On the (Im)possibility of Time-Lock Puzzles in the Quantum Random Oracle Model

Abtin Afshar [*] Kai-Min Chung [†] Yao-Ching Hsieh [‡] Yao-Ting Lin [§]
Mohammad Mahmoody [¶]

Abstract

Time-lock puzzles wrap a solution s inside a puzzle P in such a way that "solving" P to find s requires significantly more time than generating the pair (s, P), even if the adversary has access to parallel computing; hence it can be thought of as sending a message s to the future. It is known [Mahmoody, Moran, Vadhan, Crypto'11] that when the source of hardness is only a random oracle, then any puzzle generator with n queries can be (efficiently) broken by an adversary in $O(n)$ *rounds* of queries to the oracle.

In this work, we revisit time-lock puzzles in a quantum world by allowing the parties to use quantum computing and, in particular, access the random oracle in quantum superposition. An interesting setting is when the puzzle generator is efficient and classical, while the solver (who might be an entity developed in the future) is quantum powered and is supposed to need a long sequential time to succeed. We prove that in this setting there is no construction of time-lock puzzles solely from quantum (accessible) random oracles. In particular, for any n-query classical puzzle generator, our attack only asks $O(n)$ (also classical) queries to the random oracle, even though it does indeed run in *quantum* polynomial time if the honest puzzle solver needs quantum computing.

Assuming perfect completeness, we also show how to make the above attack run in exactly n rounds while asking a total of $m \cdot n$ queries where m is the query complexity of the puzzle solver. This is indeed tight in the round complexity, as we also prove that a classical puzzle scheme of Mahmoody et al. is also secure against quantum solvers who ask $n - 1$ rounds of queries. In fact, even for the fully classical case, our attack quantitatively improves the total queries of the attack of Mahmoody et al. for the case of perfect completeness from $\Omega(mn \log n)$ to mn. Finally, assuming perfect completeness, we present an attack in the "dual" setting in which the puzzle generator is quantum while the solver is classical.

We then ask whether one can extend our *classical-query* attack to the *fully* quantum setting, in which both the puzzle generator and the solver could be quantum. We show a barrier for proving such results unconditionally. In particular, we show that if the folklore simulation conjecture, first formally stated by Aaronson and Ambainis [arXiv'2009] is *false*, then there is indeed a time-lock puzzle in the quantum random oracle model that cannot be broken by classical adversaries. This result improves the previous barrier of Austrin et. al [Crypto'22] about key agreements (that can have interactions in both directions) to time-lock puzzles (that only include unidirectional communication).

[*]University of Virginia, na6xg@virginia.edu
[†]Academia Sinica, kmchung@iis.sinica.edu.tw
[‡]University of Washington, ychsieh@cs.washington.edu
[§]UCSB, yao-ting_lin@ucsb.edu. Part of the work was done when working at Academia Sinica.
[¶]University of Virginia, mohammad@virginia.edu

Contents

1 Introduction

Time lock puzzles (TLPs) allow a puzzle generator Gen to *efficiently* generate a puzzle P for a solution s, in such a way that solving the puzzle P back into s would require *significantly more time*, even if the adversary uses multiple computers in parallel. TLPs allow "sending a message to the future" as they only allow "opening the envelope" P if a significant amount of time is spent by the solver.

The work of Rivest, Shamir, and Wagner [RSW96] both presented a construction of a time-lock puzzle and also presented applications of such primitives. Their construction was based on the assumption that repeated squaring of integers modulo RSA composites cannot be expedited even with parallel computing, unless one knows the factoring of the composite in which case they can expedite the process. Hence, the puzzle generator can find the solution by "solving the puzzle" through a shortcut, while others are forced to follow the sequential path. The work of [RSW96] also suggested using TLPs for other applications such as delayed digital cash payments, sealed-bid auctions and key escrow. Boneh and Naor [BN00] further showed the usefulness of such "sequential" primitives by defining and constructing *timed commitments* and showing their use for applications such as fair contract signing. More recently, time-lock puzzles have found more applications such as non-interactive non-malleable commitments [LPS17].

Despite their usefulness, it is still not known how to build TLPs based on more standard assumptions, and particularly based on "symmetric key" primitives. One might be tempted to use *inversion* of (say, exponentially hard) one-way functions as the process of solving puzzles. However, an adversary with k times parallel computing power can expedite the search process by a factor k through a careful splitting of the search space into k subspaces. Taking symmetric primitives to their extreme (idealized) form, one can ask whether *random oracles* can be used for constructing TLPs. The good thing about oracle models in general, and the random oracle model in particular, is that one can easily define information-theoretic notions of *time* based on the total number of queries asked to them, and also define the notion of *parallel time* based on the number of *rounds* of queries that an algorithm asks to the oracle. This means that asking, say, 10 queries in parallel to the oracle only counts as a single unit of (parallel) time.

The work of Mahmoody, Moran, and Vadhan [MMV11] proved a strong barrier against constructing TLPs from symmetric primitives by ruling out constructions that solely rely on random oracles. In particular, it was proved that if a puzzle generator asks only n queries to the random oracle and that the puzzle can be solved (honestly) with m oracle queries, then there is always a way to expedite the solving process to only $O(n)$ *rounds* of queries while the total number of queries is still $\mathrm{poly}(n, m)$. Note that the polynomial limit on the total number of queries is necessary to make such an attack interesting, as it is always possible to ask *all* of the (exponentially many) queries of oracle in one round and then solve the puzzle without any further queries. The attack of [MMV11] was in fact a polynomial *time* attack, though if one was willing to give up on that feature and only aim for polynomial *number of queries* (which still suffices for ruling out a ROM-based construction) they could achieve it in exactly n rounds as well.

Motivated by the developments in the area of *quantum* cryptography, in which some or all of the parties of a cryptosystem might access quantum computation, we revisit the barrier of constructing TLPs in the random oracle model. The extension of ROMs with quantum access was formally introduced in the work of Boneh et al. [BDF$^+$11]. Therefore, one can study the existence of TLPs in the *quantum* random oracle model in which either (or both) of the puzzle generator or the puzzle solver could access the random oracle in quantum superposition. This leads us to the main questions:

> *Can we construct time-lock puzzles from random oracles if either or both of the puzzle generator and puzzle solver have quantum access to the random oracle?*

Therefore, the question above deals with three settings: (1) only the puzzle solver is quantum, (2) only the puzzle generator is quantum, (3) both algorithms are quantum. In the first two settings, the puzzle is a classical string, as it needs to be output or read by a classical algorithm. In addition, although the first two settings are not "full-fledged quantum", we find them meaningful. Particularly, the first setting is quite natural, as it gives the extra quantum power to the *more power* entity; indeed, the puzzle generator is supposed to be the more efficient entity out of the two players of the scheme.

In a related work, Unruh [Unr14] formalizes the notion of timed-release encryption[1] in the quantum setting and

[1]This is closely related to our notion of time-lock puzzles, with the difference that the puzzle solution is given to the puzzle generator at the beginning.

shows how one can bootstrap time-lock puzzles to make them *revocable*; namely, before the puzzles are solved the solver can convince the generator, through interaction, that they are not going to find the solution after all.

1.1 Our Results

Our main result is an impossibility result that extends the result of [MMV11] to the setting in which the puzzle solver is allowed to use quantum power. In particular, we prove the following theorem.

Theorem 1.1 (Attacking classical-generator quantum-solver TLPs – Informally stated, see Theorem 3.1)**.** *Any time-lock puzzle in the random oracle model with a classical-query puzzle generator and a quantum-query puzzle solver can always be broken by an attacker who asks only $O(n)$ rounds of* classical *queries to the random oracle, where n is the total number of oracle queries of the puzzle generator. The total number of queries of the attack will be polynomial in n, m where m is the number of (potentially quantum) oracle queries of the honest solver.*

Note that it would be enough to find an attacker with quantum queries to the random oracle so long as the number of rounds of queries is polynomial, however, our theorem above goes one step further and shows that the quantum nature of the honest solver cannot be crucial, as there will always be an "equivalent" classical-query solver as well.

Complexity of our attack. Our vanilla attack is not efficient, but similarly to the work of [MMV11], we are also able to make our attack efficient, though our efficient attack will run in *quantum* polynomial time, as it will need to simulate the honest solver in its head. Note that our attack's parallel complexity is only measured by its number of rounds of queries to the oracle, and hence the (sequentially long) running time of the simulation of the honest adversary in the attacker's head is ignored in this regard. However, our result shows that basing the sequential soundness *solely* on the (quantum-accessible) random oracle is not going to be possible for constructing TLPs, even when we use extra computational assumptions. In particular, one can imagine potential TLP constructions in the ROM in which *quantum polynomial-time* adversaries cannot solve certain puzzles and would be forced to ask a large number of rounds of oracle queries. This would be similar to classical results such as [BR93] in which random oracles are used along with computational assumptions. The fact that our attack runs in quantum polynomial time rules out such approaches as well.

Optimal attacks on perfectly-complete protocols. The work of [MMV11] also presented an attack with exactly n rounds of queries. They also showed that such n-round attack is optimal, in general, by presenting an n-query puzzle generator that asks all of its queries in 1 round in such a way that it requires n rounds to be solved. The scheme is based on a construction that we refer to as a pseudo-chain. In this work, we also show that optimal-round attacks exist, at least for perfectly complete schemes.

Theorem 1.2 (Tight attack on classical-generator quantum-solver TLPs – Informally stated, see Theorem 4.10)**.** *Suppose Π is a TLP as in the setting of Theorem 1.1 and with perfect completeness. Then, Π can be broken in n rounds of queries and a total of $m \cdot n$ queries.*

We further observe that the n-round is indeed optimal by proving that the same pseudo-chain scheme of [MMV11] needs n rounds of queries by the solver, even if it can ask *quantum* queries to the random oracle.

Theorem 1.3 (A TLP with a linear difficulty gap for quantum-solver– Informally stated, see Theorem 5.3)**.** *Define the puzzle-generating function f to be $f^H(x_0, x_1, \ldots, x_{n+1}) := (x_0, H(x_0) \oplus x_1, \ldots, H(x_n) \oplus x_{n+1})$. Then any attacker that makes at most n rounds of quantum queries can find x_{n+1} with at most negligible probability.*

Breaking quantum-generator classical-solver puzzles. We then turn to the next setting, in which only the puzzle generator has quantum access to the random oracle. This setting is perhaps less natural for a TLP as the puzzle generator in a TLP is usually the less resource-intensive party. However, we still find this a natural setting, in part due to its *potential applications* beyond the TLP itself. In particular, if one can break quantum-generator classical-solver TLPs in a few rounds in the ROM, it would have corollaries to the round complexity of attacks on key agreements in the ROM as well. In fact, as observed in [MMV11], the transcript T of a key agreement can be seen as a puzzle generated by the two honest parties with the solution being the agreed key k. Then, an attacker who gets T and finds k can be seen as a puzzle solver. The work of [ACC+22] showed that under the so-called polynomial compatibility conjecture, any n query perfectly complete key agreement in the quantum ROM in which the messages are classical can be broken by $\mathrm{poly}(n)$ *classical* queries to the oracle. Hence, by interpreting the attack on the key agreement as a puzzle solver as explained above, an attack on TLPs with quantum generators and classical solvers would imply novel attacks on the key agreement protocols in the quantum ROM with $O(n)$ rounds of attack queries. In part, with such motivation in mind, we take a step towards proving such attacks on TLPs by proving the following result. Indeed we show a variant of the attack of Theorem 1.2 that works for the "opposite" direction of quantum-generator classical-solver TLPs.

Theorem 1.4 (Attacking quantum-generator classical-solver TLPs – Informally stated, see Theorem 4.6)**.** *Suppose* Π *is a TLP with perfect completeness in which the puzzle* generator *can ask* n *quantum queries to the random oracle, and the solver asks* m *classical queries to the random oracle. Then,* Π *can be broken in* n *rounds and a total of* $m \cdot n$ *oracle queries.*

Theorem 1.4 is also tight, in the sense that it asks exactly n rounds of queries. However, similarly to Theorem 1.2, it requires perfect completeness. This means that if one can improve the attack of [ACC+22] to find the key with probability 1 in $\mathrm{poly}(m, n)$ total number of classical queries, then our Theorem 1.4 would immediately imply a round-optimal n-round attack.

Quantum generators and solvers. Finally, we turn to the case in which both puzzle generator and puzzle solver are allowed to use quantum access to the random oracle model. Note that, in general, in this setting one can imagine the puzzle itself to be a quantum object. For this setting, we are not able to present an attack. On the contrary, we identify a barrier for proving such result *unconditionally*, so long as the attack only uses classical queries to the oracle, which is the case in all of our attacks above. In particular, we show that if the so-called (folklore) "simulation conjecture" [AA14] is *false*, then there exists a TLP with a quantum puzzle generator and a quantum solver that cannot be solved by classical-query attackers, even if they ask any polynomial number of queries to the oracle.

The simulation conjecture states that for any algorithm Q that asks n *quantum* queries to a random oracle H, the probability $p_H = \Pr[Q^H = 1]$ can be ε-approximated for $1 - \varepsilon$ fraction of the random oracles H, for arbitrarily small $\varepsilon = 1/\mathrm{poly}(n)$ using only $\mathrm{poly}(n)$ number of *classical* queries to H. When, $p_H \in \{0, 1\}$ for all H, then this conjecture is known to be true (e.g., see [OSSS05]), but the general case is open and has resisted to be solved for more than a decade after its official exposition by [AA14].

Theorem 1.5 (Barrier for classically attacking fully quantum TLPs – Informally stated, see Theorem 6.2)**.** *If the simulation conjecture is false, then there is a time lock puzzle in the quantum random oracle model that cannot be broken by classical adversaries.*

Our theorem above does not lead to an actual useful TLP in the quantum random oracle model, as it does not even offer a meaningful gap between the (true) running time of the honest solver and that of the puzzle generator. However, it has the crucial property that a classical (potentially malicious) solver is at a full disadvantage and cannot solve the puzzle even with an arbitrary polynomial number of queries, regardless of its parallel complexity. Consequently, it only serves as a barrier for extending Theorem 1.1 using a classical attack. Having said that, it is certainly possible that Theorem 1.1 could potentially be extended to the fully quantum setting using a quantum adversary. We leave this as an intriguing open question.

1.2 Technical Overview

Here we describe some of the ideas behind the proofs of our Theorems 1.1 and 1.5. We start by describing our ideas behind the attack of Theorem 1.1.

Ideas behind our attacks of Theorem 1.1. Our starting points are the attacks (from the full version of) [MMV11], in which two different types of attacks on TLPs are presented: (1) computationally unbounded attacks with tight adaptivity n[2] and (2) poly-time attacks with adaptivity $O(n)$ (specifically as small as $2n$ rounds). When we move to

[2]In fact, [MMV11] also showed that n-adaptivity is the best one can hope for, as there is a matching positive construction.

the quantum-solver setting, we observe that their proofs suffer from incomparable issues: the first attack primarily picks its oracle queries based on the puzzle generator's algorithm, which for us is also classical, and hence the attack description is more "quantum-solver friendly". However, the *analysis* of this attack goes through conditioning on events that do not have a direct quantum variant (at least at first sight). The second attack from [MMV11] is in an opposite form: the attack's description seems to heavily rely on the solver being classical (more on this below) and hence seems to be a more challenging path to take for us here. Having said that, we show that it is indeed possible to extend a variant of the second attack to the quantum-solver setting by developing new ideas that might be useful for other contexts as well. As a bonus, we are also able to make the attack efficient (in quantum polynomial time).

The classical attack of [MMV11]. We start by describing the attack of [MMV11] at a high level, and then we address the challenges that arise when we move to the quantum solver setting and explain how to address them. Suppose the Gen makes n oracle queries. Then, the attacker will do the following in $3n$ rounds (of oracle queries), while in each round it learns more oracle queries encoded in a list/partial function $\mathcal{L}$ (that grows over time). In round i, the adversary will pick a full execution of the puzzle solver $\mathsf{Sol}^{H_i}(\mathrm{P}) \to \mathrm{s}_i$ over the given puzzle P to obtain the solution s_i, while H_i is a "random oracle" sampled in adversary's head conditioned on the learned list $\mathcal{L}$. Then, at the end of this round, the adversary will ask *all* of the queries asked by $\mathsf{Sol}^{H_i}(\mathrm{P})$ from the real oracle H and adds this information to $\mathcal{L}$. Let $\mathcal{Q}_i$ be the queries asked in this round. The key observation is that during the round by round executions that the adversary makes and learns more and more about the true oracle H (through the list $\mathcal{L}$) only in n out of those $3n$ executions the list $\mathcal{Q}_i$ can have a *new* intersection with the solver's queries $\mathcal{Q}$. So, in *most* of these executions, no such intersections exist, in which case the adversary gets a *perfectly consistent* execution of the random oracle that would be guaranteed to have the same completeness error as that of the honest solver. Therefore, in most of the rounds, the adversary finds the true answer s to the puzzle P.

Challenge: quantum queries do not co-exist. When we move to the quantum-solver setting, the adversary still can sample a full execution $\mathsf{Sol}^{H_i}(\mathrm{P})$ conditioned on a classical list of query-answer pairs $\mathcal{L}$ that it has previously learned about the true oracle H by sampling H_i conditioned on $\mathcal{L}$ in its head and running the honest solver relative to H_i. This is despite the fact that Sol is now a *quantum* algorithm, yet this is all happening in the adversary's head. However, the real challenge is to go one step further: how can the adversary *extract* the set of queries that Sol^{H_i} is asking from its oracle H_i and ask them from the real oracle? After all, it is well-known that when we move to the quantum setting, then multiple queries asked by the same algorithm Sol might *not* coexist.

Idea: learning amplitude-heavy queries, in parallel. Our main idea for resolving the challenge above is to define a classical set of queries that are well-defined to coexist and can be learned from the real oracle *in parallel*, while at the same time, these queries will provide useful information for the attacker to succeed. In particular, for every quantum execution of Sol^{H_i}, we say that a *classical* query q is ε-amplitude heavy, if there is one of the quantum queries $\tilde{q}$ (out of the m quantum queries) of Sol^{H_i} such that, the classical query q has at least amplitude ε within $\tilde{q}$. It is easy to see that the number of such queries cannot be more than $\mathrm{poly}(m/\varepsilon)$ for every execution Sol^{H_i}, and therefore, the set $\mathcal{HQ}^i_\varepsilon$ of all ε-amplitude heavy queries of this execution will constitute a sufficiently small (i.e., polynomial-size) classical set. Our adversary will then ask all of the (classical) queries inside the ε-amplitude heavy queries set $\mathcal{HQ}^i_\varepsilon$ at the end of each round i. One can now show that if the number of rounds of the attack is, say $3n$, then again in most of the attack rounds, the attacker's heavy set of queries (that it also learns at the end of the round) will not have a new collision with the puzzle generator's set of queries Q. It remains to show that this condition will again guarantee that the attacker will have a good chance of finding the true puzzle solution s in such rounds.

Final touch: one-way to hiding. Suppose the adversary has already learned a set $\mathcal{L}$ of oracle query-answer pairs about the real oracle H, and that it makes a simulation Sol^{H_i} to obtain a candidate solution s_i by sampling the oracle H_i at random conditioned on $\mathcal{L}$. Further, suppose that we somehow have the guarantee that *none* of the ε-amplitude heavy (classical) queries of the (quantum) execution Sol^{H_i} will intersect with any of the puzzle generator's queries. In this case, we show how to use the useful one-way to hiding lemma of [AHU19], and show that the execution Sol^{H_i} will be statistically close to a *perfectly consistent* execution, in which one *also conditions* on the query-answer pairs of the puzzle generator. As a result, this execution by the adversary Sol^{H_i} would have (almost) the same chance to find the true answer, as an honest solver would.

Putting things together, this means that our attack would also succeed in finding the true answer in most of the rounds out of the $3n$ rounds of the attack, and hence it can e.g., take the majority of the obtained solutions to find the correct solution.

Learning amplitude-heavy queries (parallel) efficiently. In order to make the attack efficient, the following issues need to be addressed. First, Eve cannot sample a full (exponential size) oracle to run the solver. To efficiently simulate a partially fixed quantum random oracle, we instead use $2m$-wise independent functions [Zha12], where m is the query complexity of the solver. Next, in each round, Eve cannot compute exponentially many query amplitudes of the simulated puzzle solver. Instead, we "experimentally extract" such amplitude-heavy queries. To do so, we rely on the "original version" of one-way to hiding lemma in [Unr14]. Intuitively, if a query q is of high amplitude, then running the solver until a random point and measuring the query register will output q with sufficiently large probability. Following this idea, we run such extractions many times in each *single round*, before extracting a long list of classical queries and asking them from the oracle. This leads to a quantum time-efficient algorithm to extract (almost all) ε-amplitude heavy queries, which turns out to be sufficient for the proof.

Ideas behind the proofs of the tight attacks of Theorems 1.2 and 1.4. Our attack is inspired by the classical result [BI87, Nis89, Tar89, HH87] that $D(f) \leq C_0(f) \cdot C_1(f)$, in which $f \colon \{0,1\}^N \to \{0,1\}$ is an arbitrary boolean function, D is the decision-tree complexity of f (i.e., the smallest number of adaptive queries to the N input bits to f to determine its output), and $C_b(f)$ for $b \in \{0,1\}$ is the b-certificate complexity of f (i.e., the smallest number of input bits that would provably reveal the output to be b). To see the connection, roughly speaking one has to think of the random oracle of N queries as a giant input $x = (x_1, \ldots, x_N)$ and the puzzle solution as the "output" of the function $f(x)$. (Of course the puzzle generation is randomized, but in this simplified exposition we intentionally ignore this fact.) Below we describe our ideas first for the purely classically case, in which we quantitatively improve the query-complexity of the attack of [MMV11], and then explain the additional ideas for the quantum case.

We define two specific forms of certificates for the puzzle solution being 0 or 1. Suppose P is a puzzle and $\mathrm{s} \in \{0,1\}$ is its solution. Consider a full execution of the puzzle generator, using the real oracle H, in which it contains the query-answer pairs $\mathcal{P}_G$ about the oracle H and at the end outputs (P, s). In this case, $\mathcal{P}_G$ serves as a certificate for s, because no solver can obtain the opposite answer $1 - \mathrm{s}$ with respect to any oracle that is consistent with $\mathcal{P}_G$; this holds due to the perfect completeness of the scheme. Similarly, for any execution of the solver that solves P into solution s and that it contains query-answer pairs $\mathcal{P}_S$ about the oracle H, it holds that $\mathcal{P}_S$ serves as a certificate for the solution being s for a similar reason. We will limit ourselves to use only $\mathcal{P}_G$ for $\mathrm{s} = 0$ certificates and $\mathcal{P}_S$ for $\mathrm{s} = 1$ certificates.

Now, a key observation is that, in case of perfectly complete TLPs, any b-certificate, as a set of query-answer pairs, shall be *inconsistent* with any $(1 - b)$-certificate, in that they contain a similar query with different answers, because otherwise one can extend them to a full oracle in which the solver finds a wrong solution. Therefore, this leads to the following n-round attack of total mn query complexity: so long as the puzzle solution is not determined by what the attacker knows (which includes the puzzle P), the attacker would pick a 1-certificate S for *some* oracle that is consistent with attacker's knowledge about the random oracle, and then it asks all of the queries Q in S from the real oracle. The key idea is that, if the answer is still not defined, it at least holds that the effective size of any 0-certificate (among the remaining unknown queries of the oracle) will shrink by one or more. Hence, the process stops in n rounds.

Quantum solver or quantum generator. For the case that either of the puzzle generator or the puzzle solver are quantum, we further use inspiration from the attack on key-agreement from random oracles in [ACC+22] for the case that one party is quantum and the other one is classical. In their work, they show how to associate a degree-d polynomial over variables $x_1 \ldots x_N \in \{\pm 1\}$ (where the random oracle has N queries) to the quantum state of the quantum-party conditioned on the transcript of the scheme, in which that party asks d quantum queries to the random oracle. The key idea, at a high level, that generalizes the fully-classical case described above is to replace the query-set $\mathcal{Q}_P$ of a classical party by (who is now going to be a quantum algorithm) by the set of oracle queries (that correspond to variables $x_i \in \{x_1 \ldots, x_N\}$) that in turn correspond to a *maximal monomial* in the polynomial that encodes the quantum state of the quantum party (conditioned on the transcript). Then, we show that the same key idea about the inconsistency (and hence non-empty intersections of the query sets) of the 0-certificates and 1-certificates still carries to this generalized setting. This allows us to obtain a classical attack with the same round and query complexity as the fully classical case. For the case of classical puzzle solvers, the attack indeed remains the same exact (round-optimal) attack as that of fully classical case. For the case of quantum puzzle solvers, the attacker will pick maximal monomials in the polynomial representing the solver's quantum state and ask all of the queries in that monomial in each round.

Ideas behind the proof of Theorem 1.3. Informally, the "knowledge depth" of the attacker after i rounds of queries is at x_i. Therefore, in order for the attacker to solve the puzzle better than randomly guessing x_n, the attacker must make a lucky guess. That is, there exists some $i \in [n]$ such that the attacker's queries in the i-th round hit any of $\{x_i, x_{i+1}, \ldots, x_q\}$. We use hybrid arguments to gradually collect the probabilities that the attacker makes a lucky guess in every round. Finally, we show that the attacker can only randomly guess x_n given there is no lucky guess. In particular, we design hybrids carefully in such a way that (1) in the first hybrid, the game refers to the actual real attack, (2) the last hybrid is trivially information theoretically hard to attack, as the adversary would have to guess a random string of length κ independent of its view, and (3) the neighboring hybrids are computationally indistinguishable up to certain bounds due to a standard argument. The latter relies on the fact [BBBV97, AHU19] that any quantum algorithm that makes 1 round of k-parallel quantum queries cannot distinguish whether a random oracle has been reprogrammed on S random points with advantage more than $O(\sqrt{k \cdot S/2^\kappa})$, where κ is the input length of the random oracle.

Ideas behind the barrier of Theorem 1.5. We now describe ideas behind our barrier of Theorem 1.5 against breaking fully TLPs classically. In fact, we prove something more general about key-agreements in the (quantum) random oracle model resisting classical attacks. Namely, we show that if the simulation conjecture if *false*, then there is a way for two (quantum) parties Alice and Bob to agree on a classical (random) key by calling a random oracle H in *quantum* superposition and Alice sending a single classical message c to Bob with the following property. Any computationally unbounded Eve who observes Alice's classical message and can query the random oracle H in $\mathrm{poly}(\kappa)$ points *classically* has a negligible chance of finding this key. If we can construct such a protocol, it would be easy to use it to build a TLP on top of it by having Alice hide the puzzle solution using the key and send $\mathrm{s} \oplus$ key along with the transcript (single message) c as the puzzle's description. The solver will then use Bob's algorithm to find the key, and then uncover the puzzle solution s. This protocol has the property that any classical-query adversary will have a negligible chance of solving it, establishing Theorem 1.5. Previously, [ACC$^+$22] showed how to assume the simulation conjecture is false to build an *interactive* key-agreement between quantum-powered parties in the quantum-random oracle model that was secure against polynomial-classical-query adversaries. Here, in this work, we improve their result to obtain such a protocol only with one-way communication.

Idea: classically-secure quantum key-agreement with one-way communication. At a high level, we use ideas from [ACC$^+$22] to prove a similar result, but their result ended up with an *interactive* key agreement protocol. Here, it is crucial for us to obtain a *non-interactive key-agreement* protocol which we can then use to obtain a time-lock puzzle. We follow the path of [ACC$^+$22] initially, but then the two papers diverge as follows. [ACC$^+$22] relies on a result from [HMST22] that allows how to leverage interaction to bootstrap the obtained key agreement into one with negligible soundness and completeness error. The reduction from [HMST22] is *computational*, which is a stronger reduction but comes at the cost of interaction. On the other hand, we leverage on the *information-theoretic* nature of the security of our key agreement and the fact [ACC$^+$22] already provides a weakly secure key-agreement protocol with one-way communication that has sufficiently small completeness error.

Outline of the technical steps for Theorem 1.5. To construct a classically-secure quantum key-agreement with one-way communication we go through the following steps.

Step 0 We start off with the weak key agreement protocol from [ACC$^+$22] (based on the simulation conjecture being false) in which Alice and Bob have quantum access to the random oracle, Alice sends a single message to Bob, and they have completeness error $\varepsilon = 1/\mathrm{poly}(\kappa)$ and the soundness error is $\delta = \mathrm{negl}(\kappa)$ against any adversary with a polynomial number of classical queries. As mentioned above, this completeness error $\varepsilon = 1/\operatorname{poly}(\kappa)$ can be made arbitrarily small.

Step 1 Using Goldreich-Levin's hard-core bit lemma, we turn the above protocol into a new one in which Alice and Bob agree on a *bit* that remains indistinguishable from random, even if the adversary is given the transcript.

Step 2 Using parallel repetition, we increase the length of the key. An interesting subtle point here is that we cannot rely on the vanilla hybrid arguments to argue that this parallel repetition will increase the length of the key securely; that is because the adversary's complexity class is different from that of honest players (who will be simulated as part of the hybrid argument). Despite this subtlety, since our security notion here is *information theoretic*, the hybrid argument still goes through.

Step 3 The completeness error in all the steps above can be kept $\varepsilon' < 1/\operatorname{poly}(\kappa) < 1/2$. We finally make a final change to make the completeness error also negligible: we use another parallel repetition, and this time we let Alice pick a random key key and send $\mathsf{key} \oplus \mathsf{key}_i$ for various keys that she agrees with Bob. Bob then recovers key in most of these executions and finds key by taking a majority.

1.3 Related Work

The study of so-called sequential primitives is not limited to time-lock puzzles. A closely related primitive is a *proof of sequential work* (PoSW) [MMV13, CP18]. A PoSW also has a challenger (who takes the role of the puzzle generator) and a solver, but the solver's answer back to the challenger could be *not unique*, while its validity should be efficiently verifiable. Hence, one can interpret PoSW as a TLP with more than one possible solution, and whose exact solutions might *not* be even known to the challenger at the time of generating the challenge to the solver/prover. The works of [CFHL21, BLZ21] showed that PoSW *can* be achieved in a post-quantum world from random oracles in a strong sense that stands at the opposite of our negative result of Theorem 1.1: there is a TLP in which the protocol for the honest parties is classical, while its soundness holds against adversaries who are quantum powered.

Another closely related primitive to TLPs the primitive of *verifiable delay functions* [BBBF18, LW17, Pie19, Wes19]. A VDF is similar to a time-lock puzzle, in the sense that the challenger generates a puzzle/challenge with a unique solution s, but s might not be known to the challenger during the time of generation. Yet, when the solver solves the challenge, it can *prove* the validity of its answer through a verification process that is quite fast, just like the puzzle generation phase. It was shown [MSW20] that certain special forms of VDFs are impossible to achieve in the random oracle model. It is possible that using (a generalization of) our techniques one could extend our impossibility results to such classes of VDFs as well, though we leave this exploration for the full version of this paper. See [JMRR21] for a comprehensive study of the relation between the primitives above and other closely related sequential primitives.

Our work can be seen as continuing the line of work initiated by [HY20] for proving *quantum black-box separations*. Their work proved that collision-resistant hash functions cannot be based on trapdoor permutations, even through a quantum reduction. Even though we do not prove any such results explicitly, we present attacks in the random oracle model, which can provide many computational primitives for free, and so our work would also imply corresponding black-box separations. See [AHY23, AK22] for more recent works on barriers against black-box quantum constructions.

2 Preliminaries

2.1 Basic Notations

Let $\kappa \in \mathbb{N}$ denote the security parameter. For $n \in \mathbb{N}$, let $[n]$ to denote the set $\{1, 2, \ldots, n\}$. Let U_n be a random variable that returns a random string of length n. By $\operatorname{negl}(\cdot)$ we denote a negligible function. For a finite set $\mathcal{S}$, by $x \overset{\$}{\leftarrow} \mathcal{S}$ we mean x is chosen uniformly from $\mathcal{S}$. Throughout this work, we use the standard bra-ket notation for quantum objects. For the basics of quantum computing, we refer the readers to [NC10]. By $\|\cdot\|$ we denote the 2-norm.

2.2 Quantum Computation

Definition 2.1 (Quantum Oracle Algorithm)**.** A q-query quantum oracle algorithm $\mathcal{A}^{(\cdot)}$ that has access to an oracle H, defined by the unitary O_H can be specified by a sequence of unitaries $(U_q, U_{q-1}, \ldots, U_0)$. The final state of the algorithm is defined as $U_q O_H U_{q-1} \ldots O_H U_0 |0\rangle$.

The query operator O_H is defined as $O_H|x, y\rangle \mapsto |x, y \oplus H(x)\rangle$, where we refer to the first register as the *query register*. $\Diamond$

Definition 2.2 (Query Amplitude [BBBV97])**.** Let $\mathcal{A}^{(\cdot)} = (U_q, \ldots, U_0)$ be a quantum oracle algorithm, $H : \mathcal{X} \to \mathcal{Y}$ be an oracle and $\mathcal{S} \subseteq \mathcal{X}$ be a set. The *query amplitude* $\mu(\mathcal{A}^H, \mathcal{S}) \in \mathbb{R}_{\geq 0}$ is defined as

$$\mu(\mathcal{A}^H, \mathcal{S}) \coloneqq \sum_{i=0}^{q-1} \|\Pi_{\mathcal{S}} |\psi_i^H\rangle\|^2,$$

where $\Pi_{\mathcal{S}}$ is the projector onto $\mathcal{S}$ acting on the query register of $\mathcal{A}$; each $|\psi_i^H\rangle$ denotes the state of the algorithm right before the $(i+1)^{\text{th}}$ query, i.e., $|\psi_i^H\rangle \coloneqq U_i O_H \ldots O_H U_0 |0\rangle$. $\Diamond$

If $\mathcal{S}$ is a singleton, i.e., $\{x\}$, by a slight abuse of the notation we denote it as $\mu(\mathcal{A}^H, x)$. Note that $\mu(\mathcal{A}^H, \mathcal{S})$ is at most the number of queries made by $\mathcal{A}$. Moreover, for any disjoint $\mathcal{S}_1, \mathcal{S}_2 \subseteq \mathcal{X}$, it holds that $\mu(\mathcal{A}^H, \mathcal{S}_1) + \mu(\mathcal{A}^H, \mathcal{S}_2) = \mu(\mathcal{A}^H, \mathcal{S}_1 \cup \mathcal{S}_2)$.

The following lemma is restated from a similar result in [Unr14].

Lemma 2.3 ([Unr14])**.** *For every (fixed) $H : \mathcal{X} \to \mathcal{Y}$, every (fixed) $\mathcal{S} \subseteq \mathcal{X}$, every (fixed) $z \in \{0,1\}^*$, and every quantum oracle algorithm $\mathcal{A}^{(\cdot)}$, there exists a quantum oracle algorithm $\mathsf{Ext}(\mathcal{A}^H(z))$ that uses $\mathcal{A}^{(\cdot)}$ as subroutine and outputs an element in $\mathcal{S}$ with probability $\mu(\mathcal{A}^H(z), \mathcal{S})/q$, where q is the number of queries made by $\mathcal{A}^H(z)$.*

For completeness, the proof of Lemma 2.3 and the explicit description of the extractor Ext can be found in Section A.

Lemma 2.4 (One-way-to-hiding, Theorem 3 in [AHU19], restated)**.** *Let $\mathcal{S} \subseteq \mathcal{X}$ be random. Let $F, G : \mathcal{X} \to \mathcal{Y}$ be random functions satisfying $\forall x \notin \mathcal{S},\ F(x) = G(x)$. Let z be a random bitstring. ($\mathcal{S}, F, G, z$ may have arbitrary joint distribution.) Let $\mathcal{A}^{(\cdot)}$ be a q-query quantum oracle algorithm. Let*

$$\begin{aligned} P_{\mathsf{left}} &\coloneqq \Pr[b = 1 : b \leftarrow \mathcal{A}^F(z)], \\ P_{\mathsf{right}} &\coloneqq \Pr[b = 1 : b \leftarrow \mathcal{A}^G(z)], \\ P_{\mathsf{guess}} &\coloneqq \Pr[x \in \mathcal{S} : x \leftarrow \mathsf{Ext}(\mathcal{A}^F(z))]. \end{aligned}$$

Then

$$|P_{\mathsf{left}} - P_{\mathsf{right}}| \leq 2q\sqrt{P_{\mathsf{guess}}}.$$

The same holds with $\mathsf{Ext}(\mathcal{A}^G(z))$ instead of $\mathsf{Ext}(\mathcal{A}^F(z))$ in the definition of P_{guess}.

Combining Lemma 2.3 and Lemma 2.4, we obtain the following corollary.

Corollary 2.5. *Let $\mathcal{S} \subseteq \mathcal{X}$ be random. Let $F, G : \mathcal{X} \to \mathcal{Y}$ be random functions satisfying $\forall x \notin \mathcal{S},\ F(x) = G(x)$. Let z be a random bitstring. ($\mathcal{S}, F, G, z$ may have arbitrary joint distribution.) Let $\mathcal{A}^{(\cdot)}$ be a q-query quantum oracle algorithm. Then*

$$|P_{\mathsf{left}} - P_{\mathsf{right}}| \leq 2\sqrt{q\,\mathbb{E}\,[\mu(\mathcal{A}^F(z), \mathcal{S})]},$$

where the probability of the expectation is over $\mathcal{S}, F, G$ and z.

Proof. Note that the probability P_{guess} can be written as

$$\begin{aligned} P_{\mathsf{guess}} &= \Pr\left[x \in \mathcal{S} : x \leftarrow \mathsf{Ext}(\mathcal{A}^F(z))\right] \\ &= \mathop{\mathbb{E}}_{s,f,g,z'}\left[\Pr\left[x \in \mathcal{S} \mid \mathcal{S} = s, F = f, G = g, z = z' : x \leftarrow \mathsf{Ext}(\mathcal{A}^F(z))\right]\right] \\ &= \mathop{\mathbb{E}}_{\mathcal{S},F,G,z}\left[\frac{\mu(\mathcal{A}^F(z), \mathcal{S})}{q}\right], \end{aligned}$$

where the last equality follows from Lemma 2.3. Plugging P_{guess} into Lemma 2.4 finishes the proof. $\square$

2.3 Time-Lock Puzzles in the Random Oracle Model

The following definition of time-lock puzzles in the quantum world focuses on a classical puzzle generator, while the solver and the adversary are both quantum. We only allow a constant blow-up in the parallel complexity of the attacker, compared to that of the honest solver.

Definition 2.6 (Time-Lock Puzzle with Quantum Solver)**.** A time-lock puzzle scheme consists of a randomized oracle algorithm *puzzle generator* $(\mathrm{P}, \mathrm{s}) \leftarrow \mathsf{Gen}^H(r_G)$ (where P is the puzzle and s is the correct solution), and a quantum oracle algorithm *puzzle solver* $\mathrm{s}' \leftarrow \mathsf{Sol}^H(\mathrm{P})$ (where s' is the solution that the solver finds). For a puzzle generator who makes at most n *classical* queries and a puzzle solver who makes at most m ($m \gg n$) *quantum* queries to the oracle, we expect the following properties:

- **Completeness:** If the honest solver Sol finds the correct solution with probability $1-\rho$ as follows
$$\Pr[\mathrm{s} = \mathrm{s}' : (\mathrm{P}, \mathrm{s}) \leftarrow \mathsf{Gen}^H(r_G), \mathrm{s}' \leftarrow \mathsf{Sol}^H(\mathrm{P})] \geq 1 - \rho,$$
we say the scheme has ρ completeness error. By default, we anticipate the completeness error to be negligible $\rho \leq \mathrm{negl}(\kappa)$.

- **Soundness:** We say the scheme has σ soundness error, if for every quantum oracle algorithm E that makes $\mathrm{poly}(\kappa)$ queries to the oracle H in $O(n)$ rounds,
$$\Pr[\mathrm{s} = \mathrm{s}' : (\mathrm{P}, \mathrm{s}) \leftarrow \mathsf{Gen}^H(r_G), \mathrm{s}' \leftarrow \mathsf{E}^H(\mathrm{P})] = \sigma.$$
By default, we anticipate the scheme to have $\sigma \leq \mathrm{negl}(\kappa)$ soundness error.

$\diamondsuit$

3 Attacks on Time-Lock Puzzles with Classical Puzzle Generators

In this section, we present attacks for time-lock puzzles with classical generators and quantum solvers (CGQS). The attacks can be seen as a quantum extension of the "efficient but non-optimal adversary" in the full version of [MMV11]. We provide two attacks: one is query-efficient and one is (quantum) time-efficient, but they are incomparable as they provide different trade-offs between parameters.

3.1 Inefficient Attacks on CGQS Time-Lock Puzzles

We now state our result about breaking classical-generator quantum-solver (CGQS) time-lock puzzles in the (quantum) random oracle model. In order to keep the notation clean, for the rest of the work, we sometimes omit the ceiling function of parameters and treat it as an integer when it is clear from the context.

Theorem 3.1 (Breaking CGQS Time-Lock Puzzles)**.** *Consider any time-lock puzzle scheme in the random oracle model where the generator asks n classical queries, the solver asks m quantum queries and the completeness is $1-\rho$ (see Definition 2.6). For any $\varepsilon, \delta \in (0, 1]$, there exists a randomized solver Eve (denoted by E) who asks at most q_E classical queries in at most d_E rounds and achieves the following bounds for the completeness error $\nu = 1 - \Pr[\mathrm{s} = \mathrm{s}' : (\mathrm{P}, \mathrm{s}) \leftarrow \mathsf{Gen}^H(r_G), \mathrm{s}' \leftarrow \mathsf{E}^H(\mathrm{P})]$, where the probability is over the randomness of the generator, the randomness of Eve and the random oracle H.*

1. *Small completeness error: Eve asks $q_E = \frac{4n^2m^2}{\varepsilon\delta^2}$ queries in $d_E = n/\varepsilon$ rounds with completeness error $\nu \leq \rho + \varepsilon + \delta$.*
2. *$2n$ adaptivity: Eve asks $q_E = \frac{8n^2m^2}{\delta^2}$ queries in $d_E = 2n$ rounds with completeness error $\nu \leq (2n+1)(\rho + \delta)$.*

Proof of Theorem 3.1. We first prove Part 1 of Theorem 3.1. Then we will obtain Part 2 through a modified attack and a different analysis. We first introduce some notations.

Notation. Let $\mathcal{Q}_G$ be the set of points queried by Gen^H for generating the puzzle and the solution. Let $\mathcal{P}_G$ be the set of query-answer pairs learned by Gen^H, i.e., $\mathcal{P}_G = \{(q, H(q)) \mid q \in \mathcal{Q}_G\}$. Sometimes we abuse the notation and use $(\mathcal{Q}_G, H(\mathcal{Q}_G))$ to denote the same thing ($\mathcal{P}_G$). For a partial function $F : \mathcal{S} \to \mathcal{Y}$ where $\mathcal{S}$ is a subset of the domain $\mathcal{X}$, by $\mathcal{Y}^{\mathcal{X}}|_F$ we mean the set of functions that are consistent with F. Sometimes we refer to a partial function $\mathcal{L}$ as a *list* and use $\mathcal{Q}(\mathcal{L})$ to denote its domain.

Consider the following construction:

Construction 3.2 (Random Stoppage Attack)**.** Let n, m be, in order, the number of queries made by Gen^H and Sol^H.

- Input: the puzzle P generated by Gen^H and parameters $\varepsilon, \delta \in (0, 1]$.
- Set $I = n/\varepsilon$ and initialize a list $\mathcal{L}_1 = \emptyset$.
- Pick $i^* \xleftarrow{\$} [I]$.
- For $i \in \{1, \ldots, i^*\}$, run $\mathsf{Solve}(i)$.
- Output s_{i^*}.

In the i^{th} round, the subroutine $\mathsf{Solve}(i)$ is defined in Algorithm 3.3. We note that Construction 3.2 should be understood by syntactically replacing $\mathsf{Solve}(i)$ with the code of $\mathsf{Solve}(i)$. Looking ahead, $\mathsf{Solve}(i)$ will appear again in the rest of the proof.

Algorithm 3.3. $\mathsf{Solve}(i)$:

- Set $\mathcal{Q}_i = \emptyset$.
- Sample a full oracle $H_i \xleftarrow{\$} \mathcal{Y}^{\mathcal{X}}|_{\mathcal{L}_i}$.
- Run $\mathsf{Sol}^{H_i}(\mathrm{P})$ to get the solution s_i.
- For every $x \in \mathcal{X} \setminus \mathcal{Q}(\mathcal{L}_i)$, compute the query amplitude $\mu(\mathsf{Sol}^{H_i}(\mathrm{P}), x)$.
- For every $x \in \mathcal{X} \setminus \mathcal{Q}(\mathcal{L}_i)$, if $\mu(\mathsf{Sol}^{H_i}(\mathrm{P}), x) \geq \delta' := \frac{\delta^2}{4nm}$, then let $\mathcal{Q}_i \leftarrow \mathcal{Q}_i \cup \{x\}$.
- Ask the classical queries $\mathcal{Q}_i$ from the real oracle to obtain $H(\mathcal{Q}_i)$.
- Update the list by $\mathcal{L}_{i+1} \leftarrow \mathcal{L}_i \cup (\mathcal{Q}_i, H(\mathcal{Q}_i))$.

Efficiency. In each round, since the number of queries made by Sol is at most m, there are at most $\frac{m}{\delta'} = \frac{4nm^2}{\delta^2}$ points that can be added into $\mathcal{Q}_i$. Moreover, there are at most $I = \frac{n}{\varepsilon}$ rounds. Therefore, Eve's adaptivity is at most $\frac{n}{\varepsilon}$ and the total number of Eve's queries is at most $\frac{4n^2m^2}{\varepsilon\delta^2}$.

Completeness. For analysis, we introduce the following experiment which is identical to Construction 3.2 except that it does not stop at a random point.

Experiment 3.4. Let n, m be, in order, the number of queries made by Gen^H and Sol^H.

1. Input: the puzzle P generated by Gen^H and parameters $\varepsilon, \delta \in (0, 1]$.
2. Set $I = n/\varepsilon$ and initialize a list $\mathcal{L}_1 = \emptyset$.
3. For $i \in [I]$, run $\mathsf{Solve}(i)$.
4. Output s_I.

Before proving the completeness, we introduce some notations for the above experiment. By *private queries* we mean the set $\mathcal{Q}_G \setminus \mathcal{Q}(\mathcal{L}_i)$, i.e., the set of queries that are asked by the puzzle generator but not asked by Eve before the i^{th} round. For an oracle $H : \mathcal{X} \to \mathcal{Y}$ and a set $\mathcal{S} \subseteq \mathcal{X}$, by $H[\mathcal{S}^{\circlearrowleft}]$ we mean the oracle that is obtained by resampling the function value of every point in $\mathcal{S}$ uniformly from $\mathcal{Y}$.
For every $i \in [I]$, define the event Heavy_i as

$$\mathsf{Heavy}_i \coloneqq \left[\mu(\mathsf{Sol}^{H_i}(\mathrm{P}), \mathcal{Q}_G \setminus \mathcal{Q}(\mathcal{L}_i)) \geq n\delta'\right], \tag{1}$$

i.e., the query amplitude of $\mathsf{Sol}^{H_i}(\mathrm{P})$ on the private queries is at least $n\delta'$ in the i^{th} round. We say that the i^{th} round is *heavy* if the event Heavy_i holds.

For every $i \in [I]$, define the event Find_i as

$$\mathsf{Find}_i \coloneqq [\mathcal{Q}_i \cap (\mathcal{Q}_G \setminus \mathcal{Q}(\mathcal{L}_i)) \neq \emptyset],$$

i.e., Eve learns at least one of the private queries in the i^{th} round.

First, the following lemma states that the occurrence of Heavy_i implies that Eve learns some private query in the i^{th} round.

Lemma 3.5. *For every $i \in [I]$,*

$$\Pr[\mathsf{Find}_i \mid \mathsf{Heavy}_i] = 1.$$

Equivalently, $\mathsf{Heavy}_i \subseteq \mathsf{Find}_i$.

Proof. Conditioned on the event Heavy_i happening, it means that the query amplitude of $\mathsf{Sol}^{H_i}(\mathrm{P})$ on the private queries $\mathcal{Q}_G \setminus \mathcal{Q}(\mathcal{L}_i)$ is at least $n\delta'$ in the i^{th} round. Since the number of queries made by Gen is at most n, by an averaging argument, there exists $x \in \mathcal{Q}_G \setminus \mathcal{Q}(\mathcal{L}_i)$ such that $\mu(\mathsf{Sol}^{H_i}(\mathrm{P}), x) \geq \delta'$. By construction, the point x will be added into $\mathcal{Q}_i$. Thus, Eve learns at least one of the private queries in the i^{th} round. □

Next, since the puzzle generator asks at most n queries, the number of occurring events among $\{\mathsf{Find}_i\}_{i\in[I]}$ is at most n in every execution.

Lemma 3.6. *For every $i \in [I]$, let I_{Find_i} be the indicator variable of* Find_i. *Then*

$$\Pr\left[\sum_{i\in[I]} I_{\mathsf{Find}_i} \leq n\right] = 1.$$

Proof. For the sake of contradiction, suppose there is a nonzero probability that all events $\mathsf{Find}_{i_1}, \ldots, \mathsf{Find}_{i_\ell}$ occur for some pairwise distinct $i_1, \ldots, i_\ell \in [I]$, where $\ell > n$. Then it means that for all $k \in [\ell]$, the set $\mathcal{Q}_{i_k} \cap \mathcal{Q}_G$ is nonempty. Moreover, all of them are pairwise disjoint since Eve never asks the same query. However, it will lead to the following contradiction

$$n < \ell \leq \bigcup_{k\in[\ell]} |\mathcal{Q}_{i_k} \cap \mathcal{Q}_G| \leq |\mathcal{Q}_G| \leq n.$$

□

The above two lemmas together imply that the number of occurring events among $\{\mathsf{Heavy}_i\}_{i\in[I]}$ is at most n in every execution.

Corollary 3.7. *For every $i \in [I]$, let I_{Heavy_i} be the indicator variable of* Heavy_i. *Then*

$$\Pr\left[\sum_{i\in[I]} I_{\mathsf{Heavy}_i} \leq n\right] = 1.$$

Proof. It immediately follows from Lemma 3.5 and Lemma 3.6. □

Before we move on, we rewrite Experiment 3.4 as the following equivalent experiment under the view of lazy evaluation:

Experiment 3.8.

1. Run $(\mathrm{P}, \mathrm{s}) \leftarrow \mathsf{Gen}^{(\cdot)}(r_G)$ by lazy evaluation. // Now, the set $\mathcal{P}_G$ of query-answer pairs is sampled.
2. Learn $\mathcal{L}_i \leftarrow \mathsf{E}^{(\cdot)}(\mathrm{P})$ as Step 2 in Experiment 3.4 by lazy evaluation.
   // Now, the set $\mathcal{L}_i$ of query-answer pairs is sampled.
3. For every $x \in \mathcal{X} \setminus (\mathcal{Q}_G \cup \mathcal{Q}(\mathcal{L}_i))$, uniformly sample a value y_x from $\mathcal{Y}$.
4. Combine all the pairs from Step 1 to 3 into a full oracle H.
5. Sample $H_i \leftarrow H\left[(\mathcal{Q}_G \setminus \mathcal{Q}(\mathcal{L}_i))^{\circlearrowleft}\right]$. // Private queries are resampled uniformly.
6. Run $\mathrm{s}_i \leftarrow \mathsf{Sol}^{H_i}(\mathrm{P})$.

The distribution of H_i is consistent with that in Experiment 3.4 since every query beyond $\mathcal{Q}(\mathcal{L}_i)$ is independently and uniformly sampled (in Steps 3 and 5).

We now show that, as long as the i^{th} round is not heavy, Eve's simulation will be good enough to generate the correct solution with high probability. Formally, we have the following lemma.

Lemma 3.9. *For every $i \in [I]$, it holds that*

$$\Pr\left[\mathrm{s} = \mathrm{s}_i \wedge \neg\mathsf{Heavy}_i\right] \geq \Pr\left[\mathrm{s} = \mathrm{s}' \wedge \neg\mathsf{Heavy}_i\right] - \delta.$$

where the probability is defined over Experiment 3.8 and s' is defined to be generated by $\mathsf{Sol}^{H}(P)$ instead of using H_i.

Proof. By Corollary 2.5 (setting $F = H_i$, $G = H$, $\mathcal{S} = \mathcal{Q}_G \setminus \mathcal{Q}(\mathcal{L}_i)$, $z = (\mathrm{P}, \mathrm{s})$, and $\mathcal{A}^F(z) = \mathcal{V} \circ \mathsf{Sol}^{H_i}(P)$, where $\mathcal{V}(\mathrm{s}_i) = 1$ if $\mathrm{s} = \mathrm{s}_i$, and 0 otherwise), we have

$$\begin{aligned}
&\Pr\left[\mathrm{s} = \mathrm{s}_i \wedge \neg\mathsf{Heavy}_i\right] \\
=& \Pr[\neg\mathsf{Heavy}_i] \cdot \Pr\left[\mathrm{s} = \mathrm{s}_i \mid \neg\mathsf{Heavy}_i\right] \\
=& \Pr[\neg\mathsf{Heavy}_i] \cdot \Pr\left[\mathcal{A}^F(z) = \mathcal{V} \circ \mathsf{Sol}^{H_i}(P) = 1 \mid \neg\mathsf{Heavy}_i\right] \\
\geq& \Pr[\neg\mathsf{Heavy}_i] \cdot \left(\Pr\left[\mathcal{A}^G(z) = \mathcal{V} \circ \mathsf{Sol}^{H}(P) = 1 \mid \neg\mathsf{Heavy}_i\right] - 2\sqrt{mn\delta'}\right) \\
\geq& \Pr\left[\mathcal{V} \circ \mathsf{Sol}^{H}(P) = 1 \wedge \neg\mathsf{Heavy}_i\right] - \delta \\
=& \Pr\left[\mathrm{s} = \mathrm{s}' \wedge \neg\mathsf{Heavy}_i\right] - \delta,
\end{aligned}$$

where the first inequality is due to Corollary 2.5 and the second inequality follows from our choice of parameter $\delta = 2\sqrt{mn\delta'}$. This finishes the proof. □

Finally, with the above lemmas in hand, the completeness can be proven in the following lemma:

Lemma 3.10 (Completeness of Construction 3.2)**.**

$$\Pr\left[\mathrm{s} = \mathrm{s}_{i^*} : \begin{array}{c} (\mathrm{P},\mathrm{s}) \leftarrow \mathsf{Gen}^H(r_G) \\ i^* \leftarrow [I] \\ \mathrm{s}_{i^*} \leftarrow \mathsf{Sol}^{H_{i^*}}(\mathrm{P}) \end{array}\right] \geq 1 - \rho - \varepsilon - \delta.$$

Proof. The success probability of Eve in Construction 3.2 is given by

$$\Pr\left[\mathrm{s} = \mathrm{s}_{i^*} : \begin{array}{c} (\mathrm{P},\mathrm{s}) \leftarrow \mathsf{Gen}^H(r_G) \\ i^* \leftarrow [I] \\ \mathrm{s}_{i^*} \leftarrow \mathsf{Sol}^{H_{i^*}}(\mathrm{P}) \end{array}\right] = \frac{1}{I}\sum_{i=1}^{I} \Pr\left[\mathrm{s} = \mathrm{s}_i\right]$$

$$
\begin{aligned}
&\geq \frac{1}{I}\sum_{i=1}^{I}\Pr\left[\mathrm{s}=\mathrm{s}_i \wedge \neg\mathsf{Heavy}_i\right] \overset{(i)}{\geq} \frac{1}{I}\sum_{i=1}^{I}\Pr\left[\mathrm{s}=\mathrm{s}' \wedge \neg\mathsf{Heavy}_i\right]-\delta \\
&\geq \frac{1}{I}\sum_{i=1}^{I}\left(\Pr\left[\mathrm{s}=\mathrm{s}'\right]-\Pr[\mathsf{Heavy}_i]\right)-\delta = \Pr\left[\mathrm{s}=\mathrm{s}'\right]-\frac{1}{I}\mathbb{E}\left[\sum_{i=1}^{I} I_{\mathsf{Heavy}_i}\right]-\delta \\
&\overset{\text{(ii)}}{\geq} \Pr\left[\mathrm{s}=\mathrm{s}'\right]-\varepsilon-\delta \overset{\text{(iii)}}{\geq} 1-\rho-\varepsilon-\delta,
\end{aligned}
$$

where (i) is due to Lemma 3.9, (ii) follows from Corollary 3.7 and (iii) follows from the completeness of the puzzle. □

This concludes the efficiency and completeness of Eve in Construction 3.2, and hence it finishes the proof of Part 1 of Theorem 3.1.

Part 2. We now move to prove Part 2 of Theorem 3.1. The idea is to use the *majority* of the solutions obtained in *all* rounds. Consider the following construction:

Construction 3.11 (Majority Vote Attack)**.** Let n, m be, in order, the number of queries made by Gen^H and Sol^H.

- Input: the puzzle P generated by Gen^H and a parameter $\delta \in (0,1]$.
- Set $I = 2n$ and initialize a list $\mathcal{L}_1 = \emptyset$ and a multiset $\mathcal{S} = \emptyset$.
- For $i \in \{1,\dots,I\}$, do the following:
  - Run $\mathsf{Solve}(i)$.
  - Update the solution set by $\mathcal{S} \leftarrow \mathcal{S} \cup \{\mathrm{s}_i\}$.
- Sample a full oracle $H_{I+1} \overset{\$}{\leftarrow} \mathcal{Y}^{\mathcal{X}}|_{\mathcal{L}_{I+1}}$.
- Run one more execution $\mathsf{Sol}^{H_{I+1}}(\mathrm{P})$ to get the solution s_{I+1}.
- Update the solution set by $\mathcal{S} \leftarrow \mathcal{S} \cup \{\mathrm{s}_{I+1}\}$.
- If there exists $\mathrm{s}^* \in \mathcal{S}$ that has multiplicity at least $I/2+1 = n+1$, output s^*.
  Otherwise, abort.

Efficiency. In each round, since the number of queries made by Sol is at most m, there are at most $\frac{m}{\delta'} = \frac{4nm^2}{\delta^2}$ points that can be added into $\mathcal{Q}_i$. Moreover, there are at most $I = 2n$ rounds. Therefore, Eve's adaptivity is $2n$ and the total number of Eve's queries is at most $\frac{8n^2m^2}{\delta^2}$.

Completeness. For every $i \in [I+1]$, let $\mathsf{Success}_i$ denote the event that $\mathrm{s} = \mathrm{s}_i$. Following the same argument in Lemma 3.9, Eve's simulation error is at most δ conditioned on the event $\neg\mathsf{Heavy}_i$ happening. Hence, for every $i \in [I+1]$, we have

$$
\begin{aligned}
\Pr[\mathsf{Success}_i \vee \mathsf{Heavy}_i] &= \Pr[\neg\mathsf{Heavy}_i]\cdot\Pr[\mathsf{Success}_i \mid \neg\mathsf{Heavy}_i] + \Pr[\mathsf{Heavy}_i] \\
&\geq 1-\rho-\delta.
\end{aligned}
$$

Equivalently, for every $i \in [I+1]$,

$$\Pr[\neg\mathsf{Success}_i \wedge \neg\mathsf{Heavy}_i] \leq \rho+\delta.$$

Therefore, by a union bound over $i \in [I+1]$, we have

$$\Pr\left[\bigwedge_{i\in[I+1]}(\mathsf{Success}_i \vee \mathsf{Heavy}_i)\right] \geq 1-(I+1)(\rho+\delta).$$

Suppose the above event holds for the rest of the proof. By Corollary 3.7, with probability one, the number of the happening events Heavy_i is at most n, which means the number of happening events $\mathsf{Success}_i$ is at least $n+1$. This implies the success of Eve's attack in Construction 3.11 and finishes the proof of Part 2 of Theorem 3.1. $\square$

3.2 Efficient Attacks on CGQS Time-Lock Puzzles

The attacks in the last subsection could be made efficient. In Construction 3.2 and Construction 3.11, in order to find queries with a large amplitude, Eve needs to sample a full oracle and compute exponentially many query amplitudes. Moreover, as a quantum nature, the queries made by $\mathsf{Sol}^{H_i}(\mathrm{P})$ cannot be "recorded". Fortunately, by leveraging Lemma 2.3, we still have an efficient way to extract queries with a large amplitude. Although directly invoking Lemma 2.3 guarantees some probability for successful extraction, the probability is still too low. In this way, it will increase the number of rounds (adaptivity) by a large factor.

To overcome this issue, Eve will instead run the solver many times in each round. By choosing parameters properly, we show that Eve can find queries with a large amplitude with high probability. Lastly, the efficient simulation of a (partially fixed) quantum random oracle can be done by using $2m$-wise independent functions [Zha12]. Below, we show how to convert Construction 3.2 into a quantum time-efficient attack.

Theorem 3.12 (Efficiently Breaking CGQS Time-Lock Puzzles)**.** *Consider any time-lock puzzle scheme in the random oracle model where the generator asks n classical queries, the solver asks m quantum queries and the completeness is $1-\rho$ (see Definition 2.6). For any $\varepsilon, \delta \in (0,1]$ and $\gamma > 1$, there exists an* efficient*, randomized solver Eve (denoted by E) who asks at most q_E classical queries in at most d_E rounds and in time $O(d_E \cdot T)$, where T is the running time of the honest solver, and it achieves the following bound for the completeness error $\nu = 1 - \Pr[\mathrm{s} = \mathrm{s}' : (\mathrm{P}, \mathrm{s}) \leftarrow \mathsf{Gen}^H(r_G), \mathrm{s}' \leftarrow \mathsf{E}^H(\mathrm{P})]$, where the probability is over the randomness of the generator, the randomness of Eve (including the randomness of using the quantum solver as subroutines) and the random oracle H.*

- *Small completeness error: Eve asks $q_E = \frac{4nm^2}{\varepsilon\delta^2} \ln\left(\frac{\gamma}{\gamma-1}\right)$ queries in $d_E = n/\varepsilon$ rounds with completeness error $\nu \le \rho + \gamma\varepsilon + \delta$.*

Remark 3.13. Here, we compare the parameters in Theorem 3.12 with the parameters in Theorem 3.1. Suppose we let $n = \ln\left(\frac{\gamma}{\gamma-1}\right)$ and let $n, m, \varepsilon, \delta$ be identical, then Eve's adaptivity and the number of total queries in each construction will be equivalent. In this way, we can solve the above equation and obtain $\gamma \approx 1 + e^{-n}$.

Remark 3.14. Construction 3.11 can be made efficient in a similar manner.

Proof of Theorem 3.12. Consider the following efficient variant of Construction 3.2:

Construction 3.15 (Efficient Random Stoppage Attack)**.** Let n, m be, in order, the number of queries made by Gen^H and Sol^H.

- Input: the puzzle P generated by Gen^H and parameters $\varepsilon, \delta \in (0,1], \gamma > 1$.
- Set $I = n/\varepsilon$ and initialize a list $\mathcal{L}_1 = \emptyset$.
- Pick $i^* \overset{\$}{\leftarrow} [I]$.
- For $i \in \{1, \dots, i^*\}$, do the following:
  - Set $\mathcal{Q}_i = \emptyset$.
  - Use $2m$-wise independent functions to simulate $H_i \overset{\$}{\leftarrow} \mathcal{Y}^{\mathcal{X}}|_{\mathcal{L}_i}$.
  - Run $\mathsf{Sol}^{H_i}(\mathrm{P})$ to get the solution s_i.
  - Let $\gamma' := \frac{m}{n\delta'} \ln\left(\frac{\gamma}{\gamma-1}\right)$, where $\delta' := \frac{\delta^2}{4nm}$.
  - For $j \in [\gamma']$, do the following:

    * Run $\mathsf{Ext}(\mathsf{Sol}^{H_i}(\mathrm{P}))$ to get $x \in \mathcal{X}$.
      (Note that the oracle used in each execution is fixed, i.e., H_i.)
    * If $x \notin \mathcal{Q}(\mathcal{L}_i) \cup \mathcal{Q}_i$, then $\mathcal{Q}_i \leftarrow \mathcal{Q}_i \cup \{x\}$.
  - Make classical queries to the (real) random oracle H on all points in $\mathcal{Q}_i$ and obtain $H(\mathcal{Q}_i)$.
  - Update the list by $\mathcal{L}_{i+1} \leftarrow \mathcal{L}_i \cup (\mathcal{Q}_i, H(\mathcal{Q}_i))$.
- Output s_{i^*}.

Efficiency. In each round, the size of $\mathcal{Q}_i$ is at most $\gamma' = \frac{m}{n\delta'} \ln\left(\frac{\gamma}{\gamma-1}\right) = \frac{4m^2}{\delta^2} \ln\left(\frac{\gamma}{\gamma-1}\right)$. Moreover, there are at most $I = n/\varepsilon$ rounds. Therefore, Eve's adaptivity is at most n/ε and the total number of Eve's queries is at most $\frac{4nm^2}{\varepsilon\delta^2} \ln\left(\frac{\gamma}{\gamma-1}\right)$. Notice that the iteration of the extractor in each round can be performed in parallel. In addition, the amount of randomness for initiating a $2m$-wise independent function is $O(m) = O(T)$. Hence, Eve's running time is $O(d_E \cdot T)$.

Completeness. The following lemma is the generalization of Lemma 3.5. Given that the i^{th} round is heavy, repeating the extractor in Lemma 2.3 many times will output a private query with high probability.

Lemma 3.16. *For every $i \in [I]$, it holds that*

$$\Pr[\mathsf{Find}_i \mid \mathsf{Heavy}_i] \geq \frac{1}{\gamma}.$$

Proof. Conditioned on the event Heavy_i happening, it means that the query amplitude of $\mathsf{Sol}^{H_i}(\mathrm{P})$ on the private queries $\mathcal{Q}_G \setminus \mathcal{Q}(\mathcal{L}_i)$ is at least $n\delta'$ in the i^{th} round. By Lemma 2.3, the probability of obtaining $x \in \mathcal{Q}_G \setminus \mathcal{Q}(\mathcal{L}_i)$ in each iteration of the extractor is at least $n\delta'/m$. Moreover, each iteration is independent because the randomness of the extractor comes from the choice of query and the measurement. Both of them are fresh every time. Hence, after γ' iterations, the probability of obtaining $x \in \mathcal{Q}_G \setminus \mathcal{Q}(\mathcal{L}_i)$ is at least

$$1 - \left(1 - \frac{n\delta'}{m}\right)^{\gamma'} \geq 1 - e^{\ln(1-1/\gamma)} = \frac{1}{\gamma},$$

where we use $1 - x \leq e^{-x}$ for $x \in \mathbb{R}$ and $\gamma' = \frac{m}{n\delta'} \ln\left(\frac{\gamma}{\gamma-1}\right)$. □

Lemma 3.17. *For every $i \in [I]$, let I_{Find_i} be the indicator variable of Find_i. Then*

$$\Pr\left[\sum_{i\in[I]} I_{\mathsf{Find}_i} \leq n\right] = 1.$$

Proof. The proof is the same as Lemma 3.6. □

The following corollary is the generalization of Corollary 3.7.

Corollary 3.18. *For every $i \in [I]$, let I_{Heavy_i} be the indicator variable of Heavy_i. Then*

$$\mathbb{E}\left[\sum_{i\in[I]} I_{\mathsf{Heavy}_i}\right] \leq \gamma n.$$

Proof. From Lemma 3.16, we have

$$\Pr[\mathsf{Heavy}_i] \leq \gamma \Pr[\mathsf{Heavy}_i \wedge \mathsf{Find}_i] \leq \gamma \Pr[\mathsf{Find}_i].$$

By Lemma 3.17, it holds that

$$\mathbb{E}\left[\sum_{i\in[I]} I_{\mathsf{Heavy}_i}\right] = \sum_{i\in[I]} \Pr[\mathsf{Heavy}_i] \leq \gamma \sum_{i\in[I]} \Pr[\mathsf{Find}_i] = \gamma\,\mathbb{E}\left[\sum_{i\in[I]} I_{\mathsf{Find}_i}\right] \leq \gamma n.$$

□

Finally, we have the following lemma, which is the counterpart of Lemma 3.10.

Lemma 3.19 (Completeness of Construction 3.15)**.**

$$\Pr\left[\mathrm{s} = \mathrm{s}_{i^*} : \substack{(\mathrm{P},\mathrm{s})\leftarrow\mathsf{Gen}^H(r_G)\\ i^*\leftarrow[I]\\ \mathrm{s}_{i^*}\leftarrow\mathsf{Sol}^{H_{i^*}}(\mathrm{P})}\right] \geq 1-\rho-\gamma\varepsilon-\delta.$$

Proof. Following the same lines in the proof of Lemma 3.10, we have

$$\begin{aligned}
&\Pr\left[\mathrm{s} = \mathrm{s}_{i^*} : \substack{(\mathrm{P},\mathrm{s})\leftarrow\mathsf{Gen}^H(r_G)\\ i^*\leftarrow[I]\\ \mathrm{s}_{i^*}\leftarrow\mathsf{Sol}^{H_{i^*}}(\mathrm{P})}\right]\\
&\geq \Pr\left[\mathrm{s} = \mathrm{s}' : \substack{(\mathrm{P},\mathrm{s})\leftarrow\mathsf{Gen}^H(r_G)\\ \mathrm{s}'\leftarrow\mathsf{Sol}^{H}(\mathrm{P})}\right] - \frac{1}{I}\,\mathbb{E}\left[\sum_{i\in[I]} I_{\mathsf{Heavy}_i}\right] - \delta\\
&\geq \Pr\left[\mathrm{s} = \mathrm{s}' : \substack{(\mathrm{P},\mathrm{s})\leftarrow\mathsf{Gen}^H(r_G)\\ \mathrm{s}'\leftarrow\mathsf{Sol}^{H}(\mathrm{P})}\right] - \gamma\varepsilon - \delta\\
&\geq 1-\rho-\gamma\varepsilon-\delta.
\end{aligned}$$

where the second to last inequality follows from Corollary 3.18. □

This concludes the efficiency and completeness of Eve in Construction 3.15. □

4 Round-Optimal Attack on Perfectly-Complete Time-Lock Puzzles

In this section, we present a round-optimal attack on *perfectly-complete* time-lock puzzles. As a warm-up, we describe our attack under the setting where both the puzzle generator and solver can make classical queries in Section 4.1. Compared to the round-optimal attack in [MMV11], our attack is strictly better in terms of success probability and the number of queries, though our attack is limited to perfectly complete schemes. In particular, the attack in [MMV11] needs $O((mn/\varepsilon)\log(n/\varepsilon))$ queries to find the solution with probability at least $1-\varepsilon$. While our attack only needs mn queries and finds the solution with probability 1. In Sections 4.2 and 4.3, we extend the attack to the settings in which the generator and solver, respectively, can make quantum queries.

4.1 Round-Optimal Attack on Perfectly-Complete CGCS Time-Lock Puzzles

Theorem 4.1. *Consider any time-lock puzzle scheme in the random oracle model where the generator asks n classical queries, the solver asks m classical queries and the completeness is 1. There exists a deterministic, inefficient solver Eve who asks at most nm classical queries in at most n rounds and finds the solution with probability 1.*

Definitions. We use the same notation defined in Section 3.1. Consider the full execution of the puzzle generator and (honest) solver. The *full sample* X is a tuple of random variables $(r_G, r_S, H, \mathrm{P}, \mathrm{s}, \mathrm{s}')$. Note that we only consider puzzles with perfect completeness, so every element in the support set of X must satisfy $\mathrm{s} = \mathrm{s}'$. The *view of the generator* is a tuple of random variables $v_G = (r_G, \mathcal{P}_G)$. The *view of the solver* is a tuple of random variables $v_G = (r_G, \mathrm{P}, \mathcal{P}_S)$. For any fixed view v_G, puzzle P, solution s and list $\mathcal{L}$, we say that v_G is consistent with $(\mathrm{P}, \mathrm{s}, \mathcal{L})$ if there exists some x in the support set of X such that v_G, P, s and $\mathcal{L}$ are all consistent with x. The consistency between the solver's view and $(\mathrm{P}, \mathrm{s}', \mathcal{L})$ is defined similarly except that s is replaced with s'.

Consider the following construction:

Construction 4.2. Let n, m be, in order, the number of queries made by Gen^H and Sol^H. Let $\mathcal{S}$ be the solution space.

1. Input: the puzzle P generated by Gen^H.
2. Initialize a list $\mathcal{L} = \emptyset$.
3. While the solution is not fully determined conditioned on $(\mathrm{P}, \mathcal{L})$,[3] do the following:
   (a) Pick the lexicographically first[4] $\mathrm{s}' \in \mathcal{S}$ such that there exists some solver's view v_S that is consistent with $(\mathrm{P}, \mathrm{s}', \mathcal{L})$. Pick the lexicographically first such v_S.
   (b) Query the (true) random oracle H on every point in $\mathcal{Q}_S \setminus \mathcal{Q}(\mathcal{L})$, where $\mathcal{Q}_S$ is the solver's query set defined in the view v_S. Then update the list $\mathcal{L}$ accordingly.
4. Output the remaining, unique solution.

The following observation is the crux of proving efficiency.

Lemma 4.3. *For any (fixed) puzzle* P*, list* $\mathcal{L}$*, solutions* $\mathrm{s}, \mathrm{s}' \in \mathcal{S}$ *such that* $\mathrm{s} \neq \mathrm{s}'$*, any generator's view* v_G *that is consistent with* $(\mathrm{P}, \mathcal{L}, \mathrm{s})$*, and any solver's view* v_S *that is consistent with* $(\mathrm{P}, \mathcal{L}, \mathrm{s}')$*, there exists some inconsistency between* $\mathcal{P}_G$ *and* $\mathcal{P}_S$*, i.e.,* $\exists q \in \mathcal{Q}_G \cap \mathcal{Q}_S$ *such that* $\mathcal{P}_G(q) \neq \mathcal{P}_S(q)$*, which implies that* $(\mathcal{Q}_G \cap \mathcal{Q}_S) \setminus \mathcal{Q}(\mathcal{L}) \neq \emptyset$*.*

Proof. Suppose it does not hold, then we can extend $\mathcal{P}_G, \mathcal{P}_S$ and $\mathcal{L}$ into a full oracle. Since r_G, r_S, and H are sampled independently, there exists some full sample x that is consistent with v_G and v_S simultaneously. However, this would contradict perfect completeness. □

Lemma 4.4. *The number of iterations in Step 3 of Construction 4.2 is at most n with probability* 1.

Proof. In each iteration, if all the answers in Step 3(b) already determine the solution, then we are done. Otherwise, by Lemma 4.3, the queries made in Step 3(b) must hit at least one element in $\mathcal{Q}_G \setminus \mathcal{Q}(\mathcal{L})$ of *every* generator's view that has a solution different from s'. Since the generator makes a total of n queries plus the fact that Eve always picks the lexicographically first s' in Step 3(a), inductively we can show that the latter case could occur at most n times. □

Proof of Theorem 4.1. By perfect completeness and construction, until the point Eve has queried the whole oracle, the solution will be fully determined. Hence, Eve's attack succeeds with probability 1. Moreover, by Lemma 4.4, Eve asks at most n rounds of queries. In each round, Eve asks at most m queries. This completes the proof. □

Remark 4.5. The attack in Construction 4.2 also works for perfectly-complete key agreements by replacing the solver with Alice, and the generator with Bob, respectively. It halves the number and round of queries of the attack in [BKSY11].

[3]That is, conditioned on fixed $(\mathrm{P}, \mathcal{L})$, there exist x_1, x_2 in the sample space such that *the* solution of x_1 is different from that of x_2. Note that due to perfect completeness, s and s' of every x are the same.

[4]Here, it's essential to pick the lexicographically first one for proving efficiency.

4.2 Round-Optimal Attack on Perfectly-Complete QGCS Time-Lock Puzzles

Interestingly, the same attack in Construction 4.2 also works when the puzzle generator is able to make quantum queries.

Theorem 4.6. *Consider any time-lock puzzle scheme in the random oracle model where the generator asks n quantum queries, the solver asks m classical queries and the completeness is 1. There exists a deterministic, inefficient solver Eve who asks at most nm classical queries in at most n rounds and finds the solution with probability 1.*

We define the purified view of quantum puzzle generators below. Note that since time-lock puzzles are non-interactive, defining the purified view only for the generator is sufficient for finishing the proof.

Definition 4.7 (Purified view of quantum puzzle generators)**.** Let $(\mathsf{Gen}, \mathsf{Sol})$ be a time-lock puzzle in which Gen can make quantum queries to the oracle. Let $\mathsf{Gen} = (U_n, U_{n-1}, \dots, U_0)$, where n is the number of queries made by Gen and U_i's are unitaries. WLOG we assume that Gen measures all of its registers in the computational basis at the end of the computation.[5] Consider the following registers that record every classical transcript right after the puzzle and solution are generated:

- G : The generator's internal register.
- F : The truth table of the (true) random oracle H.
- P : The register that records the classical puzzle.
- S : The register that records the classical solution.

Let $\mathcal{H}$ be the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$ where $\mathcal{X}$ is a finite set and $\mathcal{Y}$ is a finite additive group that has cardinality N. Let $\hat{\mathcal{Y}}$ be the dual group of $\mathcal{Y}$. Let the initial state $|\Psi_0\rangle$ be

$$|\Psi_0\rangle := |0\rangle_G \otimes |0\rangle_P \otimes |0\rangle_S \otimes \left(\sum_{h \in \mathcal{H}} \frac{1}{\sqrt{|\mathcal{H}|}} |h\rangle_F \right).$$

The *purified view* of quantum puzzle generators is a quantum state defined by

$$|\Psi_n\rangle := U_n O U_{n-1} O \dots U_1 O U_0 |\Psi_0\rangle = \sum_{w,\mathrm{P},\mathrm{s},D} \alpha_{w,\mathrm{P},\mathrm{s},D} |w\rangle_G |\mathrm{P}\rangle_P |\mathrm{s}\rangle_S |D\rangle_F,$$

where $D \in \hat{\mathcal{Y}}^{\mathcal{X}}$ and O is the controlled-query operator defined as

$$O : |x\rangle_X |y\rangle_Y |h\rangle_F \mapsto |x\rangle_X |y + h(x)\rangle_Y |h\rangle_F.$$

For $D \in \hat{\mathcal{Y}}^{\mathcal{X}}$, we also define $\mathcal{Q}(D) := \{x \in \mathcal{X} : D(x) \neq \hat{0}\}$. $\Diamond$

Lemma 4.8 ([Zha19])**.** *The final state $|\Psi\rangle$ of the generator can be written as*

$$|\Psi\rangle = \sum_{w,\mathrm{P},\mathrm{s},D : |D| \leq n} \alpha_{w,\mathrm{P},\mathrm{s},D} |w\rangle_G |\mathrm{P}\rangle_P |\mathrm{s}\rangle_S |D\rangle_F,$$

where $|D| := |\mathcal{Q}(D)|$.

Proof of Theorem 4.6. Similarly, it is sufficient to prove efficiency. First, we augment an additional register L starting with the content $\emptyset$ that records Eve's list. Consider any (fixed) puzzle P that has a non-zero probability. After Eve saw the puzzle P, the purified view collapses to the post-measurement state of measuring the register P and obtaining the outcome P. Hence, the state is of the form

$$|\Psi_{\mathrm{P},\emptyset}\rangle = \sum_{w,\mathrm{s},D : |D| \leq n} \alpha_{w,\mathrm{P},\mathrm{s},D,\mathcal{L}} |w\rangle_G |\mathrm{P}\rangle_P |\mathrm{s}\rangle_S |D\rangle_F |\emptyset\rangle_L. \tag{2}$$

[5]Doing so will not change the distribution of the experiment since the communication (puzzle) is classical.

In each iteration, the dynamical process of the purified view is modeled in the following recursive way. Suppose the current purified view at the beginning of the i^{th} iteration is

$$|\Psi_{\mathrm{P},\mathcal{L}}\rangle = \sum_{w,\mathrm{s},D:|D|\le n_i} \alpha_{w,\mathrm{P},\mathrm{s},D,\mathcal{L}} |w\rangle_G |\mathrm{P}\rangle_P |\mathrm{s}\rangle_S |D \sqcup \mathcal{L}\rangle_F |\mathcal{L}\rangle_L$$

for some fixed $\mathcal{L}$ and integer $n_i \ge 0$, where $D \in \hat{\mathcal{Y}}^{\mathcal{X}\setminus\mathcal{Q}(\mathcal{L})}$ and $|D \sqcup \mathcal{L}\rangle_F$ is the basis vector that stores $D(x)$ in the Fourier basis for every $x \in \mathcal{X} \setminus \mathcal{Q}(\mathcal{L})$ and stores $\mathcal{L}(x)$ in the computational basis for every $x \in \mathcal{Q}(\mathcal{L})$. We note that (2) is of this form with $\mathcal{L} = \emptyset$ and $n_1 = n$.

After Eve picks a solver's view v_S in Step 3(a), Eve will make classical queries defined in the view v_S and update $\mathcal{L}$ in register L. Let $\mathcal{L}'$ denote the updated list. Further conditioned on any possible (fixed) $\mathcal{L}'$, the purified view will be the post-measurement state corresponding to $\mathcal{L}'$ of the following form

$$|\Psi_{\mathrm{P},\mathcal{L}'}\rangle = \sum_{w,\mathrm{s},D:|D|\le n_{i+1}} \alpha_{w,\mathrm{P},\mathrm{s},D,\mathcal{L}'} |w\rangle_G |\mathrm{P}\rangle_P |\mathrm{s}\rangle_S |D \sqcup \mathcal{L}'\rangle_F |\mathcal{L}'\rangle_L.$$

We use the following notation to further fix w and s in $|\Psi_{\mathrm{P},\mathcal{L}}\rangle$:

$$|\Psi_{\mathrm{P},\mathcal{L}}\rangle = \sum_{w,\mathrm{s}} \beta_{w,\mathrm{P},\mathrm{s},\mathcal{L}} |\Psi_{\mathrm{P},w,\mathrm{s},\mathcal{L}}\rangle_{GPSF} |\mathcal{L}\rangle_L,$$

where

$$\begin{aligned}|\Psi_{\mathrm{P},w,\mathrm{s},\mathcal{L}}\rangle &:= |w\rangle_G |\mathrm{P}\rangle_P |\mathrm{s}\rangle_S \otimes \frac{1}{\beta_{w,\mathrm{P},\mathrm{s},\mathcal{L}}} \cdot \sum_{D:|D|\le n_i} \alpha_{w,\mathrm{P},\mathrm{s},D,\mathcal{L}} |D \sqcup \mathcal{L}\rangle_F \\ &= |w\rangle_G |\mathrm{P}\rangle_P |\mathrm{s}\rangle_S \otimes \sum_{D:|D|\le n_i} \gamma_{w,\mathrm{P},\mathrm{s},D,\mathcal{L}} |D \sqcup \mathcal{L}\rangle_F,\end{aligned}$$

$\beta_{w,\mathrm{P},\mathrm{s},\mathcal{L}}$ is the normalization factor so that $|\Psi_{\mathrm{P},w,\mathrm{s},\mathcal{L}}\rangle$ is of unit length, i.e.,

$$\beta_{w,\mathrm{P},\mathrm{s},\mathcal{L}} := \sum_{D:|D|\le n_{i+1}} |\alpha_{w,\mathrm{P},\mathrm{s},D,\mathcal{L}}|^2,$$

and $\gamma_{w,\mathrm{P},\mathrm{s},D,\mathcal{L}} := \alpha_{w,\mathrm{P},\mathrm{s},D,\mathcal{L}} / \beta_{w,\mathrm{P},\mathrm{s},\mathcal{L}}$. In particular, for every $|\Psi_{\mathrm{P},w,\mathrm{s},\mathcal{L}}\rangle$, we say that D is a *maximum database*[6] if $\gamma_{w,\mathrm{P},\mathrm{s},D,\mathcal{L}} \ne 0$ and there does not exist D' such that $\gamma_{w,\mathrm{P},\mathrm{s},D',\mathcal{L}} \ne 0$ and $|D'| > |D|$. Let $\mathcal{D}^{\max}_{\mathrm{P},w,\mathrm{s},\mathcal{L}}$ denote the set of all maximum databases of $|\Psi_{\mathrm{P},w,\mathrm{s},\mathcal{L}}\rangle$.

We have the following observation which is the counterpart of Lemma 4.3.

Lemma 4.9. *For any (fixed) puzzle* P*, list* $\mathcal{L}$*, solutions* $\mathrm{s},\mathrm{s}' \in \mathcal{S}$ *such that* $\mathrm{s} \ne \mathrm{s}'$*, any* w *such that* $\beta_{w,\mathrm{P},\mathrm{s},\mathcal{L}} > 0$ *in* $|\Psi_{\mathrm{P},\mathcal{L}}\rangle$ *and any solver's view* v_S *that is consistent with* $(\mathrm{P},\mathcal{L},\mathrm{s}')$*,*

$$\mathcal{Q}_S \cap \mathcal{Q}(D) \ne \emptyset$$

for every $D \in \mathcal{D}^{\max}_{\mathrm{P},w,\mathrm{s},\mathcal{L}}$.

Proof. For sake of contradiction, suppose there exist $|\Psi_{\mathrm{P},w,\mathrm{s},\mathcal{L}}\rangle$, a view v_S, and $D^* \in \mathcal{D}^{\max}_{\mathrm{P},w,\mathrm{s},\mathcal{L}}$ such that $\mathcal{Q}_S \cap \mathcal{Q}(D^*) = \emptyset$. First, we claim that

$$\|\Pi_{\mathcal{P}_S} |\Psi_{\mathrm{P},w,\mathrm{s},\mathcal{L}}\rangle\| > 0, \tag{3}$$

where $\mathcal{P}_S$ is defined in v_s and $\Pi_{\mathcal{P}_S}$ is the projection operator acting on register F defined as $\bigotimes_{x\in\mathcal{Q}_S} |\mathcal{P}_S(x)\rangle\langle\mathcal{P}_S(x)|$. In particular, consider the list $\mathcal{L}'$ defined by combining $\mathcal{L}$ and $\mathcal{P}_S$. For every $D \in \hat{\mathcal{Y}}^{\mathcal{X}\setminus\mathcal{Q}(\mathcal{L})}$, define the database that is "reprogrammed" according to $\mathcal{L}'$, i.e.,

$$D|_{\mathcal{L}'}(x) := \begin{cases} \mathcal{L}'(x) & \text{if } x \in \mathcal{Q}(\mathcal{L}') \\ D(x) & \text{if } x \in \mathcal{X} \setminus \mathcal{Q}(\mathcal{L}'). \end{cases}$$

[6]Note that D's are now defined on the restricted domain $\mathcal{X} \setminus \mathcal{Q}(\mathcal{L})$.

Also, we define the database $D^\circ \in \hat{\mathcal{Y}}^{\mathcal{X}\setminus\mathcal{Q}(\mathcal{L}')}$ as the database D^* restricted on $\mathcal{X} \setminus \mathcal{Q}(\mathcal{L}')$, i.e.,

$$D^\circ(x) := D^*(x) \quad \forall x \in \mathcal{X} \setminus \mathcal{Q}(\mathcal{L}').$$

Since we assume that $\mathcal{Q}(D^*) \cap \mathcal{Q}_S = \emptyset$, the non-zero entries of D^* will not be changed by $\mathcal{L}'$, i.e.,

$$D^*|_{\mathcal{L}'} = D^\circ \sqcup \mathcal{L}'.$$

Now we are ready to prove (3). Suppose

$$|\Psi_{\mathrm{P},w,\mathrm{s},\mathcal{L}}\rangle = |w\rangle_G |\mathrm{P}\rangle_P |\mathrm{s}\rangle_S \otimes \sum_{D:|D|\le n_i} \gamma_{w,\mathrm{P},\mathrm{s},D,\mathcal{L}} |D \sqcup \mathcal{L}\rangle_F.$$

Since we assume that $\mathcal{Q}(D^*) \cap \mathcal{Q}_S = \emptyset$, there does not exist $D \in \hat{\mathcal{Y}}^{\mathcal{X}\setminus\mathcal{Q}(\mathcal{L})}$ such that $\gamma_{w,\mathrm{P},\mathrm{s},D,\mathcal{L}} \neq 0$ and $D|_{\mathcal{L}'} = D^*|_{\mathcal{L}'} = D^\circ \sqcup \mathcal{L}'$ by the maximality of D^*. Therefore, if we consider the (sub-normalized) state after the projection

$$\Pi_{\mathcal{P}_S} |\Psi_{\mathrm{P},w,\mathrm{s},\mathcal{L}}\rangle = \sum_{\overline{D}:|\overline{D}|\le n_{i+1}} \gamma'_{w,\mathrm{P},\mathrm{s},\overline{D},\mathcal{L}'} |\overline{D} \sqcup \mathcal{L}'\rangle_F$$

where $\overline{D}$'s are defined on $\mathcal{X} \setminus \mathcal{Q}(\mathcal{L}')$, the coefficient satisfies

$$\gamma'_{w,\mathrm{P},\mathrm{s},D^\circ,\mathcal{L}'} = N^{\frac{-1}{2}|\mathcal{Q}_S \setminus \mathcal{Q}(\mathcal{L})|} \cdot \gamma_{w,\mathrm{P},\mathrm{s},D^*,\mathcal{L}} \neq 0.$$

Hence, we complete the proof of (3). However, by (3), if we represent $|\Psi_{\mathrm{P},w,\mathrm{s},\mathcal{L}}\rangle$ in the computational basis as

$$|\Psi_{\mathrm{P},w,\mathrm{s},\mathcal{L}}\rangle = |w\rangle_G |\mathrm{P}\rangle_P |\mathrm{s}\rangle_S \otimes \sum_{h\in\mathcal{H}} \gamma''_{w,\mathrm{P},\mathrm{s},h,\mathcal{L}} |h\rangle_F,$$

then there exists $h \in \mathcal{H}$ such that $\gamma''_{w,\mathrm{P},\mathrm{s},h,\mathcal{L}} \neq 0$ and h is consistent with $\mathcal{P}_S$. Consequently, we have $\Pr[h, \mathrm{P}, w, \mathrm{s}, \mathcal{L}] > 0$ which implies that $\Pr[h, \mathrm{P}, \mathrm{s}] > 0$. This means that if the random oracle is picked to be h, then there is a non-zero probability that Gen^h outputs (P, s). Moreover, since h is consistent with $\mathcal{P}_S$, when the solver takes as input P and picks the randomness r_S defined in v_S, the output will be $\mathsf{Sol}^h(\mathrm{P}; r_S) = \mathrm{s}'$. However, the above together would imply that there is a non-zero probability over the full execution that $\mathrm{s} \neq \mathrm{s}'$, which contracts the perfect completeness of the puzzle. □

We complete the proof by following the similar lines in Theorem 4.4. For any i, in the i^{th} iteration, if the learned query-answer pairs already determine the solution, then we are done. Otherwise, for every $|\Psi_{\mathrm{P},w,\mathrm{s},\mathcal{L}}\rangle$ such that $\mathrm{s} \neq \mathrm{s}'$, we must have $n_{i+1} \le n_i - 1$ by Lemma 4.9. Finally, since $n_1 = n$, the number of iterations is at most n. □

4.3 Round-Optimal Attack on Perfectly-Complete CGQS Time-Lock Puzzles

In this subsection, we consider a "dual" setting in which the generator can only make classical queries while the solver can make quantum queries.

Theorem 4.10. *Consider any time-lock puzzle scheme in the random oracle model where the generator asks n classical queries, the solver asks m quantum queries and the completeness is 1. There exists a deterministic, inefficient solver Eve who asks at most nm classical queries in at most n rounds and finds the solution with probability 1.*

Now, since the solver is quantum, it is not clear how to define a view for the solver. To overcome the challenge, consider the following experiment in which we treat the quantum solver as an *isolated* algorithm.

Experiment 4.11 (Isolated Solver's Experiment)**.**

1. Input: a puzzle P

2. Sample $H \overset{\$}{\leftarrow} \mathcal{H}$.

3. Run $\mathsf{Sol}^H(\mathrm{P})$ to get s'.

4. Output s'.

We emphasize that Experiment 4.11 totally ignores the correlation between the puzzle P and oracle H. Note that conditioned on a fixed puzzle P could change the distribution of the oracle.[7] Hence, the distribution of Experiment 4.11 does *not* faithfully reflect the distribution of the full execution. However, assuming perfect completeness, showing the *positivity* of the probability of certain events occurring is sufficient to lead to a contradiction. We define the purified view of Experiment 4.11 as follows.

Definition 4.12. For any fixed P and quantum solver $\mathsf{Sol} = (U_m, U_{m-1}, \ldots, U_0)$, the purified view of Experiment 4.11 is defined to be

$$|\Psi_{\mathrm{P}}\rangle := U_m^{\mathrm{P}} O U_{m-1}^{\mathrm{P}} O \ldots U_1^{\mathrm{P}} O U_0^{\mathrm{P}} |\Psi_0\rangle_{SS'F} |\mathrm{P}\rangle_P,$$

where the dependency between U_i and P is specified. The register S is the solver's internal register and register S' records the solver's classical solution. $\diamondsuit$

By Lemma 4.8, the purified view is of the form

$$|\Psi_{\mathrm{P}}\rangle = \sum_{w,\mathrm{s}',D:|D|\le m} \alpha_{w,\mathrm{s}',D} |w\rangle_S |\mathrm{s}'\rangle_{S'} |D\rangle_F |\mathrm{P}\rangle_P.$$

Similarly, we can also define the state $|\Psi_{\mathrm{P},w,\mathrm{s}',\mathcal{L}}\rangle$ that is further conditioned on $(w, \mathrm{s}', \mathcal{L})$ which has a non-zero probability of happening.

Construction 4.13. Let n, m be, in order, the number of queries made by Gen^H and Sol^H. Let $\mathcal{S}$ be the solution space.

1. Input: the puzzle P generated by Gen^H.

2. Initialize a list $\mathcal{L} = \emptyset$.

3. While the solution is not fully determined conditioned on $(\mathrm{P}, \mathcal{L})$, do the following:

   (a) Pick the lexicographically first $\mathrm{s}' \in \mathcal{S}$ such that there exists a tuple of fixed $(w, \mathrm{s}', \mathcal{L})$ such that $\||\Psi_{\mathrm{P},w,\mathrm{s}',\mathcal{L}}\rangle\| > 0$.

   (b) Pick the lexicographically first such $|\Psi_{\mathrm{P},w,\mathrm{s}',\mathcal{L}}\rangle$.

   (c) Pick the lexicographically first $D \in \mathcal{D}^{\max}_{\mathrm{P},w,\mathrm{s}',\mathcal{L}}$.

   (d) Query the (true) random oracle H on every point in $\mathcal{Q}(D)$, and then update the list $\mathcal{L}$ accordingly.

4. Output the remaining, unique solution.

We have the following observation that aligns with Lemma 4.9 except that the roles of the generator and solver are switched.

Lemma 4.14. *For any (fixed) puzzle* P*, list* $\mathcal{L}$*, solutions* $\mathrm{s}, \mathrm{s}' \in \mathcal{S}$ *such that* $\mathrm{s} \neq \mathrm{s}'$*, any* w *such that* $\||\Psi_{\mathrm{P},w,\mathrm{s}',\mathcal{L}}\rangle\| > 0$ *and any generator's view* v_G *that is consistent with* $(\mathrm{P}, \mathcal{L}, \mathrm{s})$*,*

$$\mathcal{Q}_G \cap \mathcal{Q}(D) \neq \emptyset$$

for every $D \in \mathcal{D}^{\max}_{\mathrm{P},w,\mathrm{s}',\mathcal{L}}$.

Proof. The proof is virtually the same as that of Lemma 4.9. Suppose not, then there exists a fixed, full oracle $h \in \mathcal{H}$ relative to which the generator could possibly generate (P, s) but $\Pr[\mathsf{Sol}^h(\mathrm{P}) = \mathrm{s}'] > 0$, which contradicts perfect completeness. $\square$

Proof of Theorem 4.10. Similarly, it remains to prove efficiency. By Lemma 4.14, $\mathcal{Q}(D)$ of the database D picked by Eve in Step 3(c) always intersects with $\mathcal{Q}_G$.[8] This ensures that there will be at most n iterations. $\square$

[7]For example, P is defined to be $H(0)$.

[8]Recall that D is defined over the restricted domain $\mathcal{X} \setminus \mathcal{Q}(\mathcal{L})$. So Eve will learn something "new" beyond $\mathcal{Q}(\mathcal{L})$.

5 Post-Quantum Security of Pseudo-Chains

Notation. For $m, n \in \mathbb{N}$, by $\mathsf{Func}(m,n)$ we denote the set of all functions from m bits to n bits. For $H \in \mathsf{Func}(m,n), x^* \in \{0,1\}^m$ and $y^* \in \{0,1\}^n$, by $H[x^* \mapsto y^*]$ we denote the function obtained by reprogramming $H(x^*)$ to y^*. That is,

$$H[x^* \mapsto y^*](x) := \begin{cases} y^* & \text{if } x = x^* \\ H(x) & \text{otherwise.} \end{cases}$$

Definition 5.1 (Pseudo-chain puzzles [MMV11]). Let $H : \{0,1\}^\kappa \to \{0,1\}^\kappa$ be a random oracle. The *pseudo-chain* puzzle experiment is defined as follows. On input the security parameter 1^κ and an integer $q \geq 0$, do the following:

1. Sample $x_0, x_1, \dots, x_{q+1} \xleftarrow{\$} \{0,1\}^\kappa$.
2. Make (one-round of) queries to the random oracle H on $(x_0, \dots, x_q)$ and obtain the answers $(H(x_0), \dots, H(x_q))$.
3. Set $\mathrm{P} := (x_0, H(x_0) \oplus x_1, \dots, H(x_q) \oplus x_{q+1})$ and $\mathrm{s} := (x_0, \dots, x_{q+1})$.
4. Send P to the (solver) algorithm $\mathcal{A}$.
5. After making queries to H, the algorithm $\mathcal{A}$ outputs s'.

We write $\langle \mathcal{C}, \mathcal{A} \rangle$ to denote the interaction between the challenger $\mathcal{C}$ and the algorithm $\mathcal{A}$ in the pseudo-chain puzzle experiment. We define $\langle \mathcal{C}, \mathcal{A} \rangle = 1$ if and only if $\mathrm{s} = \mathrm{s}'$, and 0 otherwise. It's easy to see that if $\mathcal{A}$ can make at least $(q+1)$ rounds of queries, then $\mathcal{A}$ can always find s. $\Diamond$

Definition 5.2. The k-parallel query operator is defined as

$$O^{\otimes k} : |x_1, \dots, x_k\rangle|y_1, \dots, y_k\rangle \mapsto |x_1, \dots, x_k\rangle|y_1 \oplus H(x_1), \dots, y_k \oplus H(x_k)\rangle.$$

$\Diamond$

Theorem 5.3. *For any integers $k, q \geq 0$ and any computationally unbounded algorithm $\mathcal{A}$ that makes at most q k-parallel quantum queries to a random oracle $H : \{0,1\}^\kappa \to \{0,1\}^\kappa$, it holds that*

$$\Pr[\langle \mathcal{C}, \mathcal{A} \rangle(1^\kappa, q) = 1] = O\left(\sqrt{\frac{kq^3}{2^\kappa}}\right).$$

Proof. First, we provide some intuitions before the proof. Informally, in order for the adversary $\mathcal{A}$ to solve the puzzle better than randomly guessing x_{q+1}, there are only two possibilities: (i) Collision: there are some $i, j \in \{0, 1, \dots, q\}$ such that $x_i = x_j$ and $i \neq j$ (ii) Lucky guess: there is some $i \in [q]$ such that $\mathcal{A}$'s queries in the i-th round hit any of $\{x_i, x_{i+1}, \dots, x_q\}$. Looking ahead, hybrids $\mathcal{H}_{real}$ and $\mathcal{H}_0$ capture case (i), and hybrids $\mathcal{H}_0, \dots, \mathcal{H}_q$ capture case (ii). Finally, we show that the algorithm $\mathcal{A}$ in $\mathcal{H}_q$ can only make a random guess to solve the puzzle since x_{q+1} is information-theoretically hidden from $\mathcal{A}$.

Next, for convenience, we rewrite the pseudo-chain puzzle experiment into the following equivalent experiment $\mathcal{H}_{real}$.

Experiment $\mathcal{H}_{real}$:

1. Sample $H \xleftarrow{\$} \mathsf{Func}(\kappa, \kappa)$.
2. Sample $x_0, x_1, \dots, x_{q+1} \xleftarrow{\$} \{0,1\}^\kappa$.
3. Sample $y_0, y_1, \dots, y_q \xleftarrow{\$} \{0,1\}^\kappa$.

4. Define the oracle
$$H_q := H \begin{bmatrix} x_q \mapsto y_q, \\ x_{q-1} \mapsto y_{q-1}, \\ \vdots \\ x_0 \mapsto y_0 \end{bmatrix}.$$
(Note that the order of reprogramming is from top to bottom. Hence, if there exist repeated x_i's, the function value $H_q(x_i)$ is defined to be y_j where j is the *smallest* index such that $x_i = x_j$. This convention aligns with the lazy evaluation and makes our later analysis more convenient.)
5. Set $\mathsf{P} := (x_0, H_q(x_0) \oplus x_1, \ldots, H_q(x_q) \oplus x_{q+1})$ and $\mathsf{s} := (x_0, \ldots, x_{q+1})$.
6. Send P to the algorithm $\mathcal{A}$.
7. The algorithm $\mathcal{A}$ adaptively makes q k-parallel queries to the oracle, where all the queries are answered by H_q.
8. The algorithm $\mathcal{A}$ outputs $\mathsf{s}' = (x'_0, x'_1, \ldots, x'_{q+1})$.
9. Output 1 if and only if $\mathsf{s} = \mathsf{s}'$, and 0 otherwise.

We finish the proof by hybrid arguments. Consider the following hybrid. The difference between $\mathcal{H}_0$ and the pseudo-chain puzzle experiment $\mathcal{H}_{real}$ is colored in red.

Hybrid $\mathcal{H}_0$:

1. Sample $H \xleftarrow{\$} \mathsf{Func}(\kappa, \kappa)$.
2. Sample $x_0, x_1, \ldots, x_{q+1} \xleftarrow{\$} \{0,1\}^\kappa$.
3. Sample $y_0, y_1, \ldots, y_q \xleftarrow{\$} \{0,1\}^\kappa$.
4. Define the oracle
$$H_q := H \begin{bmatrix} x_q \mapsto y_q, \\ x_{q-1} \mapsto y_{q-1}, \\ \vdots \\ x_0 \mapsto y_0 \end{bmatrix}.$$
5. Set $\mathsf{P} := (x_0, y_0 \oplus x_1, \ldots, y_q \oplus x_{q+1})$ and $\mathsf{s} := (x_0, \ldots, x_{q+1})$.
6. Send P to the algorithm $\mathcal{A}$.
7. The algorithm $\mathcal{A}$ adaptively makes q k-parallel queries to the oracle, where all the queries are answered by H_q.
8. The algorithm $\mathcal{A}$ outputs $\mathsf{s}' = (x'_0, x'_1, \ldots, x'_{q+1})$.
9. Output 1 if and only if $\mathsf{s} = \mathsf{s}'$, and 0 otherwise.

Suppose there exist repeated x_i's, then the same value $H(x_i) = y_j$ is used to generate the puzzle in $\mathcal{H}_{real}$; while independent, uniform y_i's are used in $\mathcal{H}_0$. On the other hand, if all x_i's are pairwise distinct, then the two experiments are identically distributed. Let Col be the event defined over both $\mathcal{H}_0$ and $\mathcal{H}_{real}$ that $x_0, x_1, \ldots, x_{q+1}$ are not pairwise distinct. Then we have
$$\Pr_{\mathcal{H}_{real}}[\langle \mathcal{C}, \mathcal{A} \rangle (1^\kappa, q) = 1 \wedge \overline{\mathsf{Col}}] = \Pr_{\mathcal{H}_0}[\langle \mathcal{C}, \mathcal{A} \rangle (1^\kappa, q) = 1 \wedge \overline{\mathsf{Col}}]$$
which implies
$$\left| \Pr_{\mathcal{H}_{real}}[\langle \mathcal{C}, \mathcal{A} \rangle (1^\kappa, q) = 1] - \Pr_{\mathcal{H}_0}[\langle \mathcal{C}, \mathcal{A} \rangle (1^\kappa, q) = 1] \right| \leq \Pr[\mathsf{Col}] = O\left(\frac{q^2}{2^\kappa}\right).$$
Next, consider the following sequence of hybrids in which the order of oracles answering $\mathcal{A}'s$ queries is different:

Hybrid $\mathcal{H}_i$ for $i \in \{1, 2, \dots, q-1, q\}$:

1. Sample $H \xleftarrow{\$} \mathsf{Func}(\kappa, \kappa)$.
2. Sample $x_0, x_1, \dots, x_{q+1} \xleftarrow{\$} \{0,1\}^\kappa$.
3. Sample $y_0, y_1, \dots, y_q \xleftarrow{\$} \{0,1\}^\kappa$.
4. Define a sequence of oracles for $j \in \{0, 1, \dots, i-1\}$:

$$H_j := H \begin{bmatrix} x_j \mapsto y_j, \\ x_{j-1} \mapsto y_{j-1}, \\ \vdots \\ x_0 \mapsto y_0 \end{bmatrix}.$$

5. Set $\mathrm{P} := (x_0, y_0 \oplus x_1, \dots, y_q \oplus x_{q+1})$ and $\mathrm{s} := (x_0, \dots, x_{q+1})$.
6. Send P to the algorithm $\mathcal{A}$.
7. The algorithm $\mathcal{A}$ adaptively makes q k-parallel queries to the oracle, where the queries are answered in the following way. For $j \in \{1, 2, \dots, i\}$, the j^{th} query is answered by H_{j-1}. All the remaining queries are answered by H_q.
8. The algorithm $\mathcal{A}$ outputs $\mathrm{s}' = (x'_0, x'_1, \dots, x'_{q+1})$.
9. Output 1 if and only if $\mathrm{s} = \mathrm{s}'$, and 0 otherwise.

For convenience, the oracle calls of $\mathcal{A}$ in each hybrid are summarized in Table 1.

Table 1: Oracle calls of $\mathcal{A}$ in each hybrid.

	Index of queries								
	1	2	3	$\dots$	i	$i+1$	$i+2$	$\dots$	q
$\mathcal{H}_{real}$	H_q	H_q	H_q	$\dots$	H_q	H_q	H_q	$\dots$	H_q
$\mathcal{H}_0$	H_q	H_q	H_q	$\dots$	H_q	H_q	H_q	$\dots$	H_q
$\mathcal{H}_1$	H_0	H_q	H_q	$\dots$	H_q	H_q	H_q	$\dots$	H_q
$\mathcal{H}_2$	H_0	H_1	H_q	$\dots$	H_q	H_q	H_q	$\dots$	H_q
$\vdots$									
$\mathcal{H}_i$	H_0	H_1	H_2	$\dots$	H_{i-1}	H_q	H_q	$\dots$	H_q
$\mathcal{H}_{i+1}$	H_0	H_1	H_2	$\dots$	H_{i-1}	H_i	H_q	$\dots$	H_q
$\vdots$									
$\mathcal{H}_q$	H_0	H_1	H_2	$\dots$	H_{i-1}	H_i	H_{i+1}	$\dots$	H_{q-1}

The following lemma states the statistical closeness of the hybrids. For intuition, let us consider a simpler case where $\mathcal{A}$ only makes classical queries. First, the only difference between $\mathcal{H}_i$ and $\mathcal{H}_{i+1}$ is that the $(i+1)^{\text{th}}$ oracle call is answered by H_q or H_i. Moreover, the oracles H_q and H_i can only be inconsistent on $x_{i+1}, \dots, x_q$ by definition. However, the first i queries answered by $H_0, \dots, H_{i-1}$ contain no information about $x_i, y_i, \dots, x_q, y_q, x_{q+1}$. Therefore, before the $(i+1)^{\text{th}}$ query of $\mathcal{A}$, each of the last $(q-i+1)$ coordinates of the puzzle $y_i \oplus x_{i+1}, \dots, y_q \oplus x_{q+1}$ are the XOR of two fresh, uniform bitstrings. Hence, the distribution of $x_{i+1}, \dots, x_q$ in $\mathcal{A}$'s view is the same as $(q-i)$ independent, uniform bitstrings. So the probability that some of the k parallel branches of the $(i+1)^{\text{th}}$ query hits some of $x_{i+1}, \dots, x_q$ is roughly $O(k(q-i)/2^\kappa)$.

Lemma 5.4. *For* $i \in \{0,1,\ldots,q-1\}$,

$$\left|\Pr_{\mathcal{H}_i}[\langle \mathcal{C},\mathcal{A}\rangle(1^\kappa,q)=1] - \Pr_{\mathcal{H}_{i+1}}[\langle \mathcal{C},\mathcal{A}\rangle(1^\kappa,q)=1]\right| \le O\left(\sqrt{\frac{k(q-i)}{2^\kappa}}\right).$$

Proof. We rewrite each hybrid $\mathcal{H}_i$ as the following experiment $\mathcal{H}'_i$. The differences between them are colored in red.

```
Hybrid $\mathcal{H}'_i$ for $i \in \{0,1,\ldots,q-1,q\}$ :
  1. Sample $H \xleftarrow{\$} \mathsf{Func}(\kappa,\kappa)$.
  2. Sample $x_0,x_1,\ldots,x_{i-1},x_i \xleftarrow{\$} \{0,1\}^\kappa$.        // $x_{i+1},\ldots,x_{q+1}$ are not sampled.
  3. Sample $y_0,y_1,\ldots,y_{i-1} \xleftarrow{\$} \{0,1\}^\kappa$.        // $y_i,\ldots,y_q$ are not sampled.
  4. Define a series of oracles for $j \in \{0,1,\ldots,i-1\}$:
         $H_j := H\begin{bmatrix} x_j \mapsto y_j, \\ x_{j-1}\mapsto y_{j-1}, \\ \vdots \\ x_0 \mapsto y_0 \end{bmatrix}$.
  5. Sample $z_{i+1},\ldots,z_{q+1} \xleftarrow{\$} \{0,1\}^\kappa$.
  6. Set $\mathsf{P} := (x_0, y_0 \oplus x_1, \ldots, y_{i-1} \oplus x_i, z_{i+1},\ldots,z_{q+1})$.        // s is not defined yet.
  7. Send P to $\mathcal{A}$.
  8. The algorithm $\mathcal{A}$ adaptively makes $q$ $k$-parallel queries to the oracle. For $j \in \{1,2,\ldots,i\}$, the $j^{\text{th}}$ query
     is answered by $H_{j-1}$.
  9. After answering the $i^{\text{th}}$ query of $\mathcal{A}$, sample $x_{i+1},\ldots,x_{q+1} \xleftarrow{\$} \{0,1\}^\kappa$. Let $y_i := z_{i+1} \oplus x_{i+1},\ldots,y_q :=$
     $z_{q+1} \oplus x_{q+1}$.
 10. Let $\mathsf{s} := (x_0,x_1,\ldots,x_{q+1})$ and define the oracle
         $H_q := H\begin{bmatrix} x_q \mapsto y_q, \\ x_{q-1}\mapsto y_{q-1}, \\ \vdots \\ x_0 \mapsto y_0 \end{bmatrix}$
 11. All the remaining queries of $\mathcal{A}$ are answered by $H_q$.
 12. The algorithm $\mathcal{A}$ outputs $\mathsf{s}' = (x'_0,x'_1,\ldots,x'_{q+1})$.
 13. Output 1 if and only if $\mathsf{s} = \mathsf{s}'$.
```

First, we show that $\Pr_{\mathcal{H}_i}[\langle \mathcal{C},\mathcal{A}\rangle(1^\kappa,q)=1] = \Pr_{\mathcal{H}'_i}[\langle \mathcal{C},\mathcal{A}\rangle(1^\kappa,q)=1]$. Let us consider $\mathcal{H}_i$. For the first i queries, the oracles $H_0,\ldots,H_{i-1}$ contain no information about $x_i,y_i,\ldots,x_q,y_q,x_{q+1}$. Hence, right before the $(i+1)^{\text{th}}$ query of $\mathcal{A}$, each of the last $(q-i+1)$ coordinates of the puzzle P, i.e., $y_i \oplus x_{i+1},\ldots,y_q \oplus x_{q+1}$, is simply the XOR of two fresh, uniform bitstrings. Hence, we can instead sample independent, uniform $z_{i+1},\ldots,z_{q+1}$ and define the consistent oracle H_q and solution s afterward.

Next, let us consider $\mathcal{H}'_i$ and $\mathcal{H}'_{i+1}$. Fix $H,x_0,y_0,\ldots,x_{i-1},y_{i-1},x_i$ and $z_{i+1},\ldots,z_{q+1}$ in $\mathcal{H}'_i$. Right before $\mathcal{A}$'s $(i+1)^{\text{th}}$ query, the states of $\mathcal{A}$ in $\mathcal{H}'_i$ and $\mathcal{H}'_{i+1}$ are the same. After then, the only difference between $\mathcal{H}'_i$ and $\mathcal{H}'_{i+1}$ is the oracle used to answer the $(i+1)^{\text{th}}$ query, which is either H_q or H_i. The oracles H_q and H_i could possibly differ only on $x_{i+1},\ldots,x_q$ due to the order of reprogramming. Let $|\psi_i\rangle$ be the state of $\mathcal{A}$ right before the $(i+1)^{\text{th}}$ query in both $\mathcal{H}'_i$ and $\mathcal{H}'_{i+1}$. Because $x_{i+1}, \ldots, x_q$ are sampled *after* the first i queries are made, the state $|\psi_i\rangle$ is independent of $x_{i+1}, \ldots, x_q$. Therefore, by a standard quantum hybrid technique[9] [BBBV97, AHU19], we have

$$\left| \Pr_{\mathcal{H}'_i}[\langle \mathcal{C}, \mathcal{A}\rangle(1^\kappa, q) = 1] - \Pr_{\mathcal{H}'_{i+1}}[\langle \mathcal{C}, \mathcal{A}\rangle(1^\kappa, q) = 1] \right| \leq O\left(\sqrt{\frac{k(q-i)}{2^\kappa}}\right).$$

In particular, we use the fact that any quantum algorithm that makes 1 k-parallel query can distinguish whether or not an oracle has been reprogrammed on at most S i.i.d uniform points with advantage $O(\sqrt{kS/2^\kappa})$. □

The following lemma states that the probability of $\mathcal{A}$ outputting the correct x_{q+1} in Hybrid $\mathcal{H}_q$ is no better than a random guess.

Lemma 5.5. $\Pr_{\mathcal{H}_q}[\langle \mathcal{C}, \mathcal{A}\rangle(1^\kappa, q) = 1] \leq 2^{-\kappa}$.

Proof. Consider the experiment $\mathcal{H}'_q$ defined in the proof of Lemma 5.4. For convenience, we write the description of $\mathcal{H}'_q$ explicitly in the following.

```
Hybrid $\mathcal{H}'_q$ :
1. Sample $H \xleftarrow{\$} \mathsf{Func}(\kappa, \kappa)$.
2. Sample $x_0, x_1, \ldots, x_{q-1}, x_q \xleftarrow{\$} \{0,1\}^\kappa$.        // $x_{q+1}$ is not sampled.
3. Sample $y_0, y_1, \ldots, y_{q-1} \xleftarrow{\$} \{0,1\}^\kappa$.        // $y_q$ is not sampled.
4. Define a series of oracles for $j \in \{0, 1, \ldots, q-1\}$:
       $H_j := H \begin{bmatrix} x_j \mapsto y_j, \\ x_{j-1} \mapsto y_{j-1}, \\ \vdots \\ x_0 \mapsto y_0 \end{bmatrix}.$
5. Sample $z_{q+1} \xleftarrow{\$} \{0,1\}^\kappa$.
6. Set P := $(x_0, y_0 \oplus x_1, \ldots, y_{q-1} \oplus x_q, z_{q+1})$.        // s is not defined yet.
7. Send P to the algorithm $\mathcal{A}$.
8. The algorithm $\mathcal{A}$ adaptively makes $q$ $k$-parallel queries to the oracle, where the queries are answered in the
   following way. For $j \in \{1, 2, \ldots, q\}$, the $j^{\text{th}}$ query is answered by $H_{j-1}$.
9. The algorithm $\mathcal{A}$ outputs s$' = (x'_0, x'_1, \ldots, x'_{q+1})$.
10. Sample $x_{q+1} \xleftarrow{\$} \{0,1\}^\kappa$.
11. Set $y_q := z_{q+1} \oplus x_{q+1}$ and s := $(x_0, \ldots, x_{q+1})$.
12. Output 1 if and only if s = s$'$, and 0 otherwise.
```

Observe that x_{q+1} is independently, uniformly sampled *after* $\mathcal{A}$ outputs the solution s$'$, so we have

$$\Pr_{\mathcal{H}_q}[\langle \mathcal{C}, \mathcal{A}\rangle(1^\kappa, q) = 1] \leq \Pr_{\mathcal{H}'_q}[x'_{q+1} = x_{q+1}] = 2^{-\kappa}.$$

□

[9] We note that the technique in [BBBV97] can be extended to the parallel-query setting easily as shown in [AHU19].

Putting Lemma 5.5 and Lemma 5.4 together, we have

$$\Pr_{\mathcal{H}_{real}}[\langle \mathcal{C}, \mathcal{A}\rangle(1^\kappa, q) = 1] \leq \sum_{i=0}^{q-1} O\left(\sqrt{\frac{k(q-i)}{2^\kappa}}\right) + O\left(\frac{q^2}{2^\kappa}\right) + 2^{-\kappa}$$
$$= O\left(\sqrt{\frac{kq^3}{2^\kappa}}\right).$$

This finishes the proof. □

6 Barriers for Classical Attacks on Fully Quantum Puzzles

In this section, we present a fully quantum - i.e., quantum generator and quantum solver (QGQS) - time-lock puzzle construction that is secure against polynomial-query classical adversaries, assuming the quantum simulation conjecture does not hold.

Simulation Conjecture. Let $A^{(\cdot)}$ be a quantum oracle algorithm that outputs a single bit. For every fixed oracle H, let $p(A^H) := \Pr[1 \leftarrow A^H(1^\kappa)])$, where the probability is over the execution of A. We say an algorithm B λ-approximates an algorithm A if:

$$\mathop{\mathbb{E}}_{H}[|p(A^H) - p(B^H)|] \leq \lambda$$

The following is a weaker (asymptotic) version of the folklore Simulation Conjecture, which is stated as Conjecture 4 in [AA14].

Conjecture 6.1 (Quantum Polynomial-Query Simulation Conjecture)**.** For any constant c, there exists a constant d, such that for all κ^c-query quantum algorithm $Q^{(\cdot)}(\cdot)$, there exists a deterministic κ^d-query classical algorithm $A^{(\cdot)}(\cdot)$, such that $A(1^\kappa)$ κ^{-c}-approximates $Q(1^\kappa)$ for sufficiently large κ (when accessing a random oracle).

We now formally define the main result of this section, which is a fully quantum time-lock puzzle that cannot be broken by classical-query adversaries, assuming that the simulation conjecture is false.

Theorem 6.2 (Classically breaking QGQS TLPs implies the Simulation Conjecture)**.** *If Conjecture 6.1 does not hold, there exists an infinite set $\mathcal{K}$ (of security parameters κ) such that there is a protocol between two* quantum *oracle algorithms* Gen *and* Sol *that (quantumly) access a random oracle H satisfying the following:*

- ***Completeness:*** $\Pr[\mathrm{s}' = \mathrm{s} : (\mathrm{P}, \mathrm{s}) \leftarrow \mathsf{Gen}^H, \mathrm{s}' \leftarrow \mathsf{Sol}^H(\mathrm{P})] \geq 1 - \mathrm{negl}(\kappa)$.
- ***Soundness:*** *For any computationally unbounded classical adversary $\mathcal{A}$ who ask* $\mathrm{poly}(\kappa)$ classical *queries to H, and for every $\kappa \in \mathcal{K}$, we have*
$$\Pr[\mathrm{s}'' = \mathrm{s} : (\mathrm{P}, \mathrm{s}) \leftarrow \mathsf{Gen}^H, \mathrm{s}'' \leftarrow \mathcal{A}^H(\mathrm{P})] \leq \mathrm{negl}(\kappa).$$

The exact form of Conjecture 6.1 first appeared in [ACC+22], using which they proved the following lemma.

Lemma 6.3 (Weak Key Agreement with One-Way Communication [ACC+22])**.** *If Conjecture 6.1 does not hold, there exists an infinite set $\mathcal{K}$ (of security parameters κ), such that for all polynomially small $\varepsilon_0 = 1/\mathrm{poly}(\cdot)$ there is a protocol between two* quantum *oracle algorithms Q_A and Q_B that (quantumly) access a random oracle H satisfying the following:*

- ***One-way communication:*** *Q_{A}^H sends a single classical message c to Q_{B}^H, after which they each output* $\mathsf{key}_{\mathsf{A}}$, $\mathsf{key}_{\mathsf{B}}$.
- ***Completeness:*** $\Pr[\mathsf{key}_{\mathsf{A}} = \mathsf{key}_{\mathsf{B}} : (\mathsf{key}_{\mathsf{A}}, c) \leftarrow Q_{\mathsf{A}}^H, \mathsf{key}_{\mathsf{B}} \leftarrow Q_{\mathsf{B}}^H(c)] \geq 1 - \varepsilon_0(\kappa)$.
- ***Soundness:*** *For any computationally unbounded classical adversary $\mathcal{A}$ who ask* $\mathrm{poly}(\kappa)$ classical *queries to H, and for every $\kappa \in \mathcal{K}$, we have*
$$\Pr[\mathsf{key}_{\mathsf{A}} = \mathsf{key}_{\mathsf{E}} : (\mathsf{key}_{\mathsf{A}}, c) \leftarrow Q_{\mathsf{A}}^H, \mathsf{key}_{\mathsf{E}} \leftarrow \mathcal{A}^H(c)] \leq \delta_0(\kappa),$$
where $\delta_0(\cdot)$ is a negligible function.

Outline of the technical steps. To construct a fully quantum time-lock puzzle we first show how to amplify the key agreement of Lemma 6.3 to a key agreement with negligible completeness and soundness error. Then we will show how to transform such a key agreement protocol into a time-lock puzzle. Here is a sketch of this process, where all the items below are with *one-way communication* (OWC).

Step 0 Start with the weak key agreement with OWC of Lemma 6.3.

Step 1 Construct a weakly complete and strongly sound (WCSS) single-bit key agreement with OWC. (See Construction 6.4.)

Step 2 Construct a weakly complete and strongly sound multi-bit key agreement with OWC. (See Construction 6.7.)

Step 3 Construct a strongly complete and strongly sound (SCSS) multi-bit key agreement with OWC. (See Construction 6.10.)

Step 4 Transform a multi-bit key agreement with OWC to a time-lock puzzle while preserving the soundness and completeness error. (See Construction 6.13)

Here we show how to do **Step 1**.

Construction 6.4 (WCSS Single-Bit Key Agreement with OWC)**.** Let $Q_{\mathsf{A}_0}, Q_{\mathsf{B}_0}$ be a weak key agreement scheme from Lemma 6.3 with completeness error $\varepsilon_0(\kappa)$ and soundness error $\delta_0(\kappa)$. Let H be an oracle and define Q_{A_1} and Q_{B_1} as follows:

- Q_{A_1} performs the following:
  1. Run Q_{A_0} to get $(c_0, \mathsf{key}_{\mathsf{A}_0}) \leftarrow Q^H_{\mathsf{A}_0}$.
  2. Sample $r \leftarrow \{0,1\}^{|\mathsf{key}_{\mathsf{A}_0}|}$ and compute $\mathsf{key}_{\mathsf{A}_1} = \langle r, \mathsf{key}_{\mathsf{A}_0} \rangle$.
  3. Let $c_1 = c_0 || r$
- Q_{B_1} performs the following:
  1. Let $c_1 = c_0 || r$.
  2. Run Q_{B_0} to get $\mathsf{key}_{\mathsf{B}_0} \leftarrow Q^H_{\mathsf{B}_0}(c_0)$.
  3. Compute $\mathsf{key}_{\mathsf{B}_1} = \langle r, \mathsf{key}_{\mathsf{B}_0} \rangle$.

Theorem 6.5 (Completeness of Construction 6.4)**.** *If the construction from Lemma 6.3 has completeness error $\varepsilon_0(\kappa)$, then Construction 6.4 has completeness error $\varepsilon_1 \leq \varepsilon_0$.*

Proof. By the construction, if $\mathsf{key}_{\mathsf{A}_0} = \mathsf{key}_{\mathsf{B}_0}$, then $\mathsf{key}_{\mathsf{A}_1} = \mathsf{key}_{\mathsf{B}_1}$, thus $\varepsilon_1 \leq \varepsilon_0$. □

Theorem 6.6 (Soundness of Construction 6.4)**.** *If the construction from Lemma 6.3 has a soundness error $\delta_0(\kappa)$, then Construction 6.4 has a soundness error $\delta_1 \leq n^{O(1)} \cdot \delta_0^{\Omega(1)}$, where $n = |\mathsf{key}_{\mathsf{A}_0}|$. In particular, if $\delta_1 \leq \mathrm{negl}(\kappa)$, then $\delta_0 \leq \mathrm{negl}(\kappa)$ as well.*

Proof. The proof is similar to the proof of the Hard-core bit lemma from [GL89]. In particular, the proof of the Hard-core bit lemma from [GL89] is black-box and transforms any adversary who guesses the hard-core bit with probability ρ, to an inverting adversary that guesses the pre-image with probability $\mathrm{poly}(\rho/n)$, and the same reduction works even if the pre-image and image are jointly sampled (rather than the image being a deterministic function of the pre-image). □

Next, we show how to do **Step 2**.

Construction 6.7 (WCSS Multi-Bit Key Agreement with OWC)**.** Let $Q_{\mathsf{A}_1}, Q_{\mathsf{B}_1}$ be a WCSS Single-Bit key agreement scheme from Construction 6.4 with completeness error $\varepsilon_1(\kappa)$ and soundness error $\delta_1(\kappa)$. Let $u = \mathrm{poly}(\kappa)$ (to be chosen later) be the length of the key, H be an oracle, and define Q_{A_2} and Q_{B_2} as follows:

- Q_{A_2} performs the following:
  1. Divide oracle H to u independent oracles H_i (according to some canonical division).
  2. For $i \in [u]$ run Q_{A_1} to get $(c_{1,i}, \mathsf{key}_{\mathsf{A}_1,i}) \leftarrow Q_{\mathsf{A}_1}^{H_i}$.
  3. Let $c_2 = c_{1,1}|| \cdots ||c_{1,u}$ and $\mathsf{key}_{\mathsf{A}_2} = \mathsf{key}_{\mathsf{A}_1,1}|| \cdots ||\mathsf{key}_{\mathsf{A}_1,u}$.
- Q_{B_2} performs the following:
  1. Let $c_2 = c_{1,1}|| \cdots ||c_{1,u}$.
  2. Divide oracle H to u independent oracles H_i (according to the same canonical division).
  3. For $i \in [u]$ run Q_{B_1} to get $\mathsf{key}_{\mathsf{B}_1,i} \leftarrow Q_{\mathsf{B}_1}^{H_i}(c_{1,i})$.
  4. Compute $\mathsf{key}_{\mathsf{B}_2} = \mathsf{key}_{\mathsf{B}_1,1}|| \cdots ||\mathsf{key}_{\mathsf{B}_1,u}$.

Theorem 6.8 (Completeness of Construction 6.7). *If Construction 6.4 has completeness error $\varepsilon_1(\kappa)$, then Construction 6.7 has completeness error $\varepsilon_2 \leq u \cdot \varepsilon_1$.*

Proof. By the construction, if for all $i \in [u]$, $\mathsf{key}_{\mathsf{A}_1,i} = \mathsf{key}_{\mathsf{B}_1,i}$, then $\mathsf{key}_{\mathsf{A}_2} = \mathsf{key}_{\mathsf{B}_2}$. We conclude the proof by a union bound and letting $\varepsilon_2 \leq u\varepsilon_1$. □

Theorem 6.9 (Soundness of Construction 6.7). *If Construction 6.4 has a soundness error $\delta_1(\kappa)$, then Construction 6.7 has a soundness error (advantage) $\delta_2 = u\delta_1$.*

Proof. Suppose there exists an adversary Q_{E_2} such that:

$$|\Pr[1 \leftarrow Q_{\mathsf{E}_2}^H(c_2, \mathsf{key}_{\mathsf{A}_2})] - \Pr[1 \leftarrow Q_{\mathsf{E}_2}^H(c_2, U_u)]| \geq \delta_2.$$

We show there exists an adversary Q_{E_1} that breaks the soundness in Theorem 6.6 with an advantage at least δ_1. To do so define hybrids $\{\mathcal{H}_k\}_k$ for $k \in \{0, \cdots, u\}$ (where $\mathcal{H}_0$ is the experiment of Construction 6.7) such that in hybrid $\mathcal{H}_k$, we replace $\mathsf{key}_{\mathsf{A}_1,i}$ with $\mathsf{key}'_{\mathsf{A}_1,i} \leftarrow \{0,1\}$ when constructing $\mathsf{key}_{\mathsf{A}_2}$. Namely, in hybrid $\mathcal{H}_k$, we have $\mathsf{key}_{\mathsf{A}_2} = \mathsf{key}'_{\mathsf{A}_1,1}|| \cdots ||\mathsf{key}'_{\mathsf{A}_1,k}||\mathsf{key}_{\mathsf{A}_1,k+1}|| \cdots ||\mathsf{key}_{\mathsf{A}_1,u}$. Now note that in hybrid $\mathcal{H}_u$, $\mathsf{key}_{\mathsf{A}_2}$ is completely random, thus an adversary has no advantage in distinguishing $\mathsf{key}_{\mathsf{A}_2}$ from random. Therefore, if Q_{E_2} exists, then there is an index $k^* \in [u]$ s.t. there is a computationally unbounded classical adversary Q_{E_2,k^*} that can distinguish hybrids $\mathcal{H}_{k^*-1}$ and $\mathcal{H}_{k^*}$ with an advantage at least $\delta_2/u = \delta_1$. I.e., there is Q_{E_2,k^*} such that:

$$|\Pr[1 \leftarrow Q_{\mathsf{E}_2,k^*}^H(c_2, \mathsf{key}_{\mathsf{A}_2}^{k^*-1})] - \Pr[1 \leftarrow Q_{\mathsf{E}_2,k^*}^H(c_2, \mathsf{key}_{\mathsf{A}_2}^{k^*})]| \geq \delta_1,$$

where $\mathsf{key}_{\mathsf{A}_2}^k$ is the corresponding values of $\mathsf{key}_{\mathsf{A}_2}$ in hybrid $\mathcal{H}_k$. Now construct Q_{E_1} on input (c''_1, b'') and given access to oracle H'' as follows:

1. For $i \in [u]/\{k^*\}$ sample oracles H_i'', and compute $(c''_{1,i}, \mathsf{key}''_{\mathsf{A}_1,i}) \leftarrow Q_{\mathsf{A}_1}^{H_i''}$.
2. Let $(c''_{1,k^*}, \mathsf{key}''_{\mathsf{A}_1,k^*}) = (c''_1, b'')$.
3. Let $c''_2 = c''_{1,1}|| \cdots ||c''_{1,u}$ and $\mathsf{key}''_{\mathsf{A}_2} = U_1|| \cdots ||U_1||\mathsf{key}''_{\mathsf{A}_1,k^*}|| \cdots ||\mathsf{key}''_{\mathsf{A}_1,u}$.
4. Send $(c''_2, \mathsf{key}''_{\mathsf{A}_2})$ to Q_{E_2,k^*}.
5. Answer Q_{E_2,k^*}'s queries on oracle i using H'' for $i = k^*$, and H_i'' otherwise.
6. Output whatever Q_{E_2,k^*} outputs.

Now note that $(c''_2, \mathsf{key}''_{\mathsf{A}_2})$ perfectly simulates $(c_2, \mathsf{key}_{\mathsf{A}_2}^{k^*-1})$ if $b'' = \mathsf{key}_{\mathsf{A}_1}$, and perfectly simulates $(c_2, \mathsf{key}_{\mathsf{A}_2}^{k^*})$ if $b'' \leftarrow \{0,1\}$. Thus Q_{E_1} perfectly simulates the security experiment for Q_{E_2,k^*}, therefore, has the same advantage δ_1. □

Now we show how to do **Step 3**.

Construction 6.10 (SCSS Multi-Bit Key Agreement with OWC)**.** Let $Q_{\mathsf{A}_2}, Q_{\mathsf{B}_2}$ be a WCSS Multi-Bit key agreement scheme from Construction 6.7. Let $t = \text{poly}(\kappa)$ (to be chosen later), H be an oracle, and define Q_{A_3} and Q_{B_3} as follows:

- Q_{A_3} performs the following:
  1. Divide oracle H to t independent oracles H_i (according to some canonical division).
  2. For $i \in [t]$ run Q_{A_2} to get $(c_{2,i}, \mathsf{key}_{\mathsf{A}_2,i}) \leftarrow Q_{\mathsf{A}_2}^{H_i}$.
  3. Sample $\mathsf{key}_{\mathsf{A}_3} \leftarrow \{0,1\}^t$.
  4. For $i \in [t]$ let $\mathsf{key}_i = \mathsf{key}_{\mathsf{A}_3} \oplus \mathsf{key}_{\mathsf{A}_2,i}$.
  5. Let $c_3 = c_{2,1}||\cdots||c_{2,t}||\mathsf{key}_1||\cdots||\mathsf{key}_t$.
- Q_{B_3} performs the following:
  1. Let $c_3 = c_{2,1}||\cdots||c_{2,t}||\mathsf{key}_1||\cdots||\mathsf{key}_t$.
  2. Divide oracle H to t independent oracles H_i (according to the same canonical division).
  3. For $i \in [t]$ run Q_{B_2} to get $\mathsf{key}_{\mathsf{B}_2,i} \leftarrow Q_{\mathsf{B}_2}^{H_i}(c_{2,i})$.
  4. For $i \in [t]$ compute $\mathsf{key}_{\mathsf{B}_3,i} \leftarrow \mathsf{key}_i \oplus \mathsf{key}_{\mathsf{B}_2,i}$.
  5. Compute $\mathsf{key}_{\mathsf{B}_3} = \text{maj}_i(\mathsf{key}_{\mathsf{B}_3,i})$.

Theorem 6.11 (Completeness of Construction 6.10)**.** *If Construction 6.7 has completeness error $\leq 1/4$, then Construction 6.10 has completeness error $\varepsilon_3 \leq 2^{-t/8}$.*

Proof. By the construction, if for at least $t/2$ of $i \in [t]$ we have $\mathsf{key}_{\mathsf{A}_2,i} = \mathsf{key}_{\mathsf{B}_2,i}$, then $\mathsf{key}_{\mathsf{A}_3} = \mathsf{key}_{\mathsf{B}_3}$. Since for each of the sub-prtocols the probability of $\mathsf{key}_{\mathsf{A}_2,i} = \mathsf{key}_{\mathsf{B}_2,i}$ is at least $3/4$, then by the Hoeffding inequality, the probaiblity of *not* having the correct key in at least $t/2$ of $i \in [t]$ is at most $e^{-2\sigma^2 t}$ for $\sigma = |1/2 - 1/4| = 1/4$, which implies the error to be at most $e^{-2t/16} < 2^{-t/8}$. □

Theorem 6.12 (Soundness of Construction 6.10)**.** *If Construction 6.7 has a soundness error $\delta_2(\kappa)$, then Construction 6.10 has a soundness error $\delta_3 = t\delta_2$.*

Proof. Suppose there exists an adversary Q_{E_3} such that:

$$|\Pr[1 \leftarrow Q_{\mathsf{E}_3}^H(c_3, \mathsf{key}_{\mathsf{A}_3})] - \Pr[1 \leftarrow Q_{\mathsf{E}_3}^H(c_3, U_u)]| \geq \delta_3.$$

We show there exists an adversary Q_{E_2} that breaks the soundness in Theorem 6.9 with an advantage at least δ_2. To do so define hybrids $\{\mathcal{H}_k\}_k$ for $k \in \{0, \cdots, t\}$ (where $\mathcal{H}_0$ is the experiment of Construction 6.10) such that in hybrid $\mathcal{H}_k$, we replace key_i with $\mathsf{key}'_i = \mathsf{key}_{\mathsf{A}_3} \oplus \mathsf{key}'_{\mathsf{A}_2,i}$ when constructing c_3. Namely, in hybrid $\mathcal{H}_k$, we have $c_3 = c_{2,1}||\cdots||c_{2,t}||\mathsf{key}'_1||\cdots||\mathsf{key}'_k||\mathsf{key}_{k+1}||\cdots||\mathsf{key}_t$. Now note that in hybrid $\mathcal{H}_t$, there is no information about $\mathsf{key}_{\mathsf{A}_3}$ in c_3, thus an adversary has no advantage in distinguishing $\mathsf{key}_{\mathsf{A}_3}$ from random. Therefore, if Q_{E_3} exists, then there is an index $k^* \in [t]$ s.t. there is a computationally unbounded classical adversary Q_{E_3,k^*} that can distinguish hybrids $\mathcal{H}_{k^*-1}$ and $\mathcal{H}_{k^*}$ with an advantage at least $\delta_3/t = \delta_2$. I.e., there is Q_{E_3,k^*} such that:

$$|\Pr[1 \leftarrow Q_{\mathsf{E}_3,k^*}^H(c_3^{k^*-1}, \mathsf{key}_{\mathsf{A}_3})] - \Pr[1 \leftarrow Q_{\mathsf{E}_3,k^*}^H(c_3^{k^*}, \mathsf{key}_{\mathsf{A}_3})]| \geq \delta_2,$$

where c_3^k is the corresponding values of c_3 in hybrid $\mathcal{H}_k$. Now construct Q_{E_2} on input (c''_2, x'') and given access to oracle H'' as follows:

1. For $i \in [t]/\{k^*\}$ sample oracles H_i'', and compute $(c''_{2,i}, \mathsf{key}''_{\mathsf{A}_2,i}) \leftarrow Q_{\mathsf{A}_2}^{H_i''}$.
2. Let $(c''_{2,k^*}, \mathsf{key}''_{\mathsf{A}_2,k^*}) = (c''_2, x'')$.

3. Sample a random $\mathsf{key}''_{\mathsf{A}_3} \leftarrow \{0,1\}^u$.
4. For $i \in [t]$ let $\mathsf{key}''_i = \mathsf{key}''_{\mathsf{A}_3} \oplus \mathsf{key}''_{\mathsf{A}_2,i}$.
5. Let $c''_3 = c''_{2,1}||\cdots||c''_{2,t}||U_u||\cdots||U_u||\mathsf{key}''_{k^*}||\cdots||\mathsf{key}''_t$.
6. Send $(c''_3, \mathsf{key}''_{\mathsf{A}_3})$ to Q_{E_3,k^*}.
7. Answer Q_{E_3,k^*}'s queries on oracle i using H'' for $i = k^*$, and H_i'' otherwise.
8. Output whatever Q_{E_3,k^*} outputs.

Now note that c_3 perfectly simulates $c_3^{k^*-1}$ if $x'' = \mathsf{key}_{\mathsf{A}_2}$, and perfectly simulates $c_3^{k^*}$ if $x'' \leftarrow \{0,1\}^u$. Thus Q_{E_2} perfectly simulates the security experiment for Q_{E_3,k^*}, therefore, has the same advantage δ_2.

□

Finally, we show how to do **Step 4**.

Construction 6.13 (QGQS Time-Lock Puzzle From Key Agreement with OWC)**.** Let $Q_{\mathsf{A}_3}, Q_{\mathsf{B}_3}$ be an SCSS Multi-Bit key agreement scheme from Construction 6.10 with completeness error $\varepsilon_3(\kappa)$ and soundness error $\delta_3(\kappa)$. Let H be an oracle, and define Gen and Sol as follows:

- Gen performs the following:
  1. Run Q_{A_3} to get $(c_3, \mathsf{key}_{\mathsf{A}_3}) \leftarrow Q^H_{\mathsf{A}_3}$.
  2. Let $\mathrm{P} = c_3$ and $\mathrm{s} = \mathsf{key}_{\mathsf{A}_3}$.
- Sol performs the following:
  1. Let $\mathrm{P} = c_3$.
  2. Run Q_{B_3} to get $\mathsf{key}_{\mathsf{B}_3} \leftarrow Q^H_{\mathsf{B}_3}(c_3)$.
  3. Let $\mathrm{s}' = \mathsf{key}_{\mathsf{B}_3}$.

Theorem 6.14 (Completeness of Construction 6.13)**.** *If Construction 6.10 has completeness error $\varepsilon_3(\kappa)$, then Construction 6.13 has completeness error $\varepsilon_4 \leq \varepsilon_3$. Namely, we have:*

$$\Pr[\mathrm{s}' = \mathrm{s} : (\mathrm{P}, \mathrm{s}) \leftarrow \mathsf{Gen}^H, \mathrm{s}' \leftarrow \mathsf{Sol}^H(\mathrm{P})] \geq 1 - \varepsilon_4(\kappa).$$

Proof. By the construction, if we have $\mathsf{key}_{\mathsf{A}_3} = \mathsf{key}_{\mathsf{B}_3}$, then $\mathrm{s}' = \mathrm{s}$, thus $\varepsilon_4 \leq \varepsilon_3$. □

Theorem 6.15 (Soundness of Construction 6.13)**.** *If Construction 6.10 has a soundness error $\delta_3(\kappa)$, then Construction 6.13 has a soundness error $\delta_4 = \delta_3 + 2^{-u}$. Namely, we have:*

$$\Pr[\mathrm{s}'' = \mathrm{s} : (\mathrm{P}, \mathrm{s}) \leftarrow \mathsf{Gen}^H, \mathrm{s}'' \leftarrow Q^H_{\mathsf{E}_4}(\mathrm{P})] \leq \delta_4(\kappa).$$

Proof. Suppose there exists an adversary Q_{E_4} such that:

$$\Pr[\mathrm{s}'' = \mathrm{s} : (\mathrm{P}, \mathrm{s}) \leftarrow \mathsf{Gen}^H, \mathrm{s}'' \leftarrow Q^H_{\mathsf{E}_4}(\mathrm{P})] \geq \delta_4.$$

We show there exists an adversary Q_{E_3} that breaks the soundness in Theorem 6.12 with an advantage at least δ_3. Consider hybrids $\mathcal{H}_0$ and $\mathcal{H}_1$ where $\mathcal{H}_0$ is the output of Construction 6.13, and $\mathcal{H}_1$ is similar to $\mathcal{H}_0$ except that we replace $\mathrm{s} = \mathsf{key}_{\mathsf{A}_3}$ with $\mathrm{s}' \leftarrow \{0,1\}^u$. Note that in $\mathcal{H}_1$, no adversary can find the key with a better advantage than a random guess, so if Q_{E_4} exists, then there exists Q_{E_3} that distinguishes $\mathcal{H}_0$ and $\mathcal{H}_1$ with an advantage at least $\delta_4 - 2^{-u} = \delta_3$. Construct Q_{E_3} s.t. on input (c_3, x), send $(\mathrm{P}, \mathrm{s}) = (c_3, x)$ to Q'_{E_4} and output whatever Q_{E_4} outputs. Note that Q_{E_3} perfectly simulates $\mathcal{H}_0$ if $x = \mathsf{key}_{\mathsf{A}_3}$ and perfectly simulates $\mathcal{H}_1$ if $x \leftarrow \{0,1\}^u$. Thus, Q_{E_3} has the same advantage as Q'_{E_4}. □

Proof of Theorem 6.2. To prove this theorem we only need to determine the choice of parameters in Constructions 6.3, 6.4, 6.7, 6.10, and 6.13. Let κ be chosen according to $\mathcal{K}$ in Lemma 6.3. Then let $t = u = \kappa$, $\varepsilon_0 = 1/4\kappa$ and $\delta_0 = \mathrm{negl}(\kappa)$. Then by Theorem 6.5 $\varepsilon_1 \leq 1/4\kappa$, by Theorem 6.6 $\delta_1 \leq \mathrm{negl}(\kappa)$, by Theorem 6.8 $\varepsilon_2 \leq 1/4$, by Theorem 6.9 $\delta_2 \leq \mathrm{negl}(\kappa)$, by Theorem 6.11 $\varepsilon_3 \leq \mathrm{negl}(\kappa)$, by Theorem 6.12 $\delta_2 \leq \mathrm{negl}(\kappa)$, by Theorem 6.14 $\varepsilon_4 \leq \mathrm{negl}(\kappa)$, by Theorem 6.15 $\delta_4 \leq \mathrm{negl}(\kappa)$. Finally, note that ε_4 and δ_4 are the completeness and soundness of a QGQS time-lock puzzle. □

References

[AA14] Scott Aaronson and Andris Ambainis. The need for structure in quantum speedups, 2014. 5, 29

[ACC+22] Per Austrin, Hao Chung, Kai-Min Chung, Shiuan Fu, Yao-Ting Lin, and Mohammad Mahmoody. On the impossibility of key agreements from quantum random oracles. In *Advances in Cryptology–CRYPTO 2022: 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15–18, 2022, Proceedings, Part II*, pages 165–194. Springer, 2022. 5, 7, 8, 29

[AHU19] Andris Ambainis, Mike Hamburg, and Dominique Unruh. Quantum security proofs using semi-classical oracles. In *Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part II 39*, pages 269–295. Springer, 2019. 6, 8, 10, 28

[AHY23] Prabhanjan Ananth, Zihan Hu, and Henry Yuen. On the (im)plausibility of public-key quantum money from collision-resistant hash functions. Cryptology ePrint Archive, Paper 2023/069, 2023. `https://eprint.iacr.org/2023/069`. 9

[AK22] Prabhanjan Ananth and Fatih Kaleoglu. A note on copy-protection from random oracles. *arXiv preprint arXiv:2208.12884*, 2022. 9

[BBBF18] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In *Advances in Cryptology–CRYPTO 2018: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2018, Proceedings, Part I*, pages 757–788. Springer, 2018. 9

[BBBV97] Charles H Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM journal on Computing*, 26(5):1510–1523, 1997. 8, 10, 28

[BDF+11] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random oracles in a quantum world. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 41–69. Springer, Heidelberg, December 2011. 3

[BI87] Manuel Blum and Russell Impagliazzo. Generic oracles and oracle classes. In *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pages 118–126. IEEE, 1987. 7

[BKSY11] Zvika Brakerski, Jonathan Katz, Gil Segev, and Arkady Yerukhimovich. Limits on the power of zero-knowledge proofs in cryptographic constructions. In Yuval Ishai, editor, *Theory of Cryptography*, pages 559–578, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. 19

[BLZ21] Jeremiah Blocki, Seunghoon Lee, and Samson Zhou. On the security of proofs of sequential work in a post-quantum world. In *2nd Conference on Information-Theoretic Cryptography*, page 1, 2021. 9

[BN00] Dan Boneh and Moni Naor. Timed commitments. In Mihir Bellare, editor, *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 236–254. Springer, Heidelberg, August 2000. 3

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73. ACM Press, November 1993. 4

[CFHL21] Kai-Min Chung, Serge Fehr, Yu-Hsuan Huang, and Tai-Ning Liao. On the compressed-oracle technique, and post-quantum security of proofs of sequential work. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part II*, volume 12697 of *Lecture Notes in Computer Science*, pages 598–629. Springer, Heidelberg, October 2021. 9

[CP18] Bram Cohen and Krzysztof Pietrzak. Simple proofs of sequential work. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 451–467. Springer, Heidelberg, April / May 2018. 9

[GL89] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing*, STOC '89, page 25–32, New York, NY, USA, 1989. Association for Computing Machinery. 30

[HH87] Juris Hartmanis and Lane A. Hemaspaandra. One-way functions, robustness, and the non-isomorphism of np-complete sets. In *Symposium on Computation Theory*, 1987. 7

[HMST22] Iftach Haitner, Noam Mazor, Jad Silbak, and Eliad Tsfadia. On the complexity of two-party differential privacy. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1392–1405, 2022. 8

[HY20] Akinori Hosoyamada and Takashi Yamakawa. Finding collisions in a quantum world: quantum black-box separation of collision-resistance and one-wayness. In *Advances in Cryptology–ASIACRYPT 2020: 26th International Conference on the Theory and Application of Cryptology and Information Security, Daejeon, South Korea, December 7–11, 2020, Proceedings, Part I 26*, pages 3–32. Springer, 2020. 9

[JMRR21] Samuel Jaques, Hart Montgomery, Razvan Rosie, and Arnab Roy. Time-release cryptography from minimal circuit assumptions. In *Progress in Cryptology–INDOCRYPT 2021: 22nd International Conference on Cryptology in India, Jaipur, India, December 12–15, 2021, Proceedings 22*, pages 584–606. Springer, 2021. 9

[LPS17] Huijia Lin, Rafael Pass, and Pratik Soni. Two-round and non-interactive concurrent non-malleable commitments from time-lock puzzles. In Chris Umans, editor, *58th Annual Symposium on Foundations of Computer Science*, pages 576–587. IEEE Computer Society Press, October 2017. 3

[LW17] Arjen K Lenstra and Benjamin Wesolowski. Trustworthy public randomness with sloth, unicorn, and trx. *International Journal of Applied Cryptography*, 3(4):330–343, 2017. 9

[MMV11] Mohammad Mahmoody, Tal Moran, and Salil P. Vadhan. Time-lock puzzles in the random oracle model. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 39–50. Springer, Heidelberg, August 2011. 3, 4, 5, 6, 7, 11, 18, 24

[MMV13] Mohammad Mahmoody, Tal Moran, and Salil P. Vadhan. Publicly verifiable proofs of sequential work. In Robert D. Kleinberg, editor, *ITCS 2013: 4th Innovations in Theoretical Computer Science*, pages 373–388. Association for Computing Machinery, January 2013. 9

[MSW20] Mohammad Mahmoody, Caleb Smith, and David J. Wu. Can verifiable delay functions be based on random oracles? In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *ICALP 2020: 47th International Colloquium on Automata, Languages and Programming*, volume 168 of *LIPIcs*, pages 83:1–83:17. Schloss Dagstuhl, July 2020. 9

[NC10] Michael A Nielsen and Isaac L Chuang. Quantum computation and quantum information. *Cambridge University Press*, 2010. 9

[Nis89] Noam Nisan. Crew prams and decision trees. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 327–335, 1989. 7

[OSSS05] Ryan O'Donnell, Michael Saks, Oded Schramm, and Rocco A Servedio. Every decision tree has an influential variable. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, pages 31–39. IEEE, 2005. 5

[Pie19] Krzysztof Pietrzak. Simple verifiable delay functions. In Avrim Blum, editor, *ITCS 2019: 10th Innovations in Theoretical Computer Science Conference*, volume 124, pages 60:1–60:15. LIPIcs, January 2019. 9

[RSW96] Ronald L Rivest, Adi Shamir, and David A Wagner. Time-lock puzzles and timed-release crypto. *Massachusetts Institute of Technology. Laboratory for Computer Science*, 1996. 3

[Tar89] Gábor Tardos. Query complexity, or why is it difficult to separate $\mathsf{NP}^A \cap \mathsf{coNP}^A$ from P^A by random oracles A? *Combinatorica*, 9:385–392, 1989. 7

[Unr14] Dominique Unruh. Revocable quantum timed-release encryption. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 129–146. Springer, Heidelberg, May 2014. 3, 7, 10

[Wes19] Benjamin Wesolowski. Efficient verifiable delay functions. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part III*, volume 11478 of *Lecture Notes in Computer Science*, pages 379–407. Springer, Heidelberg, May 2019. 9

[Zha12] Mark Zhandry. Secure identity-based encryption in the quantum random oracle model. In *32nd Annual International Cryptology Conference, CRYPTO 2012*, pages 758–775, 2012. 7, 16

[Zha19] Mark Zhandry. How to record quantum queries, and applications to quantum indifferentiability. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part II*, volume 11693 of *Lecture Notes in Computer Science*, pages 239–268. Springer, Heidelberg, August 2019. 20

A Omitted Proof

A.1 The description of the extractor in Lemma 2.3

We give a proof and the description of Ext for completeness.

Proof of Lemma 2.3. Define the algorithm $\mathsf{Ext}(\mathcal{A}^H(z))$ as follows:

- Pick $i \xleftarrow{\$} [q]$.
- Run $\mathcal{A}^H(z)$ until (right before) the i^{th} query.
- Measure the query register of $\mathcal{A}^H(z)$ in the computational basis to obtain the outcome $x \in \mathcal{X}$.
- Output x.

The probability that $\mathsf{Ext}(\mathcal{A}^H(z))$ successfully outputs $x \in \mathcal{S}$ is given by

$$\sum_{j=1}^{q} \Pr[i=j] \Pr[x \in \mathcal{S} \mid i=j : x \leftarrow |\psi_i^H\rangle] = \frac{1}{q}\sum_{j=1}^{q} \|\Pi_{\mathcal{S}}|\psi_j^H\rangle\|^2 = \frac{\mu(\mathcal{A}^H(z),\mathcal{S})}{q}.$$

□