since all other elements of the two distributions $\mathcal{S}_h, \mathcal{S}_{h+1}$ can be sampled given $\widehat{\mathsf{ts}}_{h,1}$ and $(i_h, \widehat{\mathsf{ts}}_{h,i_h}, |s_{h+1,1}\rangle, (x_{\pi(h)}, y_{\pi(h)}, \hat{y}_{\pi(h)}))$, as follows.

1. Sample the permutation π and the query vector $\vec{x}$ conditioned on the value $x_{\pi(h)}$.

2. For very $j \in [k] \setminus \{h\}$, sample i_j uniformly in $[t]$.

3. For all $j < h$, the transcript prefix $\widehat{\mathsf{ts}}_{h,1}$ determines all states $|s_{j,i}\rangle$ (for all $i \in [t]$), which in turn, together with $\vec{x}$, determines the distribution of $y_{\pi(j)}, \hat{y}_{\pi(j)}$ for all $j < h$ (since this distribution is specified by applying the solver B on $x_{\pi(j)}$ with quantum state that is determined by the h-prefix).

4. For all $j > h$ the outputs of both $\mathcal{S}_h, \mathcal{S}_{h+1}$ are determined as the outcomes of an identical quantum process applied to the state $|s_{h+1,1}\rangle$ (the initial state of the $(h+1)$-th sequence), considering that π and $\vec{x}$ have been determined.

We now proceed to prove Claim 5.5, and focus on the distribution of $(i_h, \widehat{\mathsf{ts}}_{h,i_h}, |s_{h+1,1}\rangle, (x_{\pi(h)}, y_{\pi(h)}, \hat{y}_{\pi(h)}))$ in the two hybrids, given that $\widehat{\mathsf{ts}}_{h,1}$ is fixed. The claim follows straightforwardly from our information theoretic Plug-In Lemma (Lemma 2.3), where the classical values y_i in the lemma corresponds to pairs $(z_{h,i}, y_{h,i}, \hat{y}_{h,i})$ generated in the h'th round in the hybrid experiment. Note that since we fixed $\widehat{\mathsf{ts}}_{h,1}$, the distribution over these classical values is also fixed, and indeed the value $\boldsymbol{s} = |s_{h+1,1}\rangle$ depends on this sequence of t values. The triple $(x_{\pi(h)}, y_{\pi(h)}, \hat{y}_{\pi(h)})$ differs between $\mathcal{S}_h$ and $\mathcal{S}_{h+1}$ since in the former it is exactly equal to the i_{h+1} element in the h-th sequence, and in the latter it is sampled from the marginal distribution of this element. We can therefore apply the plug-in lemma directly to obtain the $\sqrt{\ell/(2t)}$ bound on the trace distance as Claim 5.5 requires. This completes the proof of the claim and thus also of the lemma. □

6 Memoryless Solvers To Stateless Solvers

Theorem 6.1. *There exists a polynomial-time oracle-aided simulator* $\mathsf{SimStateless}$ *with the following properties. Let $\mathcal{B}$ be a (p, η)-persistent memoryless solver for a falsifiable non-interactive assumption P and let $\{D_\lambda\}_\lambda$ be an efficiently samplable distribution ensemble over k-tuples of P instances. Let δ be some parameter.*

Then there exists a (p, η)-persistent (but possibly inefficient) stateless solver $\mathcal{B}'' = \mathcal{B}''_\delta = (B'', \emptyset)$ for P such that the following holds. Consider sampling $\vec{x} \leftarrow D_\lambda$, and let $\mathcal{B}''(1^\lambda, \vec{x})$ be the transcript of the process that feeds the elements of $\vec{x}$ into $\mathcal{B}''$ (i.e. executes $B''(1^\lambda, x_i, \emptyset)$ for all x_i). Then $\mathsf{SimStateless}^{\mathcal{B}}(1^\lambda, 1^{1/\delta}, \vec{x})$ *makes non-adaptive black-box access to $\mathcal{B}$ and produces a distribution that is within at most δ statistical distance from $\mathcal{B}''(1^\lambda, \vec{x})$.*

Proof. The simulator $\mathsf{SimStateless}^{\mathcal{B}}$ runs as follows. Given $\vec{x}$ as input, it generates a query vector $\vec{x}'$ of length $t = k^2$ as follows. It samples, without repetitions, k indices $i_1, \ldots, i_k$ and sets $x'_j = x_{i_j}$. All other values of x' are set to 0 (or some other fixed value).

After making the queries in $\vec{z}$ to $\mathcal{B}$ and receiving an output vector $\vec{y}'$, the simulator sets $y_j = y_{i_j}$ returns $((x_1, y_1), \ldots, (x_k, y_k))$.

Let us now define the stateless adversary $\mathcal{B}''$. On input x, $B''(1^\lambda, x)$ samples $j \leftarrow [t]$ uniformly, and outputs $y = B(1^\lambda, 1^j, x, \emptyset)_y$. The solver $\mathcal{B}''$ is also (p, η)-persistent; indeed, its value is the average of values, which are all η-close to p. (Recall Remark 3.4 about persistent values for stateless and memoryless solvers.)

To bound the statistical distance between $\mathsf{SimStateless}^{\mathcal{B}}(1^\lambda, 1^{1/\delta}, \vec{x})$ and $\mathcal{B}''(1^\lambda, \vec{x})$, we consider the case where in the course of the execution of $\mathcal{B}''(1^\lambda, \vec{x})$, all j's that are sampled are distinct. This happens with probability at least $1 - k^2/t = 1 - \delta$. Conditioned on this event, $\mathcal{B}''(1^\lambda, \vec{x})$ is identically distributed as $\mathsf{SimStateless}^{\mathcal{B}}(1^\lambda, 1^{1/\delta}, \vec{x})$. It follows that in general the statistical distance is bounded by δ. □

We conclude with a corollary that combines Theorem 5.1 and Theorem 6.1.

Corollary 6.2. *There exists a polynomial-time simulator* Sim *with the following properties. Let $\mathcal{B}$ be a (p, η)-persistent ℓ-stateful solver for a falsifiable non-interactive assumption P and let $\{D_\lambda\}_\lambda$ be an efficiently samplable distribution ensemble over k-tuples of P instances. Finally, let δ be some parameter.*

Then there exists a (p, η)-persistent (but possibly inefficient) distribution over stateless solvers $\mathcal{B}'' = \mathcal{B}''_{\ell,D,\delta} = (B'', \emptyset)$ for P. Consider sampling $\vec{x}^* \leftarrow D_\lambda$, and let $\mathcal{B}''(1^\lambda, \vec{x}^*)$ be the transcript of the process that feeds the elements of $\vec{x}^*$ into $\mathcal{B}''$ (i.e. executes $B''(1^\lambda, x_i^*, \emptyset)$ for all x_i^*). Then $\mathsf{Sim}^{\mathcal{B},D}(1^\lambda, 1^\ell, 1^{1/\delta}, \vec{x}^*)$ *makes non-adaptive black-box access to $\mathcal{B}$ and produces a distribution that is within at most δ statistical distance from $\mathcal{B}''(1^\lambda, \vec{x}^*)$.*

Proof. The simulator $\mathsf{Sim}^{\mathcal{B},D}(1^\lambda, 1^\ell, 1^{1/\delta}, \vec{x}^*)$ runs as follows. Set $\delta' = \delta/2$.

1. Define an efficiently samplable distribution D' over sequences of P instances as follows. Consider the non-adaptive black-box simulator $\mathsf{SimStateless}$ from Theorem 6.1. Start by sampling $\vec{x} \leftarrow D$. Run $\mathsf{SimStateless}^{(\cdot)}(1^\lambda, 1^{1/\delta'}, \vec{x})$ up until the point where it generates its sequence of oracle queries $\vec{x}'$ (note that to this end there is no need to actually have any access to the solver itself). Let $\vec{x}'$ be the sample of D'.

2. Start an execution $\mathsf{SimStateless}^{(\cdot)}(1^\lambda, 1^{1/\delta'}, \vec{x}^*)$, until the point where the sequence of queries $\vec{x}'^*$ is generated.

3. Consider the non-adaptive black-box simulator $\mathsf{SimMemless}$ from Theorem 5.1. Execute the simulator $\mathsf{SimMemless}^{\mathcal{B},D'}(1^\lambda, 1^\ell, 1^{1/\delta'}, \vec{x}'^*)$ to obtain a transcript ts.

4. Resume the execution of $\mathsf{SimStateless}$ from step 2, plugging in the responses from ts as the solver outcome. Produce the output of $\mathsf{SimStateless}$ as the output of Sim.

To analyze, we first note that by definition $\vec{x}'^*$ is sampled from the distribution D'. Theorem 5.1 implies that there exists a distribution $\mathcal{B}'$ over memoryless adversaries such that the transcript ts is withing δ' statistical distance from having been produced by $\mathcal{B}'(1^\lambda, \vec{x}'^*)$. It therefore follows that the output of $\mathsf{Sim}^{\mathcal{B},D}(1^\lambda, 1^\ell, 1^{1/\delta}, \vec{x}^*)$ is within statistical distance δ' from $\mathsf{SimStateless}^{\mathcal{B}'}(1^\lambda, 1^{1/\delta'}, \vec{x}^*)$. We can now apply Theorem 6.1 to deduce that for each memoryless solver $\mathcal{B}'_0$ in the support of $\mathcal{B}'$, the latter is within δ' statistical distance from some $\mathcal{B}''_0(1^\lambda, \vec{x}')$ where $\mathcal{B}''_0$ is stateless. This therefore induces a distribution over stateless solvers. Applying the union bound we get that $\mathsf{Sim}^{\mathcal{B},D}(1^\lambda, 1^\ell, 1^{1/\delta}, \vec{x}^*)$ is within statistical distance at most $2\delta' = \delta$ from $\mathcal{B}''(1^\lambda, \vec{x}')$ as required.

By definition, with probability $1 - \eta$ over the sampling of the memoryless solver $\mathcal{B}'_0$ from the distribution, $\mathcal{B}'_0$ itself is (p, η)-persistent, this property carries over to $\mathcal{B}''_0$. Therefore, the distribution $\mathcal{B}''$ is by definition (p, η)-persistent. □

7 Classical Non-Adaptive Reductions and Quantum Solvers

In this section, we show that a wide class of classical reductions can be translated to the quantum setting. Specifically we start from any non-adaptive black-box reductions from classically solving P with a verifiably-polynomial image (Definition 3.2), to classically solving Q. We transform it into a quantum reduction from quantumly solving P to quantumly solving Q.

Theorem 7.1. *Assume there exists a classical non-adaptive black-box reduction from solving a non-interactive assumption Q to solving a non-interactive assumption P with a verifiably-polynomial image. Then there exists a quantum reduction from solving Q to quantumly solving P. This reduction is durable if the original classical reduction has positive advantage.*

Proof. Let $\mathcal{R}$ be a classical non-adaptive black-box reduction from solving a non-interactive assumption $Q = (G_Q, V_Q, c_Q)$ to solving a non-interactive assumption $P = (G_P, V_P, c_P)$. We present a quantum reduction $\mathcal{R}'$ from solving Q to quantumly solving P. We start by describing and analyzing $\mathcal{R}'$ with a one-shot advantage, and then extend it to address durability in the case that $\mathcal{R}$ has positive advantage. We assume w.l.o.g that $\mathcal{R}$ never makes the same query twice to its oracle function (see Remark 3.8).

Recalling Definition 3.12, $\mathcal{R}'$ takes as input $(1^\lambda, 1^t, x_Q, \mathsf{state})$, where $x_Q \in \{0,1\}^{n_Q}$ is potentially an instance of Q, and its initial state is $\mathsf{state}'_0 = (\mathsf{state}_0, B, 1^{1/\varepsilon}, 1^T)$, where we are guaranteed that $\mathcal{B} = (\mathsf{state}_0, B)$ is a P solver with advantage at least ε that runs in time at most T.

We let ε' denote the advantage of $\mathcal{R}$ in solving Q when given access to an oracle that solves P with advantage at least $\varepsilon/2$. We are guaranteed that $\varepsilon' = \mathrm{poly}(\varepsilon, \lambda^{-1})$. We set $\delta = \varepsilon'/2$ and $\eta = \min\{\varepsilon/4, \varepsilon'/2\}$.

We define a distribution D over $(\{0,1\}^{n_P})^k$ as the distribution over the set of oracle queries produced by first sampling a uniform r'_Q and using it to generate $x'_Q = G_Q(1^\lambda, r'_Q)$, and finally executing $\mathcal{R}(1^\lambda, 1^{4/\varepsilon}, x'_Q)$ to produce a k-tuple of P-instances.

Having all of these definitions in place, we can now introduce the execution of $\mathcal{R}'(1^\lambda, 1^0, x_Q, \mathsf{state}'_0)$. Namely, we start by analyzing the one-shot execution of $\mathcal{R}'$ (the case $t = 0$).

1. Let R, S be the state restoration algorithms with respect to P as guaranteed by Theorem 4.1, with parameter η as defined above. Set $(\mathsf{state}^*_0, p^*) \leftarrow S_B(\mathsf{state}_0)$. Define $\mathcal{B}_0 = \mathcal{R}^* = (R_B, \mathsf{state}^*_0)$ and recall that $\mathcal{B}_0$ is (p^*, η)-persistent, and that $\mathbb{E}[p^*] = p$.
2. Execute $\mathcal{R}(1^\lambda, 1^{3/\varepsilon}, x_Q)$ to obtain the sequence of queries $\vec{x}$.
3. Recall the simulator Sim guaranteed by Corollary 6.2. Execute $\mathsf{Sim}^{\mathcal{B}_0, D}(1^\lambda, 1^\ell, 1^{1/\delta}, \vec{x})$ to obtain a transcript ts.
4. Extract the responses to $\vec{x}$ from ts and resume the execution $\mathcal{R}$ from step 2 with these responses. Once the execution of $\mathcal{R}$ completes and a value y_Q is output, output y_Q as the output of $\mathcal{R}'$.

To analyze the one-shot value and advantage of $\mathcal{R}'$, we start by analyzing the performance of $\mathcal{R}'$ conditioned on obtaining a fixed value p^* in step 1 of the execution. In this case $\mathcal{B}_0$ is (p^*, η)-persistent, and we can invoke Corollary 6.2 to conclude that there exists a (p^*, η)-persistent distribution over stateless adversaries $\mathcal{B}''_{p^*}$ s.t. the output of $\mathcal{R}'$ is within statistical distance δ from the execution of $\mathcal{R}^{\mathcal{B}''_{p^*}}(1^\lambda, 1^{4/\varepsilon}, x_Q)$.

In turn, the execution of $\mathcal{R}^{\mathcal{B}''_{p^*}}(1^\lambda, 1^0, x_Q, \mathsf{state}'_0)$ is equivalent to executing $\mathcal{R}^{\mathcal{B}''}(1^\lambda, 1^{4/\varepsilon}, x_Q)$, where $\mathcal{B}''$ is a distribution over stateless solvers defined as follows. First sample p^* from its designated distribution, then sample $\mathcal{B}''_{p^*}$ from the (p^*, η)-persistent distribution of stateless solvers. Recall that with probability $1 - \eta$ over the sampling of $\mathcal{B}''_{p^*}$, it holds that the outcome is a (single) (p^*, η)-persistent stateless solver and therefore that $\left|\mathsf{val}_P[0, \mathcal{B}''_{p^*}] - p^*\right| \le \eta$. It follows that with probability at least $1 - \eta$:

$$\begin{aligned} |\mathbb{E}[\mathsf{val}_P[0, \mathcal{B}'']] - p| &= \left|\mathbb{E}\left[\mathsf{val}_P[0, \mathcal{B}''_{p^*}] - p^*\right]\right| \\ &\le \mathbb{E}\left[\left|\mathsf{val}_P[0, \mathcal{B}''_{p^*}] - p^*\right|\right] \\ &\le \eta \ . \end{aligned}$$

It follows that $\mathcal{B}''$ has advantage at least $\varepsilon - 2\eta \ge \varepsilon/2$ in solving P. We have therefore that $\mathcal{R}^{\mathcal{B}''}$ has advantage at least ε' in solving Q. Since the output of $\mathcal{R}'$ is within $\delta = \varepsilon'/2$ statistical distance from $\mathcal{R}^{\mathcal{B}''}$, we conclude that $\mathcal{R}'$ has advantage at least $\varepsilon'/2$. We therefore established the one-shot value of $\mathcal{R}'$.

It remains to extend the definition of $\mathcal{R}'$ beyond $t = 0$ in order to establish that it is durable when $\mathcal{R}$ has positive advantage. The basic idea is to propagate the final state of $\mathcal{B}_0$ at the end of step 4 as the initial state of the next execution, and use the persistence of $\mathcal{B}_0$ in order to execute steps 2-4 anew for each input.

In order to formalize the above intuition, we require the following definitions. We define a "shifted execution" of a solver as follows. Letting $\mathcal{B} = (B, \mathsf{state}_0)$ be a solver. We define the solver $\mathcal{B}_{+j} = (B_{+j}, \mathsf{state}_0)$ via $B(1^\lambda, 1^t, x, \mathsf{state}) = B(1^\lambda, 1^{t+j}, x, \mathsf{state})$. Namely, $\mathcal{B}_{+j}$ simply executes $\mathcal{B}$ but with a fixed offset in the t input. A second notation that we require is for the maximal number of $\mathcal{B}_0$ calls that are made in steps 2-4 of the execution above. We denote this value by M and note that it is w.l.o.g a polynomial in $\lambda, \varepsilon^{-1}$ that does not depend on the value of x_Q.

Our durable reduction is therefore as follows:

- We extend the execution of $\mathcal{R}'$ for $t = 0$ defined above as follows. First, we ensure that steps 2-4 make *exactly* M queries to $\mathcal{B}_0$, by inserting dummy queries if needed. Second, we specify the output state of the execution to be the output state of $\mathcal{B}_0$ after the last call that has been made.
- For a value of $t > 0$ the execution of $\mathcal{R}'(1^\lambda, 1^t, x, \mathsf{state})$ is by executing steps 2-4 above (with the padding to M queries), but using the shifted solver $\mathcal{B}_{0+tM}$ instead of $\mathcal{B}_0$. The output state is again the final output state of the solver $\mathcal{B}_{0+tM}$.

Note that $\mathcal{B}_0$ is (p^*, η)-persistent with respect to some purification $\widehat{\mathcal{B}}_0$. We can consider a corresponding purification $\widehat{\mathcal{R}}'$ of $\mathcal{R}'$. We note that in an extended interaction $A_z^{\widehat{\mathcal{R}}'}$, letting $\varepsilon^* = |p^* - c_P|$, it holds that with probability $1 - \eta$, for every i:

$$\mathsf{val}_Q[i, A_z^{\widehat{\mathcal{R}}'}] \ge c_Q + \varepsilon' \ ,$$

where $\varepsilon' \geq 0$ and if $\varepsilon^* - \eta \geq \varepsilon/4$, then $\varepsilon' \geq \mathrm{poly}(\varepsilon^* - \eta, \lambda^{-1})$.

Indeed, since $\mathcal{B}_0$ is (p^*, η)-persistent, in the i-th underlying invocation of $\mathcal{B}''_{p^*}$, its value as a P-solver is η-close to p^*, and hence its advantage in the corresponding invocation of $\mathcal{R}(x_Q, 1^\lambda, 1^{4/\varepsilon})$ is at least $\varepsilon^* - \eta$. If $\varepsilon^* - \eta \geq \varepsilon/4$, $\mathcal{R}$ is guaranteed to have positive advantage $\mathrm{poly}(\varepsilon^* - \eta, \lambda^{-1})$, in which case the corresponding value is $c_Q + \mathrm{poly}(\varepsilon^* - \eta, \lambda^{-1})$.

It is left to argue that

$$\mathbb{E}[\varepsilon'] \geq \mathrm{poly}(\varepsilon, \lambda^{-1}) \ .$$

This follows from an averaging argument. With probability at least $\varepsilon/2$, $\varepsilon^* \geq \varepsilon/2$. In particular with probability at least $\varepsilon/2 - \eta \geq \varepsilon/4$ it holds that $\varepsilon^* - \eta \geq \varepsilon/4$, in which case $\varepsilon' \geq \mathrm{poly}(\varepsilon/4, \lambda^{-1})$. □

8 An Impossibility Result for Search Assumptions

Our result in Section 7 transforms a classical non-adaptive reduction $\mathcal{R}$ from solving Q to classically solving P into a reduction $\mathcal{R}$ to *quantumly* solving P. It is restricted to assumptions P with a verifiably-polynomial image. While this captures a large class of assumptions, such as all decision assumptions, it certainly does not capture all assumptions of interest. In particular, it does not capture *search assumptions* where the number of possible solutions per instance could be super polynomial, such as say the hardness of inverting a one-way function where the preimage size could be super-polynomial.

In this section we show that this is somewhat inherent. We prove that for search assumptions, such a transformation cannot exist as long as the resulting reduction $\mathcal{R}'$ is explicit in the assumptions P, Q. In particular, it may obtain as input the code of the algorithms describing P, Q, but does not get any implicit non-uniform advice regarding these assumptions. Indeed, the transformation in 7 as well as the Persistence Theorem 4.1 on which it relies, the resulting quantum reduction $\mathcal{R}'$ is in fact black-box in the assumptions P, Q, and in particular explicit.

Definition 8.1 (Assumption Pair Colletion)**.** *An assumption pair collection $\mathcal{PQ}$ consists of pairs of assumptions (P, Q), each given by its corresponding (possibly non-uniform) algorithms (G_P, V_P, c_P) and (G_Q, V_Q, c_Q).*

Definition 8.2 (Explicit Reduction)**.** *An explicit quantum reduction for assumption pair collection $\mathcal{PQ}$ is an efficient algorithm $\mathcal{R}$ with the following guarantee. For any $(P, Q) \in (\mathcal{P}, \mathcal{Q})$ and any quantum solver $\mathcal{B}_P = (B_P, \mathsf{state}_0)$ for P with one-shot advantage ε and running time T, let $\mathsf{state}'_0 = (\mathsf{state}_0, (P, Q), B_P, 1^{1/\varepsilon}, 1^T)$. Then $\mathcal{B}_Q = (\mathcal{R}, \mathsf{state}'_0)$ is a solver for Q with one-shot advantage $\mathrm{poly}(\varepsilon, T^{-1}, \lambda^{-1})$ and running-time $\mathrm{poly}(T, \varepsilon^{-1}, \lambda)$.*

We say that the reduction is strongly explicit, instead of being given the explicit description of (P, Q) as part of its input, it is given oracle access to its corresponding algorithms.

Note that in the above definition state'_0 is formally a sequence

$$\mathsf{state}'_{0,\lambda} = (\mathsf{state}_{0,\lambda}, (P, Q)_\lambda, B_{P,\lambda}, 1^{1/\varepsilon(\lambda)}, 1^{T(\lambda)}) \ ,$$

where $(P, Q)_\lambda$ consist of their corresponding algorithms (possibly along with their corresponding non-uniform advice) restricted to security parameter λ (w.l.o.g circuits).

Restating our result from Section 7, we proved that for any pair collection $\mathcal{PQ}$, if for any $(P, Q) \in \mathcal{P}, \mathcal{Q}$, P has verifiably-polynomial image, and there exists a classical non-adaptive black-box reduction $\mathcal{R}_{P,Q}$ from solving Q to solving P, then there also exists a strongly explicit quantum reduction $\mathcal{R}'$ for $\mathcal{PQ}$. We prove that if P does not have a verifiably-polynomial image this may not be the case.

Theorem 8.3. *There exists an assumption pair collection $\mathcal{PQ}$, such that for any $(P, Q) \in \mathcal{P}, \mathcal{Q}$, there exists a classical non-adaptive black-box reduction $\mathcal{R}_{P,Q}$ from solving Q to solving P, but there is no strongly explicit reduction $\mathcal{R}'$ for $\mathcal{PQ}$. Assuming also post-quantum indistinguishability obfuscation, there also does not exist and explicit reduction $\mathcal{R}'$.*

We will restrict extension to ruling out explicit reductions based on indistinguishability obfuscation. The result for strongly explicit reductions is a direct extension.

The Collection $\mathcal{PQ}$. The collection is associated with a particular signature scheme $(\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$, with a corresponding message space $\mathcal{M} = (\mathcal{M}_\lambda)_{\lambda\in\mathbb{N}}$. Each pair of assumptions $(P, Q) \in \mathcal{PQ}$ has the following form:

1. For $i \in \{P, Q\}$, $P_i = (G_i, V_i, 0)$, G_i is a uniform generator and V_i is a non-uniform verifier.
2. G_P takes as input 1^λ and outputs a random message $x \leftarrow \mathcal{M}_\lambda$, whereas G_Q takes as input 1^λ and outputs two random and independent messages $x_1, x_2 \leftarrow \mathcal{M}_\lambda$.
3. V_P and V_Q have the same $\mathsf{pk} \in \mathsf{Gen}(1^\lambda)$ hardwired into their description.

   V_P takes as input $(1^\lambda, x, \sigma)$ and outputs 1 if and only if $\mathsf{Ver}(\mathsf{pk}, x, \sigma) = 1$, whereas V_Q takes as input $(1^\lambda, x_1, x_2, \sigma_1, \sigma_2)$ and it outputs 1 if and only if $\mathsf{Ver}(\mathsf{pk}, x_1, \sigma_1) = \mathsf{Ver}(\mathsf{pk}, x_2, \sigma_2) = 1$.

Claim 8.4. *For any signature scheme* $(\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ *and corresponding collection $\mathcal{PQ}$, there exists an efficient solver-aided algorithm $\mathcal{R}$ such that for every $(P, Q) \in \mathcal{PQ}$, $\mathcal{R}$ is a classical non-adaptive black-box reduction from solving P to solving Q.*

Proof. We describe the reduction $\mathcal{R}$. On input $(1^\lambda, (x_1, x_2))$, $\mathcal{R}$ queries the solver with x_1 and x_2, and obtains σ_1 and σ_2. It outputs (σ_1, σ_2). To complete the proof, we note that given any stateless classical P-solver $\mathcal{B}$ with advantage ε, $\mathcal{R}^{\mathcal{B}}$ has advantage ε^2 in solving Q, as desired. □

We now proceed to show that for an appropriately chosen signature scheme $(\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ there is no explicit quantum reduction $\mathcal{R}'$ for the collection $\mathcal{PQ}$.

Tokenized Signature Schemes. A tokenized signature scheme [BS16] is a classical signature scheme $(\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$, with message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$, with two additional efficient quantum algorithms $(\mathsf{TokenGen}, \mathsf{TokenSig})$. $\mathsf{TokenGen}$ takes as input a secret key sk and outputs a quantum state $|\mathsf{tk}\rangle$, referred to as a *signing token*, and $\mathsf{TokenSig}$ takes as input a signing token $|\mathsf{tk}\rangle$ and a message $m \in \mathcal{M}$ and outputs a signature, with the guarantee that for every message $m \in \mathcal{M}$,

$$\Pr[\mathsf{Ver}(\mathsf{pk}, m, \sigma) = 1] = 1,$$

where the probability is over $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\lambda)$, $|\mathsf{tk}\rangle \leftarrow \mathsf{TokenGen}(\mathsf{sk})$ and $\sigma \leftarrow \mathsf{TokenSig}(|\mathsf{tk}\rangle, m)$.

Definition 8.5. *A tokenized signature scheme* $(\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver}, \mathsf{TokenGen}, \mathsf{TokenSig})$ *is secure if for any efficient quantum adversary $\mathcal{A}$ there exists a negligible function μ such that for every $\lambda \in \mathbb{N}$,*

$$\Pr\left[\begin{array}{c} m_1 \neq m_2, \\ \mathsf{Ver}(\mathsf{pk}, m_i, \sigma_i) = 1 \quad \forall i \in [2] \end{array} \middle| (m_1, m_2, \sigma_1, \sigma_2) \leftarrow \mathcal{A}(\mathsf{pk}, |\mathsf{tk}\rangle)\right] = \mu(\lambda) \ ,$$

where the probability is over $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\lambda)$ *and* $|\mathsf{tk}\rangle \leftarrow \mathsf{TokenGen}(\mathsf{sk})$.

We rely on the following result by Coladangelo et a..

Theorem 8.6 ([CLLZ21])**.** *There exists a secure tokenized signature scheme assuming the existence of a post-quantum secure (classical) indistinguishability obfuscation scheme.*

Claim 8.7. *Let* $(\mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver}, \mathsf{TokenGen}, \mathsf{TokenSig})$ *be a secure tokenized signature scheme, and let $\mathcal{PQ}$ be the corresponding collection $\mathcal{PQ}$. Then there exist no explicit quantum reduction for $\mathcal{PQ}$.*

Proof. Assume toward contradiction that there exists an explicit reduction $\mathcal{R}'$ for $\mathcal{PQ}$, we show how an adversary $\mathcal{A}$ can use it to $reakthesecu$ity of the tokenized signatures (Definition 8.5).

For security parameter λ, $\mathcal{A}$ is given a public key pk and token $|\mathsf{tk}\rangle$. $\mathcal{A}$ samples $x_1, x_2 \leftarrow \mathcal{M}_\lambda$ and invokes:

$$\mathcal{R}'((x_1, x_2), \mathsf{state}'_0 = (|\mathsf{tk}\rangle, (P, Q)_\lambda, B_P, 1^{1/\varepsilon}, 1^T) \ ,$$

where:

- $(P, Q)_\lambda$ are the circuits describing the assumption corresponding to pk.
- B_P is the quantum algorithm that given $(x, |\mathsf{tk}\rangle)$, applies $\mathsf{TokenSig}$ to generate a signature σ on x.
- $\varepsilon = 1$.
- and T is the polynomial running time of B_P.

$\mathcal{A}$ obtains back from the reduction two signatures σ_1, σ_2 and outputs $(x_1, \sigma_1, x_2, \sigma_2)$.

To see that $\mathcal{A}$ breaks the security of the tokenized signature (namely manages to generate signatures on two different messages. Note that $B_P, \mathsf{state}_0 = |\mathsf{tk}\rangle$ constitute a solver for P (which generates one good signature) with probability 1, accordingly the reduction $\mathcal{R}'$ manages to solve Q with noticeable probability, generating two signatures σ_1, σ_2.

□

The above indeed relies on indistinguishability obfuscation to instantiate the tokenized signature scheme. To get an unconditional impossibility, but which only rules out *strongly* explicit reductions, we can rely on the following result of Ben-David and Sattath.

Theorem 8.8 ([BS16])**.** *There exists a classical oracle distribution relative to which there exist (information theoretically) secure tokenized signature schemes.*

The proof is a direct extension of the proof above.

9 Proving the Plug-In Lemma

We recall the formal statement of the lemma.

Lemma 9.1 (Lemma 2.3, restated)**.** *Let $\vec{Y} = (Y_1, \ldots, Y_t)$ be a vector of arbitrarily jointly distributed classical random variables. Let $\vec{y}$ be distributed according to $\vec{Y}$. Let $\boldsymbol{s}$ be an ℓ-qubit random variable that has arbitrary dependence on $\vec{y}$. We let $\vec{y}_i$ denote the prefix $\vec{y}_i = (y_1, \ldots, y_i)$ for $1 \le i \le t$, and $\vec{y}_0$ is the empty vector (and likewise for $\vec{Y}$). Let J be the uniform distribution over $[t]$ and let $j \leftarrow J$. Define $y' \leftarrow Y_J | (\vec{Y}_{j-1} = \vec{y}_{j-1})$. Then it holds that*

$$\mathrm{TD}((j, \vec{y}_{j-1}, y_j, \boldsymbol{s}), (j, \vec{y}_{j-1}, y', \boldsymbol{s})) \le \sqrt{\ell/(2t)} \ . \tag{9}$$

We prove the lemma using tools from (quantum) information theory. We recall the basic notions below and refer to [NC16, Wat18] for additional reference. We let $H(\cdot)$ denote the entropy function both in the classical case (Shannon entropy) and in the quantum case (von Neumann entropy).

We denote the mutual information function by $I(\cdot : \cdot)$ both in the classical and in the quantum setting. We note that entropy or mutual information are only well defined for variables that have a well-defined joint density matrix.[6]

Let X and Y be variables with a joint density matrix ρ_{XY}. We say that Y is a classical random variable if its reduced density matrix is diagonal. If Y is a classical variable then

$$\rho_{XY} = \sum_y p_y \rho_{X|y} \otimes |y\rangle \langle y| \ . \tag{10}$$

In this case we refer to $\rho_{X|y}$ as the conditional density matrix of X given $Y = y$, and refer to $X|y$ as the variable with this density matrix. In such a case it holds that

$$H(X|Y) = \mathop{\mathbb{E}}_{y}[H(X|y)],$$

[6] Recall the following information-theoretic identities that hold both in the classical and quantum settings. Mutual information: $I(X : Y) = H(X) + H(Y) - H(X, Y)$. Conditional entropy: $H(X|Y) = H(X, Y) - H(Y)$. Conditional mutual information: $I(X : Y|Z) = H(X|Z) + H(Y|Z) - H(X, Y|Z) = H(X|Z) - H(X|Y, Z)$.

where y is distributed according to the classical distribution of Y. We provide a proof for the sake of completeness.

Proposition 9.2. *Let X, Y be random variables with density matrix $\rho_{XY} = \sum_y \alpha_y \rho_{X|y} \otimes |y\rangle\langle y|$. Then it holds that $H(X|Y) = \mathbb{E}_{y\sim Y}[H(X|y)]$.*

Proof. As stated in [NC16, Theorem 11.8 (5)], it holds that $H(X,Y) = H(Y) + \mathbb{E}_{y\sim Y}[H(X|y)]$. Recalling that $H(X|Y) = H(X,Y) - H(Y)$, the proposition follows. □

Corollary 9.3. *Letting X, Y be as in Proposition 9.2 then $H(X|Y) \geq 0$.*

Proof. The corollary follows since for each y it holds that $X|y$ is just a quantum variable and therefore it has non-negative entropy. The expectation over non-negative values remains non-negative. □

The following simple application of the chain-rule will be useful for us.

Lemma 9.4. *Let $\vec{y} = (y_1, \ldots, y_t)$ be a vector of arbitrarily distributed classical random variables, and let $\boldsymbol{s}$ be an ℓ-qubit random variable that has arbitrary dependence on $\vec{y}$. Recall that we denote $\vec{y}_i = (y_1, \ldots, y_i)$ for $1 \leq i \leq t$, and $\vec{y}_0$ is the empty vector. Then for a uniformly distributed $J \leftarrow [t]$,*

$$I(\boldsymbol{s} : y_J | \vec{y}_{J-1}, J) \leq \frac{\ell}{t} \ . \tag{11}$$

Proof. We have that

$$\begin{aligned}
I(\boldsymbol{s} : y_J | \vec{y}_{J-1}, J) &= \mathop{\mathbb{E}}_{j}[I(\boldsymbol{s} : y_j | \vec{y}_{j-1})] && (12)\\
&= \frac{1}{t} \sum_{j\in[t]} I(\boldsymbol{s} : y_j | \vec{y}_{j-1}) && (13)\\
\text{(By definition)} \qquad &= \frac{1}{t} \sum_{j\in[t]} (H(\boldsymbol{s}|\vec{y}_{j-1}) - H(\boldsymbol{s}|\vec{y}_j)) && (14)\\
\text{(Telescopic sum)} \qquad &= \frac{1}{t}(H(\boldsymbol{s}) - H(\boldsymbol{s}|\vec{y})) && (15)\\
\text{(Corollary 9.3)} \qquad &\leq \frac{H(\boldsymbol{s})}{t} && (16)\\
(\boldsymbol{s} \text{ is } \ell\text{-qubits}) \qquad &\leq \frac{\ell}{t} \ . && (17)
\end{aligned}$$

□

Proposition 9.5. *Let Z be a classical variable and let X, Y be quantum variables with arbitrary dependence on Z. Then it holds that*

$$\mathrm{TD}(XZ, YZ) = \mathop{\mathbb{E}}_{z\sim Z}[\mathrm{TD}(X|z, Y|z)] \ . \tag{18}$$

Proof. Since Z is classical, the density matrices of XZ and YZ can be written as block-diagonal: $\rho_{XZ} = \sum_z p_z \rho_{X|z} \otimes |z\rangle\langle z|$ and $\rho_{YZ} = \sum_z p_z \rho_{Y|z} \otimes |z\rangle\langle z|$, where $p_z = \Pr[Z = z]$.

Recall that the ℓ_p norm of a block-diagonal matrix is simply the sum of norms of the blocks (since each block can be individually diagonalized). We therefore have

$$\begin{aligned}
\mathrm{TD}(XZ, YZ) &= \tfrac{1}{2}\|\rho_{XZ} - \rho_{YZ}\|_1 && (19)\\
&= \tfrac{1}{2}\| \sum_z p_z(\rho_{X|z} - \rho_{Y|z}) \otimes |z\rangle\langle z| \,\|_1 && (20)\\
&= \tfrac{1}{2} \sum_z p_z \|\rho_{X|z} - \rho_{Y|z}\|_1 && (21)\\
&= \mathop{\mathbb{E}}_{z\sim Z}[\mathrm{TD}(X|z, Y|z)] \ . && (22)
\end{aligned}$$

□

We use the following lemma which follows straightforwardly from quantum Pinsker inequality.

Lemma 9.6. *Let X, Y be arbitrary quantum variables with a joint density matrix ρ_{XY} and reduced density matrices ρ_X, ρ_Y respectively.*[7] *Then*

$$\mathrm{TD}(\rho_{XY}, \rho_X \otimes \rho_Y) \leq \sqrt{\tfrac{\ln(2)}{2} \cdot I(X:Y)} \leq \sqrt{I(X:Y)/2}\ , \tag{23}$$

where TD *denotes the trace distance.*

Proof. This is a direct application of quantum Pinsker inequality [Wat18, Theorem 5.38], when bearing in mind the connection between quantum divergence and mutual information as expressed in [Wat18, Eq. (5.110)]. □

We can finally prove the plug-in lemma.

Proof of Lemma 2.3. For convenience, we denote $y'_j = y'$. We start by noticing that by definition, conditioned on $j, \vec{y}_{j-1}$, the value $(y'_j, \boldsymbol{s})$ is simply the product distribution of the marginals of $(y_j, \boldsymbol{s})$.

Thus,

$$\begin{aligned}
\mathrm{TD}((j, \vec{y}_{j-1}, y_j, \boldsymbol{s}), (j, \vec{y}_{j-1}, y'_j, \boldsymbol{s})) &= \mathop{\mathbb{E}}_{j, \vec{y}_{j-1}} \Big[\mathrm{TD}((\boldsymbol{s}, y_j)|(j, \vec{y}_{j-1}), (\boldsymbol{s}, y'_j)|(j, \vec{y}_{j-1})) \Big] \\
\text{(Proposition 9.5)} &\leq \mathop{\mathbb{E}}_{j, \vec{y}_{j-1}} \left[\sqrt{\tfrac{1}{2} I(\boldsymbol{s}|(j, \vec{y}_{j-1}) : y_j|(j, \vec{y}_{j-1}))} \right] \\
\text{(Lemma 9.6)} &\leq \sqrt{\tfrac{1}{2} \mathop{\mathbb{E}}_{j, \vec{y}_{j-1}} [I(\boldsymbol{s}|(j, \vec{y}_{j-1}) : y_j|(j, \vec{y}_{j-1}))]} \\
\text{(Convexity (Jensen's Inequality))} &= \sqrt{\tfrac{1}{2} I(\boldsymbol{s} : y_j | j, \vec{y}_{j-1})]} \\
\text{(Lemma 9.4)} &\leq \sqrt{\ell/(2t)}\ .
\end{aligned}$$

□

References

[AC02] Mark Adcock and Richard Cleve. A quantum goldreich-levin theorem with cryptographic applications. In Helmut Alt and Afonso Ferreira, editors, *STACS 2002, 19th Annual Symposium on Theoretical Aspects of Computer Science, Antibes - Juan les Pins, France, March 14-16, 2002, Proceedings*, volume 2285 of *Lecture Notes in Computer Science*, pages 323–334. Springer, 2002.

[BCM+18] Zvika Brakerski, Paul Christiano, Urmila Mahadev, Umesh V. Vazirani, and Thomas Vidick. A cryptographic test of quantumness and certifiable randomness from a single quantum device. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 320–331, 2018.

[BDF+11] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random oracles in a quantum world. In *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, pages 41–69, 2011.

[BG11] Zvika Brakerski and Oded Goldreich. From absolute distinguishability to positive distinguishability. In Oded Goldreich, editor, *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation - In Collaboration with Lidor Avigad, Mihir*

[7] Recall that the reduced density matrix of $\rho_{X,Y}$ corresponding to X is $\rho_X = \mathsf{tr}_Y(\rho_{XY})$, where tr_Y is the linear operator that satisfies that $\mathsf{tr}_Y(|x_1\rangle\langle x_2| \otimes |y_1\rangle\langle y_2|) = |x_1\rangle\langle x_2|\mathsf{tr}(|y_1\rangle\langle y_2|)$ for any $|x_1\rangle$ and $|x_2\rangle$ in X and $|y_1\rangle$ and $|y_2\rangle$ in Y.

Bellare, Zvika Brakerski, Shafi Goldwasser, Shai Halevi, Tali Kaufman, Leonid Levin, Noam Nisan, Dana Ron, Madhu Sudan, Luca Trevisan, Salil Vadhan, Avi Wigderson, David Zuckerman, volume 6650 of *Lecture Notes in Computer Science*, pages 141–155. Springer, 2011.

[BS16] Shalev Ben-David and Or Sattath. Quantum tokens for digital signatures. *CoRR*, abs/1609.09047, 2016.

[CLLZ21] Andrea Coladangelo, Jiahui Liu, Qipeng Liu, and Mark Zhandry. Hidden cosets and applications to unclonable cryptography. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part I*, volume 12825 of *Lecture Notes in Computer Science*, pages 556–584. Springer, 2021.

[CMSZ21] Alessandro Chiesa, Fermi Ma, Nicholas Spooner, and Mark Zhandry. Post-quantum succinct arguments. *CoRR*, abs/2103.08140, 2021.

[DFM20] Jelle Don, Serge Fehr, and Christian Majenz. The measure-and-reprogram technique 2.0: Multi-round fiat-shamir and more. In *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III*, pages 602–631, 2020.

[DFMS19] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Security of the fiat-shamir transformation in the quantum random-oracle model. In *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*, pages 356–383, 2019.

[DFSS08] Ivan Damgård, Serge Fehr, Louis Salvail, and Christian Schaffner. Cryptography in the bounded-quantum-storage model. *SIAM J. Comput.*, 37(6):1865–1890, 2008.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In David S. Johnson, editor, *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washington, USA*, pages 25–32. ACM, 1989.

[HH09] Iftach Haitner and Thomas Holenstein. On the (im)possibility of key dependent encryption. In Omer Reingold, editor, *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*, volume 5444 of *Lecture Notes in Computer Science*, pages 202–219. Springer, 2009.

[JZC+18] Haodong Jiang, Zhenfeng Zhang, Long Chen, Hong Wang, and Zhi Ma. Ind-cca-secure key encapsulation mechanism in the quantum random oracle model, revisited. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part III*, pages 96–125, 2018.

[KS20] Juliane Krämer and Patrick Struck. Encryption schemes using random oracles: From classical to post-quantum security. In *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020, Paris, France, April 15-17, 2020, Proceedings*, pages 539–558, 2020.

[KYY18] Shuichi Katsumata, Shota Yamada, and Takashi Yamakawa. Tighter security proofs for GPV-IBE in the quantum random oracle model. In *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part II*, pages 253–282, 2018.

[LZ19] Qipeng Liu and Mark Zhandry. Revisiting post-quantum fiat-shamir. In *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*, pages 326–355, 2019.

[Nao03] Moni Naor. On cryptographic assumptions and challenges. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 96–109. Springer, 2003.

[NC16] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information (10th Anniversary edition).* Cambridge University Press, 2016.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93. ACM, 2005.

[TU16] Ehsan Ebrahimi Targhi and Dominique Unruh. Post-quantum security of the fujisaki-okamoto and OAEP transforms. In *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II*, pages 192–216, 2016.

[Wat18] John Watrous. *The Theory of Quantum Information.* Cambridge University Press, 2018.

[YZ21] Takashi Yamakawa and Mark Zhandry. Classical vs quantum random oracles. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part II*, volume 12697 of *Lecture Notes in Computer Science*, pages 568–597. Springer, 2021.

[Zha12] Mark Zhandry. Secure identity-based encryption in the quantum random oracle model. In *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 758–775, 2012.

[Zha19] Mark Zhandry. How to record quantum queries, and applications to quantum indifferentiability. In *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*, pages 239–268, 2019.