On the Impossibility of Post-Quantum Black-Box Zero-Knowledge in Constant Rounds

Nai-Hui Chia[1], Kai-Min Chung[2], Qipeng Liu[3], and Takashi Yamakawa[*4]

[1]QuICS, University of Maryland nchia@umd.edu
[1]Luddy School of Informatics, Computing, and Engineering, Indiana University Bloomington
[2]Institute of Information Science, Academia Sinica kmchung@iis.sinica.edu.tw
[3]Princeton University qipengl@cs.princeton.edu
[4]NTT Secure Platform Laboratories takashi.yamakawa.ga@hco.ntt.co.jp

March 20, 2021

Abstract

We investigate the existence of constant-round post-quantum black-box zero-knowledge protocols for **NP**. As a main result, we show that there is no constant-round post-quantum black-box zero-knowledge argument for **NP** unless **NP** $\subseteq$ **BQP**. As constant-round black-box zero-knowledge arguments for **NP** exist in the classical setting, our main result points out a fundamental difference between post-quantum and classical zero-knowledge protocols. Combining previous results, we conclude that unless **NP** $\subseteq$ **BQP**, constant-round post-quantum zero-knowledge protocols for **NP** exist if and only if we use non-black-box techniques or relax certain security requirements such as relaxing standard zero-knowledge to ϵ-zero-knowledge. Additionally, we also prove that three-round and public-coin constant-round post-quantum black-box ϵ-zero-knowledge arguments for **NP** do not exist unless **NP** $\subseteq$ **BQP**.

[*]This work was done while the author was visiting Princeton University.

1 Introduction

Zero-knowledge (ZK) interactive proof, introduced by Goldwasser, Micali, and Rackoff [GMR89], is a fundamental primitive in cryptography. ZK protocols provide privacy to the prover by proving a statement without revealing anything except that the statement is true even though the verifier is malicious. After many decades of study, what languages ZK protocols can express is quite understood. There have been many positive results for ZK protocols for particular languages, including quadratic residuosity [GMR89], graph isomorphism [GMW91], statistical difference problem [SV03] etc., and for all **NP** languages assuming one-way functions [GMW91, Blu86].

In addition to the expressiveness, round complexity is an important complexity measure for ZK protocols. One fascinating question regarding ZK is whether languages in **NP** have constant-round ZK protocols. In this aspect, the ZK protocol for 3-coloring [GMW91] is not ideal since that requires super-constant number of rounds if we require negligible soundness error. (In the following, we require negligible soundness error by default.) Feige and Shamir [FS90] and Brassard et al. [BCY91] presented constant-round ZK arguments[1] for **NP**. Then, under reasonable cryptographic assumptions, Goldreich and Kahan [GK96a] gave the first constant-round ZK proof[2] for **NP**.

On the other hand, generalizing above results to obtain ZK protocols against malicious quantum verifiers is nontrivial. Briefly speaking, a protocol is ZK if there exists an efficient simulator such that for all malicious verifiers, the simulator can simulate the view generated by the prover and the malicious verifier. In the quantum setting, the malicious verifier can have quantum auxiliary input and can use quantum algorithms, which gives the verifier additional power to cheat even though the simulator is also quantum. This difference fails the security proofs of previous classical results. Specifically, those security proofs rely on a technique called *rewinding*, enabling the simulator to complete the simulation by using only black-box access to the malicious verifier. This rewinding technique often cannot be applied when an adversary is quantum due to the no-cloning theorem.

Watrous [Wat09] presented the first classical ZK protocol against malicious quantum verifiers for languages in **NP**. For simplicity, we call such a protocol *post-quantum ZK protocol*. In particular, he introduced the quantum rewinding lemma and showed that, given black-box access to the malicious quantum verifier, there exists a quantum simulator assuming quantum-secure one-way functions. However, to achieve negligible soundness, Watrous's protocol needs super-constant number of rounds. Therefore, it does not satisfy the constant-round requirement.

Recently, Bitansky and Shmueli [BS20] gave the first constant-round post-quantum ZK argument for **NP** assuming Quantum Learning with Error (QLWE) and Quantum Fully Homomorphic Encryption (QFHE) assumptions. However, their result relies on a novel technique for non-black-box simulation, i.e., the simulator requires the actual description of the malicious verifier instead of using it in a black-box manner. Along this line, Chia et al. [CCY20a] presented a constant-round black-box ϵ-ZK (BB ϵ-ZK) argument for **NP** assuming quantum-secure one-way functions and a constant-round BB ϵ-ZK proof for **NP** assuming QLWE (or more generally, the existence of collapsing hash function [Unr16]). ϵ-ZK is a security notion weaker than standard ZK. Roughly speaking, while standard ZK requires that the simulator can only fail with negligible probability, ϵ-ZK allows the simulator to run in time $\mathsf{poly}(1/\epsilon)$ with failing probability at most ϵ.

Nevertheless, all these results in [BS20, CCY20a, Wat09] cannot achieve constant-round post-quantum black-box ZK (BBZK) for **NP**. In contrast, constant-round classical ZK protocols can be obtained by black-box simulation [FS90, GK96a, PW09, BCY91]. Observing this inconsistency between classical and quantum settings, one may start wondering if non-black-box simulation is

[1] The protocol only guarantees to be computationally sound.

[2] The protocol has statistical soundness.

necessary for post-quantum ZK or if we really need to sacrifice ZK security for black-box simulation. In this work, we aim to satisfy all these curiosities by answering the following question:

Do there exist constant-round post-quantum BBZK protocols for **NP***?*

Classical impossibility results. In the classical setting, certain constant-round BBZK protocols are unlikely to exist. Goldreich and Krawczyk [GK96b] showed that there do not exist three-round BBZK protocols and public-coin constant-round BBZK protocols for **NP** unless **NP** $\subseteq$ **BPP**. Barak and Lindell [BL02] proved that there is no constant-round BBZK protocol with strict-polynomial-time simulation unless **NP** $\subseteq$ **BPP**.[3] We note that a simulator is allowed to run in expected-polynomial-time in the standard definition of the ZK property, which we also follow. Indeed, all known constant-round BBZK protocols for **NP** rely on expected-polynomial-time simulation to circumvent the above impossibility result.

1.1 Our Results

In this work, we give a negative answer to the above question. In particular, we show that

Theorem 1.1. *There do not exist constant-round post-quantum BBZK protocols for* **NP** *unless* **NP** $\subseteq$ **BQP**.

We stress that Theorem 1.1 rules out constant-round post-quantum BBZK protocols with *expected-polynomial-time* simulation. This indicates a fundamental difference between classical and post-quantum BBZK. That is, although there exist constant-round classical BBZK protocols (with expected-polynomial-time simulation) for **NP**, such a protocol does not exist in the quantum setting unless **NP** $\subseteq$ **BQP**.

Along this line, to fully understand the feasibility of various constant-round post-quantum ZK protocols for **NP**, we also prove other impossibility results.

Theorem 1.2. *There do not exist constant-round public-coin post-quantum BB ϵ-ZK protocols for* **NP** *unless* **NP** $\subseteq$ **BQP**.

Theorem 1.3. *There do not exist three-round post-quantum BB ϵ-ZK protocols for* **NP** *unless* **NP** $\subseteq$ **BQP**.

In summary, combining previous works on post-quantum ZK, we are able to give detailed characterizations of the feasibility of constant-round post-quantum BBZK and BB ϵ-ZK protocols for **NP**. We summarize them as follows:

1. For post-quantum ZK, non-black-box simulation is sufficient [BS20] and necessary (Theorem 1.1) for constant round. Otherwise, one has to relax the security of ZK to ϵ-ZK for black-box simulation [CCY20a].

2. For post-quantum BB ϵ-ZK, private coin is sufficient [CCY20a] and necessary (Theorem 1.2) for constant round.

3. In the last, although it is unlikely to have three-round post-quantum BB ϵ-ZK protocols for **NP** by Theorem 1.3, there exists a five-round post-quantum BB ϵ-ZK protocol when assuming QLWE [CCY20a]. Whether there exists a four-round post-quantum BB ϵ-ZK protocol for **NP** is still open.

[3] A ZK protocol has strict-polynomial-time simulation if the simulator always runs in a fixed polynomial time.

1.2 Technical Overview

1.2.1 Impossibility of Constant-Round ZK

In this section, we start by recalling the classical impossibility result by Barak and Lindell [BL02] and provide overviews for our techniques that extend the classical impossibility to the quantum setting as well as expected polynomial time quantum simulator.

We first fix some notations that will be used in this overview. Let (P, V) be a (classical or post-quantum) zero-knowledge proof or argument system for a language L with a BB simulator Sim, with perfect completeness and the number of rounds being a constant number $2k-1$, where the first message is sent by P. We can assume all messages sent by the prover P or the verifier V are elements in a classical set $\mathcal{M}$.

Barak-Lindell Impossibility Result. As observed by Barak and Lindell [BL02], to construct constant-round zero-knowledge proofs or arguments for language not in **BPP**, one has to either allow *expected* polynomial time simulators or *non-black-box* simulators (that make inherent use of the code of the verifier). More precisely, they show that all languages that have constant round zero-knowledge proofs or arguments with strict polynomial-time black-box (BB) simulators must be trivial, i.e. in **BPP**. We give the proof sketch below and explain the potential barriers for quantizing this proof.

The simulator Sim with oracle access to a (dishonest) verifier V^* uses $V^*(x, \mathsf{aux}, \cdot)$ as a black-box routine. Here $V^*(x, \mathsf{aux}, \cdot)$ is the next-message function of V^*, which takes a statement x, an auxiliary input aux, a random tape r and a message transcript $\boldsymbol{m} = (m_1, \cdots, m_i)$ (all messages sent by P, of length at most k), and outputs the next message sent to the prover.

To show an algorithm that decides L, we first define a random aborting verifier V^*. V^* works almost in the same way as V, except on each input transcript $\boldsymbol{m}$, it refuses to answer and aborts with some probability. In other words, let H be a random oracle that independently on each input transcript $\boldsymbol{m}$ of length $i \leq k$, outputs 0 (aborting) with probability $1-\epsilon_i$ and 1 (non-aborting) with some non-negligible probability ϵ_i [4]; then V^* works in the same way as V if $H(\boldsymbol{m}_j) = 1$ for all prefix $\boldsymbol{m}_j$ of $\boldsymbol{m}$ and aborts otherwise. Note that H is treated as the auxiliary input feed to V^*.

We are now ready to define an algorithm $\mathcal{B}$ that decides L: on input x, it samples a random tape r and a random aborting oracle H, then runs $\mathsf{Sim}(x)$ with oracle access to $V^*(x, \mathsf{aux} := (H, r), \cdot)$; $\mathcal{B}$ outputs 1 (accepts) if and only if the simulator outputs an accepting transcript.

The proof consists of two parts:

- On $x \in L$, $\mathcal{B}$ accepts with non-negligible probability.
- On $x \notin L$, $\mathcal{B}$ accepts with negligible probability.

The first bullet point is easier, simply invoking zero-knowledge property. For the second bullet point, a more delicate argument is needed. The core of the proof is to turn $\mathcal{B}$ into a cheating prover that tries to prove a statement x which is not in the language L.

For $x \in L$, the simulator will output the same transcript distribution as the distribution induced by the interaction between the honest prover P and the random aborting verifier V^*. When V^* aborts, it never gives an accepting transcript. When V^* never aborts, the transcript is always an accepting one, by the perfect completeness of the underlying proof system $\Pi = (P, V)$. Since in each round V^* aborts with probability $1-\epsilon_i$, the probability that V^* never aborts in the execution

[4] In the actual proof by Barak and Lindell, the probability ϵ_i is set as ϵ^{2^i} for some chosen ϵ.

is $\epsilon^* = \prod_i \epsilon_i$. By zero-knowledge property, the simulator will output an accepting transcript with probability roughly ϵ^*, which is non-negligible as all ϵ_i are chosen to be some non-negligible function. Thus, $\mathcal{B}$ on input $x \in L$, accepts with non-negligible probability.

For $x \notin L$, we argue that the simulator will almost never output an accepting transcript, thus $\mathcal{B}$ on $x \notin L$ never outputs 1. By a delicate argument[5], one can show that except with small probability, Sim can never make two queries $\boldsymbol{m} = (m_1, \cdots, m_{i-1}, m_i)$ and $\boldsymbol{m}' = (m_1, \cdots, m_{i-1}, m_i')$ whose $m_i \neq m_i'$ and V^* does not abort on both $\boldsymbol{m}$ and $\boldsymbol{m}'$. In other words, during the whole execution of Sim, it never gets to see two different continuations of the same transcript. Therefore, the execution of Sim with oracle access to V^* can be roughly simulated by an algorithm with only interaction to V^*. Further notice that the interaction with V^* is simply an interaction with V plus random aborting, the execution of Sim can be therefore simulated by an algorithm (cheating prover) P^* with interaction with V (instead of V^*). If the simulator outputs an accepting transcript, then the interaction between P^* and V also outputs an accepting transcript. Since the statement x is not in L, by the soundness of the proof system Π, any (efficient) prover P^* can not convince V. We then conclude that $\mathcal{B}$ on $x \notin L$ never accepts.

We notice that the first half of the proof (for $x \in L$) relies only on the zero-knowledge property of Π. This part can be generalized to the quantum setting. The real barrier of quantizing the proof is from the second part (for $x \notin L$). Recall that in the second part of the proof, we need to argue that Sim never sees two different continuations of the same transcript. However, for a quantum simulator, even a single quantum query would completely reveal answers of possibly all transcripts. We resolve the problem in the next section.

Impossibility for Strictly Polynomial-Time Simulator. We first extend the classical result to the quantum setting, showing that all languages that have constant round post-quantum zero-knowledge proofs or arguments with strict polynomial-time BB simulators must be trivial, i.e. in **BQP**.

Let Sim be the quantum strict polynomial-time BB simulator. Roughly speaking[6], the simulator Sim uses a (dishonest) $V^*(x, \mathsf{aux}, \cdot)$ as a quantum black-box routine. It will be more clear when we define V^* below. Similar to the classical proof, we construct a random aborting verifier V^* based on the honest V and we show that there is an efficient quantum algorithm that makes use of the simulator and the random aborting verifier and decides language L. As mentioned in the previous section, although the idea follows from [BL02], we show barriers for lifting the proof and how we overcome them.

Random Aborting Verifier. A random aborting verifier V^* is similar to that defined in the classical proof, except the aborting probability $\epsilon_1 = \cdots = \epsilon_k = \epsilon$ are the same. Let H be a random oracle that independently on each input transcript $\boldsymbol{m}$ of length $i \leq k$, outputs 0 (aborting) with probability $1 - \epsilon$ and 1 (non-aborting) with some non-negligible probability ϵ. Thus, a quantum query made by the simulator Sim will become

$$\begin{aligned} |\boldsymbol{m}, 0\rangle &\to |\boldsymbol{m}, V(x, r, \boldsymbol{m})\rangle && H(\boldsymbol{m}) = 1; \\ |\boldsymbol{m}, 0\rangle &\to |\boldsymbol{m}, 0\rangle && \text{otherwise.} \end{aligned}$$

[5] By choosing each ϵ_i properly. This is the place where the proof requires the running time of the simulator is strict polynomial, instead of expected polynomial. Otherwise, such ϵ_i may not exist.

[6] To formally define it, we follow the definition in [Unr12], see Section 2.1.

In the above notation, $V(x, r, \cdot)$ is the next-message function of V corresponding to the statement x and its random tape r. Importantly, the quantum oracle access to $V^*(x, r, \cdot)$ can be simulated by constant number of quantum oracle access to $V(x, r, \cdot)$ and H. Later in the construction of our algorithm, Sim chooses a random tape for V, we assume Sim has oracle access to H and can compute $V(x, r, \cdot)$ by itself.

First Attempt. A natural approach is to consider the following algorithm $\mathcal{B}$: on input x, it samples a random tape r, a random oracle H and runs the simulator Sim on input x with oracle access to H, outputs 1 if the transcript produced by Sim is an accepting transcript and compatible with H. Here "compatible" means that conditioned on this transcript, it never aborts.

For $x \in L$, Sim would output an accepting transcript with probability roughly ϵ^k, which is the same probability as that the random aborting verifier accepts a proof. Because ϵ will be chosen as an inverse polynomial and k is a constant, $\mathcal{B}$ on $x \in L$ accepts with non-negligible probability.

Although this algorithm works on input $x \in L$, there is still an issue. As briefly mentioned at the end of the last section, the idea underlines the classical proof is: a strict polynomial-time BB simulator will not have "enough time" to obtain two non-abort responses from the verifier; thus one would use any execution of Sim that outputs 1 to convince an honest verifier V. Such a claim is not trivial in the quantum setting as a single quantum query to V^*, even if V^* aborts with very high but still non-negligible probability, reveals exponentially many non-abort responses. Thus, we can not conclude that the algorithm does not accept $x \notin L$.

Measure-and-Reprogram. A naive solution would be to measure all quantum queries made by the simulator. As long as all queries become classical, ideally we can resolve the issue. Though, this approach works for $x \notin L$, the modified algorithm may never accept $x \in L$, as measuring all the quantum queries can be easily identified. Our idea is to apply a refined way of measuring and extracting quantum queries – the "measure-and-reprogram" technique that was first introduced for proving the post-quantum security of Fiat-Shamir [DFMS19, DFM20]. Very informally, by applying the technique, we obtain the following oracle algorithm $\widetilde{\mathsf{Sim}}^H$:

- It picks k queries out of all q queries made by Sim, which will be measured later and runs Sim as a subroutine.
- Every time Sim makes a query that is supposed to be measured, $\widetilde{\mathsf{Sim}}$ measures the query and reprograms the oracle H on the measured point in a "certain" way.

Intuitively, this allows us to exactly measure the transcript that will be outputted by Sim at the end of the execution while still preserving its succeeding probability. Followed by the "measure-and-reprogram" lemma[7], by doing the above changes, the probability that the output transcript of $\widetilde{\mathsf{Sim}}$ gets measured during "measure-and-reprogram" and it is an accepting transcript is non-negligible.

We can then define a new algorithm $\mathcal{B}'$ based on $\widetilde{\mathsf{Sim}}$: on input x, it samples a random tape r, a random oracle H and runs the algorithm $\widetilde{\mathsf{Sim}}$ on input x with oracle access to H, outputs 1 if the output transcript is an accepting transcript and compatible with the updated H. Here H gets updated in the measure-and-reprogram process. This algorithm only partially solves the issue. Recall our goal is to show the algorithm can be turn into a cheating prover, thus all its queries can be simulated by only having interaction with a honest verifier V. It now makes k out of q queries classical, and these k queries are exactly what will be the output transcript. These k queries can then be simulated by the interaction with V. However, for the other $q - k$ queries, they are still quantum queries and may be hard to answer if it only sees an interaction with V.

[7] We rely on a variant by [YZ20], also see [DFMS19, DFM20].

Finish the Proof. Actually, it turns out that the other $q-k$ queries can be easily answered, by preparing an empty oracle H_0 (which outputs 0 on every input) instead of a real random oracle H. Thus, the algorithm for deciding L is the following:

$\widetilde{\mathcal{B}}$: On input x, it samples a random tape r, an empty oracle H_0 and runs the algorithm $\widetilde{\mathsf{Sim}}$ on input x with oracle access to H_0, outputs 1 if the output transcript is an accepting transcript and compatible with the updated H_0.

The only difference between $\widetilde{\mathcal{B}}$ and $\mathcal{B}'$ is that the underlying $\widetilde{\mathsf{Sim}}$ gets either an empty oracle H_0 or a sparse oracle H (each output is 1 with probability ϵ). By [HRS15, Lemma 3], the advantage of distinguish an empty or a sparse oracle by a q-quantum-query algorithm is at most $8q^2\epsilon$. Therefore, for any $x \in L$, $\widetilde{\mathcal{B}}$ outputs 1 with probability at least that of $\mathcal{B}'$ outputs 1 minus $8q^2\epsilon$. By carefully tuning ϵ, we can show that $\widetilde{\mathcal{B}}$ still accepts $x \in L$ with non-negligible probability.

For $x \notin L$, we want to turn $\widetilde{\mathcal{B}}$ into a cheating prover P^*. Because P^* can never convince V on $x \notin L$, $\widetilde{\mathcal{B}}$ should never accept x unless with negligible probability. Assuming the first quantum query made by $\widetilde{\mathsf{Sim}}$ is not going to be measured. In this case, H_0 does not get updated and $\widetilde{\mathsf{Sim}}$ makes the first quantum query to $V^*(x, r, H_0, \cdot)$. By the definition of the random aborting verifier, it always aborts on any input. Therefore, it can answer the first quantum query by simply always returning 0, without getting any response from the real verifier $V(x, r, \cdot)$. If the first quantum query needs to be measured, it will be part of the final output transcript. P^* can obtain and record the response by doing the interaction with $V(x, r, \cdot)$. Similarly, whenever $\widetilde{\mathsf{Sim}}$ makes a quantum query to $V^*(x, r, H_0, \cdot)$ (where H_0 is the updated oracle so far), the only non-abort responses come from the input $\boldsymbol{m}$ such that all its prefix $\boldsymbol{m}_j$ satisfying $H_0(\boldsymbol{m}_j) = 1$. Because H_0 is initialized as an empty oracle, every input $\boldsymbol{m}$ satisfying $H_0(\boldsymbol{m}) = 1$ must be measured and reprogrammed at certain point in the execution of $\widetilde{\mathsf{Sim}}$, its response $V(x, r, \boldsymbol{m})$ is already known and recorded. For $\boldsymbol{m}$ such that $H(\boldsymbol{m}) = 0$, we do not need to know its response. Overall, P^* can simulate any quantum query in the execution of $\widetilde{\mathsf{Sim}}^{H_0}$. Therefore, $\widetilde{\mathcal{B}}$ never accepts $x \notin L$ except with negligible probability.

Thus, post-quantum constant-round zero-knowledge proofs or arguments with strict polynomial-time BB simulators for all languages in **NP** do not exist unless **NP** $\subseteq$ **BQP**.

Impossibility for Expected Polynomial-Time Simulator. As discussed by Barak and Lindell [BL02], a natural attempt to extend the classical impossibility proof to expected-time BB simulators is by truncating the execution of a simulator (see Section 1.3 of [BL02]); they pointed out that such an attempt would fail. Imagine a expected polynomial-time BB simulator (with expected running time $q/2$) has oracle access to an aborting verifier V^* with a very small non-aborting probability, say $\epsilon = q^{-10}$. As long as we truncate the execution of the simulator when the running time is significantly smaller than q^{10}, it would never get any non-abort response from the aborting verifier and is not able to produce any accepting transcript.

Another way to interpret the above argument is: By the impossibility of strict-poly BB simulation, we can say that a simulator has to "learn" the aborting probability. If a BB simulator is only allowed to make a bounded number of queries (which is independent of the aborting probability ϵ), it almost can never learn ϵ, as long as $1/\epsilon$ is significantly larger.

We show that, informally, if there is an expected polynomial-time BB simulator, then by truncating this simulator, it is still a "good-enough" simulator for a specific aborting verifier while it does not measure/learn the aborting probability. We know that this can not happen for all languages in **NP** unless **NP** $\subseteq$ **BQP**.

A crucial difference between quantum and classical malicious verifiers is that the quantum malicious verifier can use an auxiliary input qubit to control the aborting probability in "superposition". This implies that this auxiliary qubit can somehow "entangle with the runtime" of the protocol or the simulation. Therefore, conditioned on accepting, the state of this auxiliary qubit after the interaction between the real prover and the verifier can be far from the state after the simulation since the black-box simulator requires to "measure" the aborting probability (and thus measures the control bit). By combining this observation and our impossibility result on the strict polynomial-time simulation, we can also fail the expected polynomial-time simulation. To be more specific, consider the following verifier $\widetilde{V}^*$ that runs a honest verifier V and a random aborting verifier V^* (with non-aborting probability ϵ) in superposition:

- It prepares a control bit $|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ at the beginning.
- It runs V and V^* in superposition: on input $\boldsymbol{m}$, if the control bit is 0, it never aborts and behaves as V; otherwise the control bit is 1, it aborts with probability $1-\epsilon$ as V^*, by querying an internal random oracle.
- Finally, it outputs a classical bit b indicating whether it accepts and a single qubit in the control bit register.

If the control bit is 0, we know that V always accepts by perfect completeness of Π. If the control bit is 1, it accepts with ϵ^k. Thus, if $\widetilde{V}^*$ accepts (the classical output $b = 1$), the output qubit is proportional to $|0\rangle + \sqrt{\epsilon^k}\,|1\rangle$[8].

Let Sim be the expected polynomial-time BB simulator that makes $q/2$ queries in expectation. Consider a truncated simulator $\mathsf{Sim}_{\mathsf{trunc}}$ for $\widetilde{V}^*$ which halts after Sim tries to make the $(q+1)$-th query. By Markov inequality, we know the probability that Sim will halt within the first q queries is at least $1/2$. When Sim makes at most q queries, it outputs $b = 1$ with probability at least $1/2$ (because when the control bit is 0, V always accepts). It is worth noting that this two events are independent. Thus, $\mathsf{Sim}_{\mathsf{trunc}}$ would output $b = 1$ with probability at least $1/4$. As we know when $b = 1$, $\widetilde{V}^*$ always outputs the qubit $|0\rangle + \sqrt{\epsilon^k}\,|1\rangle$. By zero-knowledge property, $\mathsf{Sim}_{\mathsf{trunc}}$ should also output a state close to $|0\rangle + \sqrt{\epsilon^k}\,|1\rangle$ when the classical output $b = 1$. However, this can not happen for languages outside $\mathbf{BQP}$. By our impossibility result for strict poly-time simulators, such a bounded query BB simulator would essentially need to measure the aborting probability of $\widetilde{V}^*$, which necessarily collapse the control qubit (as the aborting probability for control bit 0 is 0, and for control bit 1 is ϵ). Therefore, we conclude that post-quantum constant-round zero-knowledge proofs or arguments with (expected) polynomial-time BB simulators for all languages in $\mathbf{NP}$ do not exist unless $\mathbf{NP} \subseteq \mathbf{BQP}$.

On the Efficiency of Malicious Verifier. In the explanation so far, we considered a malicious verifier that relies on a random oracle. For making the verifier efficient, a standard technique is to simulate a random oracle by using a $2q$-wise independent function when the number of queries is at most q [Zha12b]. Though we show that this works in our setting, it is not as trivial as one would expect due to some technical reasons.[9] Therefore, in the main body, we first consider an inefficient malicious verifier that simulates the random oracle by a completely random function, and then we explain how we make the verifier be efficient without affecting the proof.

[8] In the real execution, $\widetilde{V}^*$ would be entangled with its internal random oracle H and make the final qubit a mixed state. Nonetheless, we show such entanglement can be uncomputed by $\widetilde{V}^*$ and the final qubit is a pure state. For simplicity, we omit the details here.

[9] The reason is related to that we have to uncompute the entanglement between H and $\widetilde{V}^*$'s final qubit as explained in Footnote 8

1.2.2 Impossibility of Constant-Round Public-Coin or Three-Round ϵ-ZK

In the classical setting, Goldreich and Krawczyk [GK96b] proved the impossibility of constant-round public-coin or three-round ZK arguments. It is easy to see that their result also rules out ϵ-ZK arguments by essentially the same proof. Roughly speaking, we translate their proof into the quantum setting by again relying on the measure-and-reprogram technique [DFMS19, DFM20]. We give more details of each case below.

Constant-Round Public-Coin Case. For a constant-round public-coin protocol $\Pi = (P, V)$ for an **NP** language L, we consider a malicious verifier V^* that derives its messages by applying a random oracle on the current transcript. From the view of the honest prover, V^* is perfectly indistinguishable from the honest verifier V. Thus, if V^* interacts with the honest prover given on common input $x \in L$ and prover's private input $w \in R_L(x)$, it always accepts by the completeness of the protocol. Let Sim be a simulator for the ϵ-ZK property. By the above observation, when Sim is given oracle access to V^* on input $x \in L$, it should let V^* accept with probability at least 0.9 since otherwise V^* may notice the difference with a constant advantage, which violates the ϵ-ZK property.[10] On the other hand, we observe that an accepting transcript between V^* is essentially an accepting proof for the non-interactive argument Π_{ni} obtained by applying Fiat-Shamir transform to the protocol Π since the way of deriving the verifier's messages is the same as that in the Fiat-Shamir transform. Noting that V^* can be simulated given oracle access to the random oracle, if Sim lets V^* accept on some $x \notin L$ with non-negligible probability, such a simulator can be directly translated into an adversary that breaks the soundness of Π_{ni} in the quantum random oracle model. On the other hand, it is shown in [DFM20] that Fiat-Shamir transform preserves soundness up to polynomial security loss for constant-round public-coin protocols, and thus Π_{ni} has negligible soundness error. This means that Sim lets V^* accept with negligible probability for any $x \notin L$. Combining the above, we can decide if $x \in L$ by simulating an interaction between Sim and V^* and then seeing if V^* accepts finally. This means $L \in \mathbf{BQP}$. Therefore, such a protocol for all **NP** does not exist unless $\mathbf{NP} \subseteq \mathbf{BQP}$.

Three-Round Case. This case is similar to the constant-round public-coin case except that we apply a random oracle to obtain verifier's private randomness rather than a verifier's message itself. Due to this difference, we cannot directly relate the $x \notin L$ case to the soundness of Fiat-Shamir, and we need more careful analysis.

For a three-round protocol $\Pi = (P, V)$ for an **NP** language L, we consider a malicious verifier V^* that derives its *private randomness* by applying a random oracle on *prover's first message*. From the view of the honest prover, V^* is perfectly indistinguishable from the honest verifier V. Thus, if V^* interacts with the honest prover given on common input $x \in L$ and prover's private input $w \in R_L(x)$, it always accepts by the completeness of the protocol. Let Sim be a simulator for the ϵ-ZK property. By the above observation, when Sim is given oracle access to V^* on input $x \in L$, it should let V^* accept with probability at least 0.9 similarly to the constant-round public-coin case. However, unlike the constant-round public-coin case above, we cannot directly say that Sim let V^* accept with negligible probability on input $x \notin L$ because V^* derives the private randomness by the random oracle, which is different from the Fiat-Shamir transform. Therefore we need additional ideas.

Sim can be seen as an algorithm that makes quantum queries to the next-message-generation function F_{next}, which outputs V^*'s second-round message taking prover's first-round message on

[10] The choice of the constant is arbitrary.

input, and output-decision function F_{out}, which decides if V^* accepts taking a transcript as input. (Note these functions depend on the random oracle.) Finally, it outputs an accepting transcript with probability at least 0.9. First, we claim that we can assume that Sim only has the oracle F_{next} and does not make any query to F_{out} if we admit a polynomial security loss. Intuitively, this is because if it makes a query on which F_{out} returns "accept", then it could have used this query as its final output. Though this is trivial in the classical setting, it is not in the quantum setting. Fortunately, we can prove this by relying on the one-way to hiding lemma [Unr15, AHU19] in the quantum setting as well. Thus, we think of Sim as an algorithm that makes quantum queries to F_{next} and outputs an accepting transcript with probability at least $\frac{1}{\mathsf{poly}(\lambda)}$ when $x \in L$.

Next, we apply the measure-and-reprogram lemma of [DFM20] to $\mathsf{Sim}^{F_{\mathsf{next}}}$. That is, we consider an experiment $\mathsf{Exp}_{\mathsf{MaR}}(x)$ which roughly works as follows: The experiment simulates $\mathsf{Sim}^{F_{\mathsf{next}}}(x)$ except that a randomly chosen Sim's query is measured (let m_P be the outcome), its response is replaced with a freshly sampled message m_V independently of F_{next}, and the oracle is updated to be consistent to this response thereafter. Finally, the experiment outputs the transcript output by Sim.

By using the measure-and-reprogram lemma, the probability (which we denote by $p(x)$ in the following) that $\mathsf{Exp}_{\mathsf{MaR}}(x)$ outputs an accepting transcript whose first and second messages match m_P and m_V is $\frac{1}{\mathsf{poly}(\lambda)}$ times the probability that $\mathsf{Sim}^{F_{\mathsf{next}}}(x)$ outputs an accepting transcript. Thus, $p(x)$ is at least $\frac{1}{\mathsf{poly}(\lambda)}$ for all $x \in L$.

On the other hand, we can prove that $p(x)$ is negligible for all $x \notin L$ by using soundness of the protocol Π. Indeed, we can construct a cheating prover P^* that simulates $\mathcal{S}^{F_{\mathsf{next}}}(x)$ where F_{next} is simulated according to a random oracle chosen by P^*, sends m_P to the external verifier as the first message, embeds verifier's response as m_V, and sends the third message derived from the output of $\mathcal{S}^{F_{\mathsf{next}}}(x)$ to the external verifier. It is easy to see that P^* perfectly simulates the environment of $\mathsf{Exp}_{\mathsf{MaR}}(x)$ for $\mathcal{S}^{F_{\mathsf{next}}}(x)$ and thus the probability that the verifier accepts is at least $p(x)$. Therefore, by the assumed soundness, $p(x)$ is negligible.

By combining above, we can decide if $x \in L$ by simulating $\mathsf{Exp}_{\mathsf{MaR}}(x)$ and then seeing if the output is an accepting transcript whose first and second messages match m_P and m_V. (Note that we can efficiently check if the transcript is accepting if we sample m_V by ourselves so that we know the corresponding verifier's private randomness). This means $L \in \mathbf{BQP}$. Therefore, such a protocol for all $\mathbf{NP}$ does not exist unless $\mathbf{NP} \subseteq \mathbf{BQP}$.

1.3 More Related Work

Jain et al. [JKMR09] showed that there does not exist constant-round public-coin or three-round post-quantum BBZK *proofs* for $\mathbf{NP}$ unless $\mathbf{BQP} \subseteq \mathbf{NP}$. Indeed, they showed that this holds even if the last message in the protocol can be quantum. We believe that our impossibility results can also be extended to this setting, but we focused on classical protocols in the context of post-quantum security for simplicity. If we focus on classical protocols, our impossibility results are stronger than theirs as we also rule out BB ϵ-*ZK arguments*. To the best of our knowledge, the work of [JKMR09] is the only known result on the impossibility of quantum BBZK.

We review additional related works on lower bounds of ZK protocols in the classical setting. Katz [Kat08] proved that there does not exist four-round BBZK *proofs* for $\mathbf{NP}$ unless $\mathbf{NP} \subseteq \mathbf{coMA}$. It is interesting to study if we can extend this to rule out four-round post-quantum BB ϵ-ZK proofs for $\mathbf{NP}$ under a reasonable complexity assumption. We note that there exists four-round (classical) BBZK *arguments* for $\mathbf{NP}$ under the existence of one-way functions [BJY97]. It is also interesting to study if we can extend their construction to construct four-round post-quantum BB ϵ-ZK for

NP. (Note that it is necessary to relax ZK property by Theorem 1.1.)

Kalai, Rothblum, and Rothblum [KRR17] proved that there does not exist constant-round public-coin ZK *proofs* for **NP** even with *non-uniform simulation* under certain assumptions on obfuscation. Fleischhacker, Goyal, and Jain [FGJ18] proved that there does not exist three-round ZK *proofs* for **NP** even with *non-uniform simulation* under the same assumptions. Though these results are shown in the classical setting, it might be possible to extend them to the quantum setting by assuming similar assumptions against quantum adversaries. However, that would result in impossibility for *proofs* whereas our impossibility covers *arguments* though limited to BB simulation.

2 Preliminaries

Basic Notations. We denote by λ the security parameter throughout the paper. For a positive integer $n \in \mathbb{N}$, $[n]$ denotes a set $\{1, 2, ..., n\}$. For a finite set $\mathcal{X}$, $x \xleftarrow{\$} \mathcal{X}$ means that x is uniformly chosen from $\mathcal{X}$. For a finite set $\mathcal{X}$ and a positive integer k, $\mathcal{X}^{\leq k}$ is defined to be $\bigcup_{i\in[k]} \mathcal{X}^i$. For finite sets $\mathcal{X}$ and $\mathcal{Y}$, $\mathsf{Func}(\mathcal{X}, \mathcal{Y})$ denotes the set of all functions with domain $\mathcal{X}$ and range $\mathcal{Y}$.

A function $f : \mathbb{N} \to [0, 1]$ is said to be *negligible* if for all polynomial p and sufficiently large $\lambda \in \mathbb{N}$, we have $f(\lambda) < 1/p(\lambda)$; it is said to be *overwhelming* if $1 - f$ is negligible, and said to be *noticeable* if there is a polynomial p such that $f(\lambda) \geq 1/p(\lambda)$ for sufficiently large $\lambda \in \mathbb{N}$. We denote by poly an unspecified polynomial and by negl an unspecified negligible function.

We use PPT and QPT to mean (classical) probabilistic polynomial time and quantum polynomial time, respectively. For a classical probabilistic or quantum algorithm $\mathcal{A}$, $y \xleftarrow{\$} \mathcal{A}(x)$ means that $\mathcal{A}$ is run on input x and outputs y. When $\mathcal{A}$ is a classical probabilistic algorithm, we denote by $\mathcal{A}(x; r)$ the execution of $\mathcal{A}$ on input x and randomness r. When $\mathcal{A}$ is a quantum algorithm that takes a quantum advice, we denote by $\mathcal{A}(x; \rho)$ the execution of $\mathcal{A}$ on input x and an advice ρ.

We use the bold font (like $\mathbf{X}$) to denote quantum registers, and $\mathcal{H}_{\mathbf{X}}$ to mean the Hilbert space corresponding to the register $\mathbf{X}$. For a quantum state ρ, $M_{\mathbf{X}} \circ \rho$ means a measurement in the computational basis on the register $\mathbf{X}$ of ρ. For quantum states ρ and ρ', $\mathsf{TD}(\rho, \rho')$ denotes trace distance between them.

Standard Computational Models.

- A PPT algorithm is a probabilistic polynomial time (classical) Turing machine. A PPT algorithm is also often seen as a sequence of uniform polynomial-size circuits.
- A QPT algorithm is a polynomial time quantum Turing machine. A QPT algorithm is also often seen as a sequence of uniform polynomial-size quantum circuits.
- An adversary (or malicious party) is modeled as a non-uniform QPT algorithm $\mathcal{A}$ (with quantum advice) that is specified by sequences of polynomial-size quantum circuits $\{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$ and polynomial-size quantum advice $\{\rho_\lambda\}_{\lambda\in\mathbb{N}}$. When $\mathcal{A}$ takes an input of λ-bit, $\mathcal{A}$ runs $\mathcal{A}_\lambda$ taking ρ_λ as an advice.

Indistinguishability of Quantum States. We define computational and statistical indistinguishability of quantum states similarly to [BS20].

We may consider random variables over bit strings or over quantum states. This will be clear from the context. For ensembles of random variables $\mathcal{X} = \{X_i\}_{\lambda\in\mathbb{N},i\in I_\lambda}$ and $\mathcal{Y} = \{Y_i\}_{\lambda\in\mathbb{N},i\in I_\lambda}$ over the same set of indices $I = \bigcup_{\lambda\in\mathbb{N}} I_\lambda$ and a function δ, we write $\mathcal{X} \overset{comp}{\approx}_\delta \mathcal{Y}$ to mean that for any

non-uniform QPT algorithm $\mathcal{A} = \{\mathcal{A}_\lambda, \rho_\lambda\}$, there exists a negligible function negl such that for all $\lambda \in \mathbb{N}$, $i \in I_\lambda$, we have

$$|\Pr[\mathcal{A}_\lambda(X_i; \rho_\lambda)] - \Pr[\mathcal{A}_\lambda(Y_i; \rho_\lambda)]| \leq \delta(\lambda) + \mathsf{negl}(\lambda).$$

Especially, when we have the above for $\delta = 0$, we say that $\mathcal{X}$ and $\mathcal{Y}$ are computationally indistinguishable, and simply write $\mathcal{X} \overset{comp}{\approx} \mathcal{Y}$.

Similarly, we write $\mathcal{X} \overset{stat}{\approx}_\delta \mathcal{Y}$ to mean that for any unbounded time algorithm $\mathcal{A}$, there exists a negligible function negl such that for all $\lambda \in \mathbb{N}$, $i \in I_\lambda$, we have

$$|\Pr[\mathcal{A}(X_i)] - \Pr[\mathcal{A}(Y_i)]| \leq \delta(\lambda) + \mathsf{negl}(\lambda).^{11}$$

Especially, when we have the above for $\delta = 0$, we say that $\mathcal{X}$ and $\mathcal{Y}$ are statistically indistinguishable, and simply write $\mathcal{X} \overset{stat}{\approx} \mathcal{Y}$. Moreover, we write $\mathcal{X} \equiv \mathcal{Y}$ to mean that X_i and Y_i are distributed identically for all $i \in I$

2.1 Interactive Proof and Argument.

We define interactive proofs and arguments similarly to [BS20, CCY20b].

Notations. For an **NP** language L and $x \in L$, $R_L(x)$ is the set that consists of all (classical) witnesses w such that the verification machine for L accepts (x, w).

A classical interactive protocol is modeled as an interaction between interactive classical polynomial-time machines P referred to as a prover and V referred to as a verifier. We denote by $\langle P(x_P), V(x_V)\rangle(x)$ an execution of the protocol where x is a common input, x_P is P's private input, and x_V is V's private input. We denote by $\mathsf{OUT}_V\langle P(x_P), V(x_V)\rangle(x)$ the final output of V in the execution. An honest verifier's output is $\top$ indicating acceptance or $\bot$ indicating rejection. We say that the protocol is public-coin if the honest verifier V does not use any private randomness, i.e., each message sent from V is a uniform string of a certain length and V's final output is derived by applying an efficiently computable classical function on the transcript.

Definition 2.1 (Interactive Proof and Argument for **NP**). *A classical interactive proof or argument Π for an **NP** language L is an interactive protocol between a PPT prover P and a PPT verifier V that satisfies the following:*

Perfect Completeness. *For any $x \in L$, and $w \in R_L(x)$, we have*

$$\Pr[\mathsf{OUT}_V\langle P(w), V\rangle(x) = \top] = 1$$

Statistical/Computational Soundness. *We say that an interactive protocol is statistically (resp. computationally) sound if for any unbounded-time (resp. non-uniform QPT) cheating prover P^*, there exists a negligible function negl such that for any $\lambda \in \mathbb{N}$ and any $x \in \{0,1\}^\lambda \setminus L$, we have*

$$\Pr[\mathsf{OUT}_V\langle P^*, V\rangle(x) = \top] \leq \mathsf{negl}(\lambda).$$

We call an interactive protocol with statistical (resp. computational) soundness an interactive proof (resp. argument).

[11] In other words, $\mathcal{X} \overset{stat}{\approx}_\delta \mathcal{Y}$ means that there exists a negligible function negl such that the trace distance between ρ_{X_i} and ρ_{Y_i} is at most $\delta(\lambda)+\mathsf{negl}(\lambda)$ for all $\lambda \in \mathbb{N}$ and $i \in I_\lambda$ where ρ_{X_i} and ρ_{Y_i} denote density matrices corresponding to X_i and Y_i.

Malicious verifier and black-box simulator. For a formal definition of black-box quantum zero-knowledge, we give a model of quantum malicious verifiers against classical interactive protocols. A malicious verifier V^* is specified by a sequence of unitary U_λ^* over the internal register $\mathbf{V}_\lambda$ and the message register $\mathbf{M}_\lambda$ (whose details are explained later) and an auxiliary input ρ_λ indexed by the security parameter $\lambda \in \mathbb{N}$. We say that V^* is non-uniform QPT if the sizes of U_λ^* and ρ_λ are polynomial in λ. In the rest of this paper, λ is always set to be the length of the statement x to be proven, and thus we omit λ for notational simplicity.

Its internal register $\mathbf{V}$ consists of the statement register $\mathbf{X}$, auxiliary input register $\mathbf{Aux}$, and verifier's working register $\mathbf{W}$, and part of $\mathbf{V}$ is designated as the output register $\mathbf{Out}$. V^* interacts with an honest prover P of a protocol Π on a common input x and P's private input $w \in R_L(x)$ in the following manner:

1. $\mathbf{X}$ is initialized to x, $\mathbf{Aux}$ is initialized to ρ, and $\mathbf{W}$ and $\mathbf{M}$ are initialized to be $|0\rangle$.
2. P (with private input w) and V^* run the protocol Π as follows:
   (a) On V^*'s turn, it applies the unitary U^*, measures $\mathbf{M}$, and sends the measurement outcome to P.
   (b) On P's turn, when it sends a message to the verifier, $\mathbf{M}$ is overwritten by the message. Note that this can be done since $\mathbf{M}$ is measured in the previous V^*'s turn.
3. After P sends the final message of Π, V^* applies U^* and outputs the state in $\mathbf{Out}$, tracing out all other registers.

We denote by $\langle P(w), V^*(\rho)\rangle(x)$ the above execution and by $\mathsf{OUT}_{V^*}\langle P(w), V^*(\rho)\rangle(x)$ the final output of V^*, which is a quantum state over $\mathbf{Out}$.

A quantum black-box simulator Sim is modeled as a quantum oracle Turing machine (e.g., see [BBBV97]). We say that Sim is expected-QPT (resp. strict-QPT) if the expected (resp. maximum) number of steps is polynomial in the input length counting an oracle access as a unit step. For an input x and a malicious verifier V^* specified by a unitary U^* and auxiliary input ρ, Sim works over the input register $\mathbf{Inp}$, verifier's internal register $\mathbf{V}$, message register $\mathbf{M}$, and its working register $\mathbf{S}$ as follows. $\mathbf{Inp}$ and the sub-register $\mathbf{X}$ of $\mathbf{V}$ are initialized to x, the sub-register $\mathbf{Aux}$ of $\mathbf{V}$ is initialized to ρ, and all other registers ($\mathbf{W}$, $\mathbf{M}$, and $\mathbf{S}$) are initialized to $|0\rangle$. Sim is given oracle access to U^* and its inverse $U^{*\dagger}$ and can apply any unitary over $\mathbf{Inp}$, $\mathbf{M}$, and $\mathbf{S}$, but it is not allowed to directly act on $\mathbf{V}$ (except for the invocations of U^* or $U^{*\dagger}$). We denote by $\mathsf{Sim}^{V^*(x;\rho)}(x)$ the above execution and by $\mathsf{OUT}_{V^*}(\mathsf{Sim}^{V^*(x;\rho)}(x))$ the output of V^*, i.e., final state in $\mathbf{Out}$ tracing out all other registers after the execution.

Based on the above formalization, we define post-quantum black-box zero-knowledge proof/argument as follows.

Definition 2.2 (Post-Quantum Black-Box Zero-Knowledge Proof and Argument)**.** *A post-quantum black-box zero-knowledge proof (resp. argument) for an* **NP** *language L is a classical interactive proof (resp. argument) for L that satisfies the following property in addition to perfect completeness and statistical (resp. computational) soundness:*

Quantum Black-Box Zero-Knowledge. *There exists an expected-QPT simulator* Sim *such that for any non-uniform QPT malicious verifier V^* with an auxiliary input ρ, we have*

$$\{\mathsf{OUT}_{V^*}\langle P(w), V^*(\rho)\rangle(x)\}_{\lambda,x,w} \overset{comp}{\approx} \{\mathsf{OUT}_{V^*}(\mathsf{Sim}^{V^*(x;\rho)}(x))\}_{\lambda,x,w}$$

where $\lambda \in \mathbb{N}$, $x \in L \cap \{0,1\}^\lambda$, and $w \in R_L(\lambda)$.

Quantum Black-Box Zero-Knowledge for Inefficient Verifiers. In the above definition, we restrict a malicious verifier V^* to be non-uniform QPT. On the other hand, We say that Sim works for inefficient verifiers if the above holds even for all possibly inefficient malicious verifiers V^*. To the best of our knowledge, all known black-box simulation techniques (in classical or quantum settings) work even for inefficient verifiers. The reason of defining this notion is that we first prove the impossibility of quantum black-box simulation for inefficient verifiers (Theorem 3.7) as it is simpler than that for efficient verifiers (Theorem 3.1). (Remark that the impossibility of quantum black-box simulation for inefficient verifiers is weaker than the impossibility of quantum black-box simulation for efficient verifiers.) We stress that we finally extend it to prove the impossibility of quantum black-box simulation for efficient verifiers (Theorem 3.1).

We next define a weaker version of zero-knowledge called *ϵ-zero-knowledge* following [CCY20a].

Definition 2.3 (Post-Quantum Black-Box ϵ-Zero-Knowledge Proof and Argument)**.** *A post-quantum black-box ϵ-zero-knowledge proof (resp. argument) for an* **NP** *language L is a classical interactive proof (resp. argument) for L that satisfies the following property in addition to perfect completeness and statistical (resp. computational) soundness:*

Quantum Black-Box ϵ-Zero-Knowledge. *For any noticeable ϵ, there exists a strict-QPT simulator Sim such that for any non-uniform QPT malicious verifier V^* with an auxiliary input ρ, we have*

$$\{\mathsf{OUT}_{V^*}\langle P(w), V^*(\rho)\rangle(x)\}_{\lambda,x,w} \overset{comp}{\approx}_{\epsilon} \{\mathsf{OUT}_{V^*}(\mathsf{Sim}^{V^*(x;\rho)}(x))\}_{\lambda,x,w}$$

where $\lambda \in \mathbb{N}$, $x \in L \cap \{0,1\}^\lambda$, and $w \in R_L(\lambda)$. Note that the running time of Sim may depend on $1/\epsilon$.

Remark 1. *In the definition of quantum black-box ϵ-zero-knowledge, we assume that Sim runs in strict-QPT rather than expected-QPT without loss of generality. Indeed, if we have a expected-QPT simulator, then we can consider a truncated version of it that immediately halts if its running time exceeds the expected running time too much. It is easy to see that this truncated version of the simulator is still good enough for ϵ-zero-knowledge (unlike for the full-fledged zero-knowledge). A similar observation is also given in [BL02].*

2.2 Useful Lemmas

The following lemma is heavily used throughout the paper.

Lemma 2.4 ([Zha12b])**.** *For any sets $\mathcal{X}$ and $\mathcal{Y}$ of classical strings and q-quantum-query algorithm $\mathcal{A}$, we have*

$$\Pr[\mathcal{A}^H = 1 : H \xleftarrow{\$} \mathsf{Func}(\mathcal{X},\mathcal{Y})] = \Pr[\mathcal{A}^H = 1 : H \xleftarrow{\$} \mathcal{H}_{2q}]$$

where $\mathcal{H}_{2q}$ is a family of $2q$-wise independent hash functions from $\mathcal{X}$ to $\mathcal{Y}$.

The following two lemmas are used in Section 3.

Lemma 2.5 ([HRS15, Lemma 3])**.** *Let $\mathcal{X}$ be a finite set, $\epsilon \in [0,1]$ be a non-negative real number, and $\mathcal{H}_\epsilon$ be a distribution over $H : \mathcal{X} \to \{0,1\}$ such that we have $\Pr[H(x) = 1] = \epsilon$ independently for each $x \in \mathcal{X}$. Let $H_0 : \mathcal{X} \to \{0,1\}$ be the function that returns 0 for all inputs $x \in \mathcal{X}$. Then for any algorithm $\mathcal{A}$ that makes at most q quantum queries, we have*

$$\left|\Pr\left[\mathcal{A}^H = 1 : H \xleftarrow{\$} \mathcal{H}_\epsilon\right] - \Pr\left[\mathcal{A}^{H_0} = 1\right]\right| \leq 8q^2\epsilon.$$

Lemma 2.6 (SWAP test)**.** *There is a QPT algorithm, called the SWAP test, which satisfies the following: the algorithm takes a product state $\rho \otimes \sigma$ as input, and accepts with probability $\frac{1+\mathrm{Tr}(\rho\sigma)}{2}$. Especially, when ρ is a pure state $|\phi\rangle\langle\phi|$, the probability is $\frac{1+\langle\phi|\sigma|\phi\rangle}{2}$.*

We will use a corollary of the one-way to hiding lemma [Unr15, AHU19] in Section 5. First, we introduce a special case of the one-way to hiding lemma in [AHU19].

Lemma 2.7 (A Special Case of One-Way to Hiding Lemma [AHU19])**.** *Let $S \subseteq \mathcal{X}$ be random. Let z be a random bit string. (S and z may have an arbitrary joint distribution.) Let $F_S : \mathcal{X} \to \{0,1\}$ be the function defined by*

$$F_S(x) := \begin{cases} 1 & \text{if } x \in S \\ 0 & \text{otherwise} \end{cases}.$$

Let $F_\emptyset : \mathcal{X} \to \{0,1\}$ be the function that outputs 0 on all inputs. Let $\mathcal{A}$ be an oracle-aided quantum algorithm that makes at most q quantum queries. Let $\mathcal{B}$ be an algorithm that on input z chooses $i \xleftarrow{\$} [q]$, runs $\mathcal{A}^{F_\emptyset}(z)$, measures $\mathcal{A}$'s i-th query, and outputs the measurement outcome. Then we have

$$\left|\Pr[\mathcal{A}^{F_S}(z) \in S] - \Pr[\mathcal{A}^{F_\emptyset}(z) \in S]\right| \leq 2\sqrt{(q+1)\Pr[\mathcal{B}(z) \in S]}.$$

We show a simple corollary of the above lemma, which roughly says that if we can find an element of S by making polynomial number of quantum queries to F_S, then we can find an element of S without making any query to F_S with a polynomial reduction loss.

Corollary 2.8. *Let S, z, F_S, $F_\emptyset$, $\mathcal{A}$, and $\mathcal{B}$ be defined as in Lemma 2.7. Let $\mathcal{C}$ be an algorithm (without any oracle) that works as follows: On input z, it flips a bit $b \xleftarrow{\$} \{0,1\}$. If $b = 0$, then it runs $\mathcal{A}^{F_\emptyset}(z)$ by simulating $F_\emptyset$ by itself and if $b = 1$, then it runs $\mathcal{B}(z)$. Then we have*

$$\sqrt{\Pr[\mathcal{C}(z) \in S]} \geq \frac{\Pr[\mathcal{A}^{F_S}(z) \in S]}{4\sqrt{q+1}}.$$

Proof. By Lemma 2.7, we have

$$\left|\Pr[\mathcal{A}^{F_S}(z) \in S] - \Pr[\mathcal{A}^{F_\emptyset}(z) \in S]\right| \leq 2\sqrt{(q+1)\Pr[\mathcal{B}(z) \in S]}.$$

which implies

$$\begin{aligned}
\Pr[\mathcal{A}^{F_S}(z) \in S] &\leq 2\sqrt{(q+1)\Pr[\mathcal{B}(z) \in S]} + \Pr[\mathcal{A}^{F_\emptyset}(z) \in S] \\
&\leq 2\sqrt{(q+1)\Pr[\mathcal{B}(z) \in S]} + 2\sqrt{(q+1)\Pr[\mathcal{A}^{F_\emptyset}(z) \in S]} \\
&\leq 2\sqrt{2(q+1)(\Pr[\mathcal{B}(z) \in S] + \Pr[\mathcal{A}^{F_\emptyset}(z) \in S])} \\
&= 2\sqrt{2(q+1)(2\Pr[\mathcal{C}(z) \in S])}.
\end{aligned}$$

Corollary 2.8 immediately follows from the above. □

2.3 Measure-and-Reprogram Lemma

We review the measure-and-reprogram lemma of [DFM20] with notations based on those in [YZ20]. We first give intuitive explanations for these notations, which are taken from [YZ20]. For a

quantumly-accessible classical oracle $\mathcal{O}$, we denote by $\mathcal{O} \leftarrow \mathsf{Reprogram}(\mathcal{O}, x, y)$ to mean that we reprogram $\mathcal{O}$ to output y on input x. For a q-quantum-query algorithm $\mathcal{A}$, function $H : \mathcal{X} \to \mathcal{Y}$, and $\mathbf{y} = (y_1, ..., y_k) \in \mathcal{Y}^k$, we denote by $\tilde{\mathcal{A}}[H, \mathbf{y}]$ to mean an algorithm that runs $\mathcal{A}$ w.r.t. an oracle that computes H except that randomly chosen k queries are measured and the oracle is reprogrammed to output y_i on i-th measured query. Formal definitions are given below:

Definition 2.9 (Reprogramming Oracle)**.** *Let $\mathcal{A}$ be a quantum algorithm with quantumly-accessible oracle $\mathcal{O}$ that is initialized to be an oracle that computes some classical function from $\mathcal{X}$ to $\mathcal{Y}$. At some point in an execution of $\mathcal{A}^{\mathcal{O}}$, we say that we reprogram $\mathcal{O}$ to output $y \in \mathcal{Y}$ on $x \in \mathcal{X}$ if we update the oracle to compute the function $H_{x,y}$ defined by*

$$H_{x,y}(x') := \begin{cases} y & \text{if } x' = x \\ H(x') & \text{otherwise} \end{cases}$$

where H is the function computed by $\mathcal{O}$ before the update. This updated oracle is used in the rest of execution of $\mathcal{A}$. We denote by $\mathcal{O} \leftarrow \mathsf{Reprogram}(\mathcal{O}, x, y)$ the above reprogramming procedure.

Lemma 2.10. *(Measure-and-Reprogram Lemma, Rephrasing of [DFM20, Lemma 4]) Let $\mathcal{X}$, $\mathcal{Y}$, and $\mathcal{Z}$ be sets of classical strings and k be a positive integer. Let $\mathcal{A}$ be a q-quantum-query algorithm that is given quantum oracle access to an oracle that computes a function from $\mathcal{X}$ to $\mathcal{Y}$ and a (possibly quantum) input inp and outputs $\mathbf{x} \in \mathcal{X}^k$ and $z \in \mathcal{Z}$. For a function $H : \mathcal{X} \to \mathcal{Y}$ and $\mathbf{y} = (y_1, ..., y_k) \in \mathcal{Y}^k$, we define a measure-and-reprogram algorithm $\tilde{\mathcal{A}}[H, \mathbf{y}]$ as follows:*

$\widetilde{A}[H, \mathbf{y}](\mathsf{inp})$**:** *Given a (possibly quantum) input inp, it works as follows:*

1. *For each $i \in [k]$, uniformly pick $(j_i, b_i) \in ([q] \times \{0, 1\}) \cup \{(\bot, \bot)\}$ conditioned on that there exists at most one $i \in [k]$ such that $j_i = j^*$ for all $j^* \in [q]$.*
2. *Run $\mathcal{A}^{\mathcal{O}}(\mathsf{inp})$ where the oracle $\mathcal{O}$ is initialized to be a quantumly-accessible classical oracle that computes H, and when $\mathcal{A}$ makes its j-th query, the oracle is simulated as follows:*
   (a) *If $j = j_i$ for some $i \in [k]$, measure $\mathcal{A}$'s query register to obtain x'_i, and do either of the following.*
      i. *If $b_i = 0$, reprogram $\mathcal{O} \leftarrow \mathsf{Reprogram}(\mathcal{O}, x'_i, y_i)$ and answer $\mathcal{A}$'s j_i-th query by using the reprogrammed oracle.*
      ii. *If $b_i = 1$, answer $\mathcal{A}$'s j_i-th query by using the oracle before the reprogramming and then reprogram $\mathcal{O} \leftarrow \mathsf{Reprogram}(\mathcal{O}, x'_i, y_i)$.*
   (b) *Otherwise, answer $\mathcal{A}$'s j-th query by just using the oracle $\mathcal{O}$ without any measurement or reprogramming.*
3. *Let $(\mathbf{x} = (x_1, ..., x_k), z)$ be $\mathcal{A}$'s output.*
4. *For all $i \in [k]$ such that $j_i = \bot$, set $x'_i := x_i$.*
5. *Output $\mathbf{x}' := ((x'_1, ..., x'_k), z)$.*

Then for any q-quantum query algorithm $\mathcal{A}$, inp, $H : \mathcal{X} \to \mathcal{Y}$, $\mathbf{x}^* = (x^*_1, ..., x^*_k) \in \mathcal{X}^k$ such that $x^*_i \neq x^*_{i'}$ for all $i \neq i'$, $\mathbf{y} = (y_1, ..., y_k) \in \mathcal{Y}^k$, and a relation $R \subseteq \mathcal{X}^k \times \mathcal{Y}^k \times \mathcal{Z}$, we have

$$\begin{aligned} &\Pr[\mathbf{x}' = \mathbf{x}^* \wedge (\mathbf{x}', \mathbf{y}, z) \in R : (\mathbf{x}', z) \xleftarrow{\$} \widetilde{A}[H, \mathbf{y}](\mathsf{inp})] \\ &\quad \geq \frac{1}{(2q+1)^{2k}} \Pr[\mathbf{x} = \mathbf{x}^* \wedge (\mathbf{x}, \mathbf{y}, z) \in R : (\mathbf{x}, z) \xleftarrow{\$} \mathcal{A}^{H_{\mathbf{x}^*, \mathbf{y}}}(\mathsf{inp})]. \end{aligned}$$

where $H_{\mathbf{x}^*,\mathbf{y}} : \mathcal{X} \to \mathcal{Y}$ *is defined as*

$$H_{\mathbf{x}^*,\mathbf{y}}(x') := \begin{cases} y_i & \text{if } \exists i \in [k] \text{ s.t. } x' = x^*_i \\ H(x') & \text{otherwise} \end{cases}.$$

Remark 2. *The above lemma is a rephrasing of [DFM20, Lemma 4] (taking [DFM20, Remark 5] into account) given in [YZ20, Definition 4.5, Lemma 4.6].*[12]

Especially, we will rely on the following special case of Lemma 2.10.

Lemma 2.11 (Measure-and-Reprogram Lemma, Ordered Queries)**.** *Let* $\mathcal{X}$ *and* $\mathcal{Y}$ *be sets of classical strings and* k *be a positive integer. Let* $\mathcal{A}$ *be a* q*-quantum-query algorithm that is given quantum oracle access to an oracle that computes a function from* $\mathcal{X}$ *to* $\mathcal{Y}$ *and outputs* $\mathbf{x} \in \mathcal{X}^k$*. For a function* $H : \mathcal{X}^{\le k} \to \mathcal{Y}$ *and* $\mathbf{y} = (y_1, ..., y_k) \in \mathcal{Y}^k$*, we define an algorithm* $\widetilde{A}^{\mathsf{ord}}[H, \mathbf{y}]$ *as follows:*

$\widetilde{A}^{\mathsf{ord}}[H, \mathbf{y}]$**:** *It works as follows:*

1. *For each* $i \in [k]$*, uniformly pick* $(j_i, b_i) \in ([q] \times \{0, 1\}) \cup \{(\bot, \bot)\}$ *conditioned on that there exists at most one* $i \in [k]$ *such that* $j_i = j^*$ *for all* $j^* \in [q]$*.*
2. *Run* $\mathcal{A}^{\mathcal{O}}$ *where the oracle* $\mathcal{O}$ *is initialized to be a quantumly-accessible classical oracle that computes* H*, and when* $\mathcal{A}$ *makes its* j*-th query, the oracle is simulated as follows:*
   (a) *If* $j = j_i$ *for some* $i \in [k]$*, measure* $\mathcal{A}$*'s query register to obtain* $\mathbf{x}'_i = (x'_{i,1}, ..., x'_{i,k_i})$ *where* $k_i \le k$ *is determined by the measurement outcome, and do either of the following.*
      i. *If* $b_i = 0$*, reprogram* $\mathcal{O} \leftarrow \mathsf{Reprogram}(\mathcal{O}, \mathbf{x}'_i, y_i)$ *and answer* $\mathcal{A}$*'s* j_i*-th query by using the reprogrammed oracle.*
      ii. *If* $b_i = 1$*, answer* $\mathcal{A}$*'s* j_i*-th query by using the oracle before the reprogramming and then reprogram* $\mathcal{O} \leftarrow \mathsf{Reprogram}(\mathcal{O}, \mathbf{x}'_i, y_i)$*.*
   (b) *Otherwise, answer* $\mathcal{A}$*'s* j*-th query by just using the oracle* $\mathcal{O}$ *without any measurement or reprogramming.*
3. *Let* $\mathbf{x} = (x_1, ..., x_k)$ *be* $\mathcal{A}$*'s output.*
4. *For all* $i \in [k]$ *such that* $j_i = \bot$*, set* $\mathbf{x}'_i = \mathbf{x}_i$ *where* $\mathbf{x}_i := (x_1, \cdots, x_i)$*.*
5. *Output* $\mathbf{x}'_k$ *if for all* $i \in [k]$*,* $\mathbf{x}'_i$ *is a prefix of* $\mathbf{x}'_k$*, and output* $\bot$ *otherwise.*

Then for any q*-quantum query algorithm* $\mathcal{A}$*,* $H : \mathcal{X}^{\le k} \to \mathcal{Y}$*,* $\mathbf{x}^* = (x^*_1, ..., x^*_k) \in \mathcal{X}^k$*, and* $\mathbf{y} = (y_1, ..., y_k)$*, we have*

$$\Pr[\widetilde{A}^{\mathsf{ord}}[H, \mathbf{y}] = \mathbf{x}^*] \ge \frac{1}{(2q+1)^{2k}} \Pr\left[\mathcal{A}^{H^{\mathsf{ord}}_{\mathbf{x}^*,\mathbf{y}}} = \mathbf{x}^*\right]$$

where $H^{\mathsf{ord}}_{\mathbf{x}^*,\mathbf{y}} : \mathcal{X}^{\le k} \to \mathcal{Y}$ *is defined as*

$$H^{\mathsf{ord}}_{\mathbf{x}^*,\mathbf{y}}(\mathbf{x}') := \begin{cases} y_i & \text{if } \exists i \in [k] \text{ s.t. } \mathbf{x}' = (x^*_1, ..., x^*_i) \\ H(\mathbf{x}') & \text{otherwise} \end{cases}.$$

Proof. We apply Lemma 2.10 for the following setting, where we append [2.10] and [2.11] to characters to distinguish those in Lemma 2.10 and 2.11, respectively (e.g., $\mathcal{X}$[2.10] and $\mathcal{X}$[2.11] are $\mathcal{X}$ in Lemma 2.10 and 2.11, respectively).

[12] Note a minor notational difference from [YZ20] that the roles of i and j are swapped.

- $\mathcal{X}[2.10] := \mathcal{X}[2.11]^{\leq k}$, $\mathcal{Y}[2.10] := \mathcal{Y}[2.11]$, $\mathcal{Z}[2.10] := \emptyset$
- $\mathsf{inp}[2.10]$ is a null string and $H[2.10] := H[2.11]$.
- $\mathbf{x}^*[2.10] := (x_1^*[2.11], (x_1^*[2.11], x_2^*[2.11]), ..., (x_1^*[2.11], ..., x_k^*[2.11]))$.
- $\mathcal{A}[2.10]$ works similarly to $\mathcal{A}[2.11]$ except that $\mathcal{A}[2.10]$ outputs

$$(x_1[2.10], ..., x_k[2.10]) := (x_1[2.11], (x_1[2.11], x_2[2.11]), ..., (x_1[2.11], ..., x_k[2.11]))$$

  where $(x_1[2.11], ..., x_k[2.11])$ is the output of $\mathcal{A}[2.11]$.
- $R[2.10]$ is a trivial relation, i.e., $R[2.10] := \mathcal{X}[2.10]^k \times \mathcal{Y}[2.10]^k$.

Then we have $H_{\mathbf{x}^*,\mathbf{y}}[2.10] = H^{\mathsf{ord}}_{\mathbf{x}^*,\mathbf{y}}[2.11]$ are defined as the same functions in Lemmas 2.10 and 2.11, and thus the r.h.s. of the inequalities in Lemmas 2.10 and 2.11 are the same probability. The l.h.s of the inequality in Lemma 2.10 corresponds to the probability that we have $\mathbf{x}'_i = (x_1^*, ..., x_i^*)$ for all $i \in [k]$ such that $\mathbf{x}'_i$ is defined (i.e., $i = k$ or $j_i \neq \bot$) where the probability is taken over randomness of $\widetilde{A}^{\mathsf{ord}}[H, \mathbf{y}]$. Especially, when this event happens, we have $\mathbf{x}'_k = \mathbf{x}^*$ and $\mathbf{x}'_i$ is a prefix of $\mathbf{x}'_k$ for all $i \in [k]$ such that $j_i \neq \bot$, i.e., $\widetilde{A}^{\mathsf{ord}}[H, \mathbf{y}] = \mathbf{x}^*$. Therefore, Lemma 2.10 implies Lemma 2.11.

□

3 Impossibility of BB ZK for Constant-Round Arguments

In this section, we prove the following theorem.

Theorem 3.1. *If there exists a constant-round post-quantum black-box zero-knowledge argument for a language L, then $L \in \mathbf{BQP}$.*

The rest of this section is devoted to prove the above theorem. Specifically, we prove the theorem by the following steps:

1. In Section 3.1, we prove the impossibility for a strict-polynomial-time simulator that works for inefficient verifiers. (See the paragraph after Definition 2.2 for the meaning of that "a simulator works for inefficient verifiers".) This part can be seen as a quantum version of the classical impossibility result of [BL02] (except that we consider inefficient verifiers).
2. In Section 3.2, we prove the impossibility for a *expected-polynomial-time simulator* that works for inefficient verifiers. This is proven by reducing it to the strict-polynomial-time case, which relies on a novel inherently quantum technique.
3. In Section 3.3, we prove Theorem 3.1, i.e., the impossibility for a expected-polynomial-time simulator that only works for *efficient* verifiers. This part is basically done by replacing a random function with $2q$-wise independent function relying on Lemma 2.4, where some delicate argument is needed due to technical reasons.

First, we define notations that are used throughout this section. Let $\Pi = (P, V)$ be a classical constant-round interactive argument for a language L. Without loss of generality, we assume that P sends the first message, and let (P, V) be $(2k-1)$-round protocol where P sends $k = O(1)$ messages in the protocol and V sends $(k-1)$ messages.

We assume that all messages sent between P and V are elements of a classical set $\mathcal{M}$ (e.g., we can take $\mathcal{M} := \{0,1\}^\ell$ for sufficiently large ℓ). Let $\mathcal{R}$ be V's randomness space. For any fixed

statement x and randomness $r \in \mathcal{R}$, V's message in $(2i+1)$-th round can be seen as a deterministic function of $(m_1, ..., m_i)$ where $(m_1, ..., m_i)$ are prover's first i messages. Similarly, V's final decision can be seen as a deterministic function of all prover's messages $(m_1, ..., m_k)$. We denote this function by $F[x, r] : \mathcal{M}^{\leq k} \to \mathcal{M} \cup \{\top, \bot\}$. Here, $F[x, r]$ outputs an element of $\mathcal{M}$ if the input is in $\mathcal{M}^{\leq k-1}$ and outputs $\top$ or $\bot$ if the input is in $\mathcal{M}^k$. We denote by $\mathsf{Acc}[x, r] \subseteq \mathcal{M}^k$ the subset consisting of all $(m_1, ..., m_k)$ such that $F[x, r](m_1, ..., m_k) = \top$, i.e, all accepting transcripts corresponding to the statement x and randomness r. A quantum algorithm having black-box access to V means the algorithm has quantum superposition access to the function $F[x, r]$.

3.1 Strict-Polynomial-Time Simulation for Inefficient Verifier

Our first goal is to prove the impossibility of strict-polynomial-time black-box simulation for inefficient verifiers. Assuming $\Pi = (P, V)$ is a constant-round post-quantum black-box zero-knowledge argument for a language L. We will show the impossibility for *random aborting verifiers*, which works similarly to the honest verifier V except that it aborts with probability $1-\epsilon$ for some $\epsilon \in [0, 1]$ in each round. More precisely, for any $\epsilon \in [0, 1]$, we consider a malicious verifier V^* with an auxiliary input $|\psi_\epsilon\rangle_{\mathbf{R},\mathbf{H}}$ as follows.

Intuitively, V^* works in the same way as V except on each input $(m_1, \cdots, m_i) \in \mathcal{M}^{\leq k}$, it returns $F[x, r](m_1, \cdots, m_i)$ with probability ϵ and aborts (outputs $\bot$) with probability $1 - \epsilon$. Therefore, V^* prepares randomness r for V's decision function $F[x, r]$ and a random function H that decides if it outputs $\bot$. Here H is drawn from a distribution $\mathcal{H}_\epsilon$ where $\mathcal{H}_\epsilon$ is a distribution over $H : \mathcal{M}^{\leq k} \to \{0, 1\}$ such that we have $\Pr[H(m_1, ..., m_i) = 1] = \epsilon$ independently for each $(m_1, ..., m_i) \in \mathcal{M}^{\leq k}$.

$|\psi_\epsilon\rangle_{\mathbf{R},\mathbf{H}}$ is a superposition of (r, H) according to the distribution $\{(r, H) : r \xleftarrow{\$} \mathcal{R}, H \xleftarrow{\$} \mathcal{H}_\epsilon\}$ Formally,

$$|\psi_\epsilon\rangle_{\mathbf{R},\mathbf{H}} = \sum_{r\in\mathcal{R}, H\in\mathsf{Func}(\mathcal{M}^{\leq k},\{0,1\})} \sqrt{\frac{D(H)}{|\mathcal{R}|}} |r, H\rangle_{\mathbf{R},\mathbf{H}}$$

where D is the density function corresponding to $\mathcal{H}_\epsilon$. Here, H is represented as a concatenation of function values $H(\boldsymbol{m})$ for all $\boldsymbol{m} \in \mathcal{M}^{\leq k}$. We denote by $\mathbf{H}_{\boldsymbol{m}}$ the sub-register of $\mathbf{H}$ that stores $H(\boldsymbol{m})$.

Remark 3. *Note that the state $|\psi_\epsilon\rangle_{R,H}$ is exponentially large (of length $O(|\mathcal{M}|^k)$). Therefore our malicious verifier V^* (which uses this state as its auxiliary input) is inefficient.*

V^* works over its internal register $(\mathbf{X}, \mathbf{Aux} = (\mathbf{R}, \mathbf{H}), \mathbf{W} = (\mathbf{Count}, \mathbf{M}_1, ..., \mathbf{M}_k, \mathbf{B}))$ and an additional message register $\mathbf{M}$. We define the output register as $\mathbf{Out} := \mathbf{B}$. It works as follows where $\mathbf{Count}$ stores a non-negative integer smaller than k (i.e. $\{0, 1, \cdots, k-1\}$), each register of $\mathbf{M}_1, ..., \mathbf{M}_k$ and $\mathbf{M}$ stores an element of $\mathcal{M}$ and $\mathbf{B}$ stores a single bit. $\mathbf{M}$ is the register to store messages from/to external prover, and $\mathbf{M}_i$ is a register to record the i-th message from the prover.

We next explain the unitary U^* for V^*. The interaction between V^* and the honest prover P has been formally defined in Section 2.1. We recall it here.

1. V^* takes inputs a statement x and a quantum auxiliary input $|\psi_\epsilon\rangle$: $\mathbf{X}$ is initialized to be $|x\rangle_{\mathbf{X}}$ where x is the statement to be proven, $\mathbf{Aux}$ is initialized to be $|\psi_\epsilon\rangle_{\mathbf{R},\mathbf{H}}$, and all other registers are initialized to be 0.

2. *Verifier V^* on round $< k$*: Upon receiving the i-th message from P for $i < k$ in $\mathbf{M}$, swap $\mathbf{M}$ and $\mathbf{M}_i$ and increment the value in $\mathbf{Count}$[13]. We note that V^* can know i since it keeps track of which round it is playing by the value in $\mathbf{Count}$. Let $(m_1, ..., m_i)$ be the messages sent from P so far. Then do the following in superposition where $(m_1, ..., m_i)$, r, H and m are values in registers $(\mathbf{M}_1, ..., \mathbf{M}_i)$, $\mathbf{R}$, $\mathbf{H}$ and $\mathbf{M}$:

   - If $H(m_1, ..., m_i) = 0$, then do nothing.
   - If $H(m_1, ..., m_i) = 1$, then add $F[x, r](m_1, ..., m_i)$ to the message register m.

   It then measures the register $\mathbf{M}$ and sends the result to the prover P.

   Unitary U^* on $\mathbf{Count} < k-1$: It acts on registers $\mathbf{Count}$ (whose value is less than $k-1$), $\mathbf{X}, (\mathbf{M}_1, ..., \mathbf{M}_k)$, $\mathbf{R}$, $\mathbf{H}$ and $\mathbf{M}$:

   - It reads the value j in $\mathbf{Count}$ and increments it to $i := j + 1$. It swaps $\mathbf{M}$ and $\mathbf{M}_i$ (in superposition).
   - Let $x, (m_1, \cdots, m_i), r, H$ and m be the values in registers $\mathbf{X}, (\mathbf{M}_1, \cdots, \mathbf{M}_i), \mathbf{R}, \mathbf{H}$ and $\mathbf{M}$. Let $F^*[x, r, H]$ be the following function: on input $(m_1, \cdots, m_i) \in \mathcal{M}^{\leq k-1}$,

$$F^*[x, r, H](m_1, ..., m_i) := \begin{cases} F[x, r](m_1, ..., m_i) & \text{if } H(m_1, ..., m_i) = 1 \\ 0 & \text{otherwise} \end{cases}.$$

     It then applies the function in superposition.

$$|x, m_1, \cdots, m_i, r, H, m\rangle \to |x, m_1, \cdots, m_i, r, H, m + F^*[x, r, H](m_1, \cdots, m_i)\rangle.$$

3. *Verifier V^* on round k*: Upon receiving the k-th message from P in $\mathbf{M}$, swap $\mathbf{M}$ and $\mathbf{M}_k$ and increment the value in $\mathbf{Count}$. Then flip the bit in $\mathbf{B}$ if $(m_1, ..., m_k) \in \mathsf{Acc}[x, r]$ and $H(m_1, ..., m_i) = 1$ for all $i \in [k]$ where $(m_1, ..., m_k)$, r, and H are values in registers $(\mathbf{M}_1, ..., \mathbf{M}_k)$, $\mathbf{R}$, and $\mathbf{H}$.

   Unitary U^* on $\mathbf{Count} = k-1$: It acts on registers $\mathbf{Count}$ (whose value is exactly equal to $k-1$), $\mathbf{X}, (\mathbf{M}_1, ..., \mathbf{M}_k)$, $\mathbf{R}$, $\mathbf{H}$ and $\mathbf{B}$:

   - It reads the value $j = k-1$ in $\mathbf{Count}$ and sets it to 0. It swaps $\mathbf{M}$ and $\mathbf{M}_k$ (in superposition).
   - Let $x, (m_1, \cdots, m_k), r, H$ and b be the values in registers $\mathbf{X}, (\mathbf{M}_1, \cdots, \mathbf{M}_k), \mathbf{R}, \mathbf{H}$ and $\mathbf{B}$. Let $F^*[x, r, H]$ be the following function: on input $(m_1, \cdots, m_k) \in \mathcal{M}^k$,

$$F^*[x, r, H](m_1, ..., m_k) := \begin{cases} 1 & \text{if } H(m_1,...,m_i)=1, \forall i \in [k] \\ & \text{and } F[x,r](m_1,\cdots,m_k)=\top \\ 0 & \text{otherwise} \end{cases}.$$

     It applies the function in superposition.

$$|x, m_1, \cdots, m_k, r, H, b\rangle \to |x, m_1, \cdots, m_k, r, H, b + F^*[x, r, H](m_1, \cdots, m_k)\rangle.$$

With the description of V^* above, we have the following observation.

Observation 1. *Let $M_{\mathbf{R},\mathbf{H}}$ be the measurement on registers $\mathbf{R}, \mathbf{H}$. For any (inefficient) black-box simulator Sim it has zero advantage of distinguishing if it has black-box access to $V^*(x; |\psi\rangle_{\mathbf{R},\mathbf{H}})$ or $V^*(x; M_{\mathbf{R},\mathbf{H}} \circ |\psi\rangle_{\mathbf{R},\mathbf{H}})$.*

[13] More precisely, this maps $|i\rangle$ to $|(i+1) \bmod k\rangle$.

Observation 1 says that even for an unbounded simulator with black-box access to $V^*(x;|\psi\rangle_{\mathbf{R},\mathbf{H}})$, it has no way to tell if the auxiliary input $|\psi\rangle_{\mathbf{R},\mathbf{H}}$ gets measured at the beginning or never gets measured.

This is because registers $\mathbf{H}$ and $\mathbf{R}$ are only used as control qubits throughout the execution of $\mathsf{Sim}^{V^*(x;|\psi_\epsilon\rangle)}(x)$, we can trace out registers $\mathbf{R},\mathbf{H}$ while preserving the behavior of this simulator.

Observation 2. *V^* can be simulated giving oracle access to $F^*[x,r,H]$.*

This can be easily seen from the description of U^* above.

Observation 3. *Given x and r, a quantum oracle that computes $F^*[x,r,H]$ can be simulated by $2k$ quantum oracle access to H.*

This is can be easily seen from the definition of $F^*[x,r,H]$. Note that we require $2k$ queries instead of k queries since we need to compute k values of H to compute $F^*[x,r,H]$ (when the input is in $\mathcal{M}^k$) and then need to uncompute them.

We then prove the following lemma.

Lemma 3.2. *For any $q=\mathsf{poly}(\lambda)$ there exists noticeable ϵ^*_q such that the following holds for any noticeable $\epsilon\le\epsilon^*_q$: if there exists a quantum black-box simulator Sim that makes at most q quantum queries, such that we have*

$$\Pr\left[M_{\mathbf{B}}\circ\mathsf{OUT}_{V^*}\left(\mathsf{Sim}^{V^*(x;|\psi_\epsilon\rangle)}(x)\right)=1\right]\ge\frac{\epsilon^k}{4}-\mathsf{negl}(\lambda)$$

for all $x\in L\cap\{0,1\}^\lambda$ where $M_{\mathbf{B}}$ means measuring and outputting the register $M_{\mathbf{B}}$, then we have $L\in\mathbf{BQP}$.

The above lemma immediately implies the following corollary, which can be seen as the quantum generalization of the result of [BL02].

Corollary 3.3. *If there exists a constant-round post-quantum black-box zero-knowledge argument for a language L with a simulator that makes a fixed polynomial number of queries and works for all possibly inefficient verifiers, then $L\in\mathbf{BQP}$.*

Proof. Let $\Pi=(P,V)$ be a constant-round post-quantum black-box zero-knowledge argument for a language L where P sends $k=O(1)$ messages and V^* and $|\psi_\epsilon\rangle$ are defined above. Suppose that there exists a fixed polynomial $q=\mathsf{poly}(\lambda)$ such that there exists a quantum black-box simulator Sim for Π that works for all possibly inefficient verifiers. When V^* takes an auxiliary input $|\psi_\epsilon\rangle$, it can be seen as a verifier that works similarly to the honest verifier except that it aborts with probability $1-\epsilon$ in each round, and $\mathbf{B}$ takes 1 if and only if it does not abort until the end of the protocol. Therefore, for any $x\in L\cap\{0,1\}^\lambda$ and its witness $w\in R_L(x)$, we have

$$\Pr[M_{\mathbf{B}}\circ\mathsf{OUT}_{V^*}(\langle P(w),V^*(|\psi_\epsilon\rangle)\rangle(x))=1]=\epsilon^k.$$

By the zero-knowledge property, the above equality implies

$$\Pr\left[M_{\mathbf{B}}\circ\mathsf{OUT}_{V^*}\left(\mathsf{Sim}^{V^*(x;|\psi_\epsilon\rangle)}(x)\right)=1\right]=\epsilon^k-\mathsf{negl}(\lambda)\ge\frac{\epsilon^k}{4}-\mathsf{negl}(\lambda).$$

Since this holds for arbitrary ϵ, by Lemma 3.2, this implies $L\in\mathbf{BQP}$.

□

Remark 4. *The above proof assumes that the simulator works for possibly inefficient verifiers since V^* is inefficient as noted in Remark 3. Actually, we can generalize it to rule out a strict-polynomial-query simulator that only works for efficient verifiers by considering an efficient variant of V^* that is indistinguishable from V^* from the view of Sim that makes at most q queries. Since this generalized version is also subsumed by Theorem 3.1, we omit the details.*

Then we prove Lemma 3.2.

Proof of Lemma 3.2. By Observation 1, we can assume $|\psi_\epsilon\rangle$ is measured at the beginning. In other words, the auxiliary state is sampled as $|r\rangle_{\mathbf{R}}|H\rangle_{\mathbf{H}}$ for $r \xleftarrow{\$} \mathcal{R}, H \xleftarrow{\$} \mathcal{H}_\epsilon$. Once r and H are fixed, the unitary U^* (corresponding to V^*) and its inverse can be simulated by a single quantum access to a classical function $F^*[x,r,H]$ (defined in the description of V^*).

Moreover, if we let $\mathsf{Acc}^*[x,r,H] \subseteq \mathsf{Acc}[x,r]$ be the set of $\boldsymbol{m} = (m_1,...,m_k) \in \mathsf{Acc}[x,r]$ such that $H(m_1,...,m_i) = 1$ for all $i \in [k]$, after the execution of $\mathsf{Sim}^{V^*(x;|r\rangle|H\rangle)}(x)$, $\mathbf{B}$ contains 1 if and only if $(\mathbf{M}_1,...,\mathbf{M}_k)$ contains an element in $\mathsf{Acc}^*[x,r,H]$. Therefore, for proving Lemma 3.2, it suffices to prove the following lemma.

Lemma 3.4. *For any $q = \mathsf{poly}(\lambda)$ there exists noticeable ϵ_q^* such that the following holds for any noticeable $\epsilon \leq \epsilon_q^*$: if there exists an oracle-aided quantum algorithm $\mathcal{S}$ that makes at most q quantum queries,[14] such that we have*

$$\Pr_{r\xleftarrow{\$}\mathcal{R},H\xleftarrow{\$}\mathcal{H}_\epsilon}\left[\mathcal{S}^{F^*[x,r,H]}(x) \in \mathsf{Acc}^*[x,r,H]\right] \geq \frac{\epsilon^k}{4} - \mathsf{negl}(\lambda)$$

for all $x \in L \cap \{0,1\}^\lambda$, then we have $L \in \mathbf{BQP}$.

We prove the above lemma below. Assuming Lemma 3.4, we show Lemma 3.2 holds. For any quantum black-box simulator Sim that makes at most q quantum queries, such that

$$\Pr\left[M_{\mathbf{B}} \circ \mathsf{OUT}_{V^*}\left(\mathsf{Sim}^{V^*(x;|\psi_\epsilon\rangle)}(x)\right) = 1\right] \geq \frac{\epsilon^k}{4} - \mathsf{negl}(\lambda).$$

By Observation 1, we have

$$\begin{aligned}&\Pr_{r\xleftarrow{\$}\mathcal{R},H\xleftarrow{\$}\mathcal{H}_\epsilon}\left[M_{\mathbf{B}} \circ \mathsf{OUT}_{V^*}\left(\mathsf{Sim}^{V^*(x;|r,H\rangle)}(x)\right) = 1\right]\\ =&\Pr\left[M_{\mathbf{B}} \circ \mathsf{OUT}_{V^*}\left(\mathsf{Sim}^{V^*(x;|\psi_\epsilon\rangle)}(x)\right) = 1\right] \geq \frac{\epsilon^k}{4} - \mathsf{negl}(\lambda).\end{aligned}$$

Finally, we note that $M_{\mathbf{B}} \circ \mathsf{OUT}_{V^*}\left(\mathsf{Sim}^{V^*(x;|r,H\rangle)}(x)\right)$ can be computed by only having black-box access to $F^*[x,r,H]$ (by Observation 2). It outputs 1 (the register $\mathbf{B}$ is 1) if and only if the values $(m_1,\cdots,m_k)$ in $\mathbf{M}_1,\cdots,\mathbf{M}_k$ are in $\mathsf{Acc}^*[x,r,H]$. Thus, there is an algorithm $\mathcal{S}$ that computes $M_{\mathbf{B}} \circ \mathsf{Sim}^{V^*(x;|r,H\rangle)}(x)$ and measures registers $\mathbf{M}_1,\cdots,\mathbf{M}_k$. Such an algorithm $\mathcal{S}$ satisfies the requirement in Lemma 3.4. Therefore L is in $\mathbf{BQP}$.

□

The remaining part is to prove Lemma 3.4.

[14] Though $\mathcal{S}$ can be seen as a quantum black-box simulator for a malicious classical verifier, we do not call it a simulator since this deviates our syntax of quantum black-box simulators defined in Section 2.1

Proof of Lemma 3.4. We let $\epsilon_q^* := 1/(256k^2q^2(4kq+1)^{2k})$ and $\epsilon \leq \epsilon_q^*$ be an arbitrary noticeable function in λ. In the following, we simply write r and H in subscripts of probabilities to mean $r \xleftarrow{\$} \mathcal{R}$ and $H \xleftarrow{\$} \mathcal{H}_\epsilon$ for notational simplicity. As observed in Observation 3, $F^*[x,r,H]$ can be simulated by $2k$ quantum invocations of H if we know x and r. Therefore, we can view $\mathcal{S}^{F^*[x,r,H]}(x)$ as an oracle-aided algorithm with quantum access to H in which x and r are hardwired, and makes at most $2kq$ queries to H. We denote this algorithm by $\mathcal{A}[x,r]^H$. Then for any $x \in L \cap \{0,1\}^\lambda$, we have

$$\Pr_{r,H}\left[\mathcal{A}[x,r]^H \in \mathsf{Acc}^*[x,r,H]\right] = \Pr_{r,H}\left[\mathcal{S}^{F^*[x,r,H]}(x) \in \mathsf{Acc}^*[x,r,H]\right] \geq \frac{\epsilon^k}{4} - \mathsf{negl}(\lambda). \tag{1}$$

We apply Lemma 2.11 to $\mathcal{A}[x,r]$. For any x, r, H, $\boldsymbol{m}^* = (m_1^*, ..., m_k^*) \in \mathcal{M}^k$, and $\boldsymbol{\beta} = (\beta_1, ..., \beta_k) \in \{0,1\}^k$, we have

$$\Pr\left[\widetilde{\mathcal{A}[x,r]}^{\mathsf{ord}}[H, \boldsymbol{\beta}] = \boldsymbol{m}^*\right] \geq \frac{1}{(2\cdot 2kq+1)^{2k}} \Pr\left[\mathcal{A}[x,r]^{H^{\mathsf{ord}}_{\boldsymbol{m}^*,\boldsymbol{\beta}}} = \boldsymbol{m}^*\right] \tag{2}$$

where $\widetilde{\mathcal{A}[x,r]}^{\mathsf{ord}}[H,\boldsymbol{\beta}]$ and $H^{\mathsf{ord}}_{\boldsymbol{m}^*,\boldsymbol{\beta}}$ are as defined in Lemma 2.11 and $2kq$ is the number of queries to H made by $\mathcal{A}[x,r]$.

Let $\mathbf{1} := (1, ..., 1) \in \{0,1\}^k$ and $H_0 : \mathcal{M}^{\leq k} \to \{0,1\}$ be the zero-function i.e., $H_0(m_1, ..., m_i) = 0$ for all $(m_1, ..., m_i) \in \mathcal{M}^{\leq k}$. Then we prove the following claims.

Claim 3.5. *For any $x \in L \cap \{0,1\}^\lambda$, we have*

$$\Pr_r\left[\widetilde{\mathcal{A}[x,r]}^{\mathsf{ord}}[H_0, \mathbf{1}] \in \mathsf{Acc}[x,r]\right] \geq \frac{1}{8(4kq+1)^{2k}} - \mathsf{negl}(\lambda).$$

Claim 3.6. *For any $x \in \{0,1\}^\lambda \setminus L$, we have*

$$\Pr_r\left[\widetilde{\mathcal{A}[x,r]}^{\mathsf{ord}}[H_0, \mathbf{1}] \in \mathsf{Acc}[x,r]\right] = \mathsf{negl}(\lambda).$$

Roughly, we prove Claim 3.5 by using the ordered version of measure-and-reprogram lemma (Lemma 2.11) and Claim 3.6 by reducing to soundness of the protocol Π. Proofs of these claims are given later.

In the rest of this proof, we prove Lemma 3.4 assuming that Claim 3.5 and 3.6 are true. We construct a QPT algorithm $\mathcal{B}$ that decides L.

$\mathcal{B}$ takes x as input, and its goal is to decide if $x \in L$. It randomly chooses $r \xleftarrow{\$} \mathcal{R}$, runs $\widetilde{\mathcal{A}[x,r]}^{\mathsf{ord}}[H_0, \mathbf{1}]$, and outputs 1 if the output is in $\mathsf{Acc}[x,r]$ (by computing $F[x,r]$).

By Claim 3.5, for any $x \in L \cap \{0,1\}^\lambda$, we have

$$\Pr[\mathcal{B}(x) = 1] \geq \frac{1}{8(4kq+1)^{2k}} - \mathsf{negl}(\lambda).$$

On the other hand, by Claim 3.6 for any $x \in \{0,1\}^\lambda \setminus L$, we have

$$\Pr[\mathcal{B}(x) = 1] \leq \mathsf{negl}(\lambda).$$

This means $L \in \mathbf{BQP}$. This completes the proof of Lemma 3.4. □

What are left are proofs of Claim 3.5 and 3.6.

Proof of Claim 3.5. Noting that $\widetilde{\mathcal{A}[x,r]}^{\mathsf{ord}}[H,\boldsymbol{\beta}]$ can be seen as an oracle-aided algorithm that makes at most $2kq$ quantum queries to H, by the indistinguishability of sparse and zero functions (Lemma 2.5), for any $x, r, \boldsymbol{\beta}$, we have

$$\left|\Pr_H\left[\widetilde{\mathcal{A}[x,r]}^{\mathsf{ord}}[H,\boldsymbol{\beta}]\in\mathsf{Acc}[x,r]\right]-\Pr\left[\widetilde{\mathcal{A}[x,r]}^{\mathsf{ord}}[H_0,\boldsymbol{\beta}]\in\mathsf{Acc}[x,r]\right]\right|\le 32k^2q^2\epsilon. \tag{3}$$

Let D_ϵ be a distribution over $\{0,1\}^k$ whose each coordinate takes 1 with probability ϵ independently and $\widehat{H}(\boldsymbol{m}) := (H(m_1), H(m_1,m_2), ..., H(m_1,...,m_k))$ for $\boldsymbol{m}=(m_1,...,m_k)$. Then for any $x\in L\cap\{0,1\}^\lambda$, we have

$$\begin{aligned}
&\Pr_r\left[\widetilde{\mathcal{A}[x,r]}^{\mathsf{ord}}[H_0,\mathbf{1}]\in\mathsf{Acc}[x,r]\right]\\
\ge&\Pr_{r,H}\left[\widetilde{\mathcal{A}[x,r]}^{\mathsf{ord}}[H,\mathbf{1}]\in\mathsf{Acc}[x,r]\right]-32k^2q^2\epsilon\\
=&\sum_{\boldsymbol{m}^*\in\mathsf{Acc}[x,r]}\Pr_{r,H}\left[\widetilde{\mathcal{A}[x,r]}^{\mathsf{ord}}[H,\mathbf{1}]=\boldsymbol{m}^*\right]-32k^2q^2\epsilon\\
\ge&\frac{1}{(4kq+1)^{2k}}\sum_{\boldsymbol{m}^*\in\mathsf{Acc}[x,r]}\Pr_{r,H}\left[\boldsymbol{m}=\boldsymbol{m}^*:\boldsymbol{m}\xleftarrow{\$}\mathcal{A}[x,r]^{H^{\mathsf{ord}}_{\boldsymbol{m}^*,\mathbf{1}}}\right]-32k^2q^2\epsilon\\
=&\frac{\epsilon^{-k}}{(4kq+1)^{2k}}\sum_{\boldsymbol{m}^*\in\mathsf{Acc}[x,r]}\Pr_{r,H,\boldsymbol{\beta}\xleftarrow{\$}D_\epsilon}\left[\boldsymbol{m}=\boldsymbol{m}^*\wedge\boldsymbol{\beta}=\mathbf{1}:\boldsymbol{m}\xleftarrow{\$}\mathcal{A}[x,r]^{H^{\mathsf{ord}}_{\boldsymbol{m}^*,\boldsymbol{\beta}}}\right]-32k^2q^2\epsilon\\
=&\frac{\epsilon^{-k}}{(4kq+1)^{2k}}\sum_{\boldsymbol{m}^*\in\mathsf{Acc}[x,r]}\Pr_{r,H}\left[\boldsymbol{m}=\boldsymbol{m}^*\wedge\widehat{H}(\boldsymbol{m})=\mathbf{1}:\boldsymbol{m}\xleftarrow{\$}\mathcal{A}[x,r]^{H}\right]-32k^2q^2\epsilon\\
=&\frac{\epsilon^{-k}}{(4kq+1)^{2k}}\Pr_{r,H}\left[\boldsymbol{m}\in\mathsf{Acc}[x,r]\wedge\widehat{H}(\boldsymbol{m})=\mathbf{1}:\boldsymbol{m}\xleftarrow{\$}\mathcal{A}[x,r]^{H}\right]-32k^2q^2\epsilon\\
=&\frac{\epsilon^{-k}}{(4kq+1)^{2k}}\Pr_{r,H}\left[\mathcal{A}[x,r]^H\in\mathsf{Acc}^*[x,r,H]\right]-32k^2q^2\epsilon\\
\ge&\frac{\epsilon^{-k}}{(4kq+1)^{2k}}\left(\epsilon^k/4-\mathsf{negl}(\lambda)\right)-32k^2q^2\epsilon\\
=&\frac{1}{4(4kq+1)^{2k}}-32k^2q^2\epsilon-\mathsf{negl}(\lambda)\\
=&\frac{1}{8(4kq+1)^{2k}}-\mathsf{negl}(\lambda),
\end{aligned}$$

where the first inequality follows from Eq. 3 for $\boldsymbol{\beta}:=\mathbf{1}$, the second inequality follows from Eq. 2, the third inequality follows from Eq. 1, and the last equality follows from $\epsilon\le\epsilon^*_q = 1/(256k^2q^2(4kq+1)^{2k})$. This completes the proof of Claim 3.5. □

Proof of Claim 3.6. We consider a cheating prover P^* described as follows. Intuitively, P^* just runs $\widetilde{\mathcal{A}[x,r]}^{\mathsf{ord}}[H_0,\mathbf{1}]$ where r is chosen by the external verifier. We first look at how $\widetilde{\mathcal{A}[x,r]}^{\mathsf{ord}}[H_0,\mathbf{1}]$

runs: it runs $\mathcal{S}$ in the experiment and uses the oracle access to H and $F[x,r]$ to simulate the oracle $F^*[x,r,H]$.

The only difference between $\widetilde{\mathcal{A}[x,r]}^{\mathsf{ord}}[H_0,\mathbf{1}]$ and P^* is that $\widetilde{\mathcal{A}[x,r]}^{\mathsf{ord}}[H_0,\mathbf{1}]$ can compute $F[x,r]$ on its own because it samples and knows the random tape r but P^* does not know the randomness r of the honest verifier. However, we show that it can still answer queries to $F[x,r]$ because it needs r only when responding to measured queries, and P^* can then send such (classical) queries to the external verifier to get the response.

More precisely, P^* only makes queries to $F[x,r]$ on measured inputs $\boldsymbol{m}$, because only in this case the updated oracle in the game $\mathcal{O}(\boldsymbol{m})$ may not be 0; in all inputs $\boldsymbol{m}$, because $\mathcal{O}(\boldsymbol{m})$ is initialized as 0 and never gets updated in the experiment, the output of $F[x,r]$ on that input is not needed.

Formally, P^* is described as follows. We will mark the difference between P^* and $\widetilde{\mathcal{A}[x,r]}^{\mathsf{ord}}[H_0,\mathbf{1}]$ with underline.

$P^*(x)$**:** The cheating prover P^* interacts with the external verifier as follows:

1. For each $i\in[k]$, uniformly pick $(j_i,b_i)\in([2kq]\times\{0,1\})\cup\{(\bot,\bot)\}$ conditioned on that there exists at most one $i\in[k]$ such that $j_i=j^*$ for all $j^*\in[2kq]$.
2. Run $\mathcal{S}$ where the oracle $F^*[x,r,H]$ is simulated by additional oracles $\mathcal{O}$ and $\mathcal{F}$ where $\mathcal{O}$ and $\mathcal{F}$ play the roles of H and $F[x,r]$, respectively.
   The oracle $\mathcal{O}$ (for simulating H) and $\mathcal{F}$ (for simulating $F[x,r]$) are initialized to be an oracle that just return 0. For the rest of the description, we can assume $\mathcal{S}$ is now making queries to both $\mathcal{O}$ and $\mathcal{F}$.
   When $\mathcal{S}$ makes its j-th query to $\mathcal{O}$,
   (a) If $j=j_i$ for some $i\in[k]$, measure $\mathcal{S}$'s query register to obtain $\boldsymbol{m}'_i=(m'_{i,1},...,m'_{i,k_i})$ for some $k_i\le k$. If the transcript between V at this point is inconsistent to $\boldsymbol{m}'_i$ (i.e., there is $\ell\in[k_i]$ such that P^* already sent an ℓ-th message different from $m'_{i,\ell}$ to the external verifier), then just abort. Otherwise, run the protocol between the external verifier until $2k_i$-th round by using $(m'_{i,1},...,m'_{i,k_i})$ as the first k_i prover's messages. It then updates $\mathcal{F}$ such that for each $j\in[k_i]$, $\mathcal{F}$ on input $(m'_{i,1},\cdots,m'_{i,j})$ is compatible with the current transcript.
      i. If $b_i=0$, reprogram $\mathcal{O}\leftarrow\mathsf{Reprogram}(\mathcal{O},\boldsymbol{m}'_i,1)$ and answer $\mathcal{S}$'s j_i-th query by using the reprogrammed oracle.
      ii. If $b_i=1$, answer $\mathcal{S}$'s j_i-th query by using the oracle before the reprogramming and then reprogram $\mathcal{O}\leftarrow\mathsf{Reprogram}(\mathcal{O},\boldsymbol{m}'_i,1)$.
   (b) Otherwise, answer $\mathcal{S}$'s j-th query by just using the oracle $\mathcal{O}$.
   When $\mathcal{S}$ makes its query to $\mathcal{F}$, it uses the current updated oracle $\mathcal{F}$.
3. Let $\boldsymbol{m}=(m_1,...,m_k)$ be $\mathcal{S}$'s output. If the protocol between the external verifier has not been completed yet, complete the protocol by using messages $\boldsymbol{m}$. Again, if $\boldsymbol{m}$ is inconsistent to the transcript so far, just abort.

Note that since P^* uses the interaction with V to perfectly simulate the oracle access to $F[x,r]$, by definitions of $\widetilde{\mathcal{A}[x,r]}^{\mathsf{ord}}[H_0,\mathbf{1}]$, it is straightforward to see that P^* succeeds in letting V accept with probability at least $\Pr_r[\widetilde{\mathcal{A}[x,r]}^{\mathsf{ord}}[H_0,\mathbf{1}]\in\mathsf{Acc}[x,r]]$. Noting that $\widetilde{\mathcal{A}[x,r]}^{\mathsf{ord}}[H_0,\mathbf{1}]$ returns $\bot$ whenever any two of measured queries are inconsistent (i.e., one is not a prefix of the other), and thus when it does not return $\bot$, P^* does not abort either in the corresponding execution. Therefore, the negligible soundness of the protocol ensures $\Pr_r[\widetilde{\mathcal{A}[x,r]}^{\mathsf{ord}}[H_0,\mathbf{1}]\in\mathsf{Acc}[x,r]]=\mathsf{negl}(\lambda)$.

This completes the proof of Claim 3.6. □

3.2 Expected-Polynomial-Time Simulation for Inefficient Verifiers

In the previous section, we proved that strict-polynomial-time black-box simulation is impossible. In this section, as a first step to prove Theorem 3.1, we prove that even expected-polynomial-time black-box simulation is impossible if we require it to work for all *inefficient* malicious verifiers.

Theorem 3.7. *If there exists a constant-round post-quantum black-box zero-knowledge argument for a language L with a simulator that works for all inefficient malicious verifiers, then $L \in \mathbf{BQP}$.*

Though this theorem is subsumed by Theorem 3.1, we first prove this since the proof is simpler and thus we believe that it is easier for readers to understand the proof of Theorem 3.1 if we first give the proof of Theorem 3.7.

Our main idea is to consider a malicious verifier $\widetilde{V}^*$ that runs the honest verifier and a random aborting verifier in superposition. Roughly, $\widetilde{V}^*$ works over the same registers as those of V^* and one additional register $\mathbf{Cont}$ that stores 1-qubit that plays the role of a "control qubit". We define an auxiliary input

$$|\widetilde{\psi}_\epsilon\rangle_{\mathbf{Cont},\mathbf{R},\mathbf{H}} := \frac{1}{\sqrt{2}}\left(|0\rangle_{\mathbf{Cont}} + |1\rangle_{\mathbf{Cont}}\right) \otimes |\psi_\epsilon\rangle_{\mathbf{R},\mathbf{H}}$$

where $|\psi_\epsilon\rangle_{\mathbf{R},\mathbf{H}}$ is as defined in Section 3.1. Given a statement x and an auxiliary input $|\widetilde{\psi}_\epsilon\rangle_{\mathbf{Cont},\mathbf{R},\mathbf{H}}$, $\widetilde{V}^*$ runs the honest verifier V if the value in $\mathbf{Cont}$ is 0 and the random aborting verifier V^* if the value in $\mathbf{Cont}$ is 1 in superposition. Then it "adjusts $\mathbf{H}$" so that the states in $\mathbf{H}$ becomes the same in both cases of $\mathbf{Cont} = 0$ and $\mathbf{Cont} = 1$. The motivation of introducing this step is to make the final state in $\mathbf{Cont}$ be a pure state, which is essential for our analyses to work (in particular for latter Lemma 3.9). For describing this "adjusting" procedure, we first prove the following lemma.

Lemma 3.8. *For any $\boldsymbol{m} = (m_1, ..., m_k) \in \mathcal{M}^k$, let $S_{\boldsymbol{m}} \subseteq \mathsf{Func}(\mathcal{M}^{\le k}, \{0,1\})$ be the subset consisting of all H such that $H(m_1, ..., m_i) = 1$ for all $i \in [k]$. There exists a unitary $U_{\boldsymbol{m}}$ such that*

$$U_{\boldsymbol{m}} \sum_{H \in \mathsf{Func}(\mathcal{M}^{\le k}, \{0,1\})} \sqrt{D(H)}\,|H\rangle_{\mathbf{H}} = \sum_{H \in S_{\boldsymbol{m}}} \sqrt{\frac{D(H)}{\epsilon^k}}\,|H\rangle_{\mathbf{H}}\,.$$

Proof. Recall that H is encoded as a concatenation of $H(\boldsymbol{m}')$ for all $\boldsymbol{m}' \in \mathcal{M}^{\le k}$ and $\mathbf{H}_{\boldsymbol{m}'}$ denotes the register to store $H(\boldsymbol{m}')$. Then it is easy to see that we have

$$\sum_{H \in \mathsf{Func}(\mathcal{M}^{\le k}, \{0,1\})} \sqrt{D(H)}\,|H\rangle_{\mathbf{H}} = \bigotimes_{\boldsymbol{m}' \in \mathcal{M}^{\le k}} \left(\sqrt{1-\epsilon}\,|0\rangle_{\mathbf{H}_{\boldsymbol{m}'}} + \sqrt{\epsilon}\,|1\rangle_{\mathbf{H}_{\boldsymbol{m}'}}\right)$$

and

$$\sum_{H \in S_{\boldsymbol{m}}} \sqrt{\frac{D(H)}{\epsilon^k}}\,|H\rangle_{\mathbf{H}} = \left(\bigotimes_{\boldsymbol{m}' \in \mathsf{Prefix}_{\boldsymbol{m}}} |1\rangle_{\mathbf{H}_{\boldsymbol{m}'}}\right) \otimes \left(\bigotimes_{\boldsymbol{m}' \notin \mathsf{Prefix}_{\boldsymbol{m}}} \left(\sqrt{1-\epsilon}\,|0\rangle_{\mathbf{H}_{\boldsymbol{m}'}} + \sqrt{\epsilon}\,|1\rangle_{\mathbf{H}_{\boldsymbol{m}'}}\right)\right)$$

where $\mathsf{Prefix}_{\boldsymbol{m}} \subseteq \mathcal{M}^{\le k}$ is the set of all prefixes of $\boldsymbol{m}$, i.e., $\mathsf{Prefix}_{\boldsymbol{m}} = \{m_1, (m_1, m_2), ..., (m_1, ..., m_k)\}$. For each $\boldsymbol{m}' \in \mathcal{M}^{\le k}$, we define $U'_{\boldsymbol{m}'}$ as a unitary on $\mathbf{H}_{\boldsymbol{m}'}$ that satisfies

$$U'_{\boldsymbol{m}'}\,|1\rangle_{\mathbf{H}_{\boldsymbol{m}'}} = \sqrt{1-\epsilon}\,|0\rangle_{\mathbf{H}_{\boldsymbol{m}'}} + \sqrt{\epsilon}\,|1\rangle_{\mathbf{H}_{\boldsymbol{m}'}}\,.$$

We define $U_{\boldsymbol{m}}$ as

$$U_{\boldsymbol{m}} := \prod_{\boldsymbol{m}' \in \mathsf{Prefix}_{\boldsymbol{m}}} {U'_{\boldsymbol{m}'}}^{\dagger}.$$

Then the equation in Lemma 3.8 clearly holds. □

The formal description of $\widetilde{V}^*$ is given below.

$\widetilde{V}^*$ works over its internal register $\mathbf{V} = (\mathbf{X}, \mathbf{Aux} = (\mathbf{Cont}, \mathbf{R}, \mathbf{H}), \mathbf{W} = (\mathbf{Count}, \mathbf{M}_1, ..., \mathbf{M}_k, \mathbf{B}))$ and an additional message register $\mathbf{M}$ where $\mathbf{Cont}$ is a single-qubit register and all other registers are similar to those of V^* in Section 3.1 except that $\mathbf{Aux}$ contains an additional register $\mathbf{Cont}$. The output register is designated as $\mathbf{Out} := (\mathbf{Cont}, \mathbf{B})$. The unitary $\widetilde{U}^*$ for $\widetilde{V}^*$ is defined as follows

$$\begin{aligned}&\widetilde{U}^* \left(|0\rangle_{\mathbf{Cont}} |\mathsf{other}_0\rangle_{\mathbf{Other}} + |1\rangle_{\mathbf{Cont}} |\mathsf{other}_1\rangle_{\mathbf{Other}}\right)\\ =&|0\rangle_{\mathbf{Cont}} \left(U_{\mathsf{hon}} |\mathsf{other}_0\rangle_{\mathbf{Other}}\right) + |1\rangle_{\mathbf{Cont}} \left(U^* |\mathsf{other}_1\rangle_{\mathbf{Other}}\right),\end{aligned}$$

where $\mathbf{Other}$ denotes all registers except for $\mathbf{Cont}$, U^* is the unitary for V^* as defined in Section 3.1, and U_{hon} is the unitary that corresponds to the honest verifier with an additional "adjusting unitary" $U_{\boldsymbol{m}}$ on $\mathbf{H}$. Formally, U_{hon} is defined as follows.

Unitary U_{hon}: It non-trivially acts on registers $\mathbf{Count}$, $\mathbf{X}$, $(\mathbf{M}_1, ..., \mathbf{M}_k)$, $\mathbf{R}$, $\mathbf{H}$, $\mathbf{M}$, and $\mathbf{B}$:

- It reads the value j in $\mathbf{Count}$ and increments it to $i = j + 1 \bmod k$. It swaps $\mathbf{M}$ and $\mathbf{M}_i$ (in superposition).
- It does either of the following depending on the value in $\mathbf{Count}$ in superposition.
  1. If the value in $\mathbf{Count}$ is $i \neq 0$ (i.e., it is not in the final round), it applies the following unitary over $\mathbf{X}$, $\mathbf{M}_1, ..., \mathbf{M}_i$, $\mathbf{R}$, and $\mathbf{M}$:

$$|x, m_1, \cdots, m_i, r, m\rangle \to |x, m_1, \cdots, m_i, r, m \oplus F[x, r](m_1, \cdots, m_i)\rangle\,.$$

  2. If the value in $\mathbf{Count}$ is 0 (i.e., it is in the final round), it applies the following unitary over $\mathbf{X}$, $\mathbf{M}_1, ..., \mathbf{M}_k$, $\mathbf{R}$, and $\mathbf{B}$:

$$|x, m_1, \cdots, m_k, r, b\rangle \to |x, m_1, \cdots, m_k, r, b \oplus F[x, r](m_1, \cdots, m_k)\rangle\,.$$

     Then it applies the "adjusting unitary" over $\mathbf{M}_1, ..., \mathbf{M}_k$, and $\mathbf{H}$:

$$|m_1, \cdots, m_k, H\rangle \to U_{\boldsymbol{m}} |m_1, \cdots, m_k, H\rangle\,.$$

     where $\boldsymbol{m} := (m_1, ..., m_k)$. That is, it first puts $F[x, r](\boldsymbol{m})$ into $\mathbf{B}$, then adjusts $\mathbf{H}$ using the unitary $U_{\boldsymbol{m}}$.

Lemma 3.9. *For any $x \in L \cap \{0,1\}^\lambda$ and $w \in R_L(x)$, suppose that we run $\langle P(w), \widetilde{V}^*(|\widetilde{\psi}_\epsilon\rangle)\rangle(x)$ and measure $\mathbf{B}$ and the outcome is 1. Then the resulting state in $\mathbf{Cont}$ (tracing out other registers) is a pure state $|\phi_\epsilon\rangle_{\mathbf{Cont}} := \sqrt{\frac{1}{1+\epsilon^k}} |0\rangle_{\mathbf{Cont}} + \sqrt{\frac{\epsilon^k}{1+\epsilon^k}} |1\rangle_{\mathbf{Cont}}$.*

Proof. Let $|\eta\rangle$ be the final state of the internal register of $\widetilde{V}^*$ after executing $\langle P(w), \widetilde{V}^*(|\widetilde{\psi}_\epsilon\rangle)\rangle(x)$. For $\beta \in \{0,1\}$, let $|\eta_\beta\rangle$ be the final state of the internal register of $\widetilde{V}^*$ after executing $\langle P(w), \widetilde{V}^*(|\widetilde{\psi}_\epsilon^{(\beta)}\rangle)\rangle(x)$.

where $|\widetilde{\psi}_\epsilon^{(\beta)}\rangle_{\mathbf{Cont},\mathbf{R},\mathbf{H}} := |\beta\rangle_{\mathbf{Cont}} \otimes |\psi_\epsilon\rangle_{\mathbf{R},\mathbf{H}}$. Since $\widetilde{V}^*$ only uses **Cont** as a control register and $|\widetilde{\psi}_\epsilon\rangle_{\mathbf{Cont},\mathbf{R},\mathbf{H}} = \frac{1}{\sqrt{2}}\left(|\widetilde{\psi}_\epsilon^{(0)}\rangle_{\mathbf{Cont},\mathbf{R},\mathbf{H}} + |\widetilde{\psi}_\epsilon^{(1)}\rangle_{\mathbf{Cont},\mathbf{R},\mathbf{H}}\right)$, it is easy to see that we have

$$|\eta\rangle = \frac{1}{\sqrt{2}}\left(|\eta_0\rangle + |\eta_1\rangle\right).$$

In the following, when we consider summations over $\boldsymbol{m}$, r, and H, they are over all $\boldsymbol{m} \in \mathcal{M}^{\leq k}$, $r \in \mathcal{R}$, and $H \in \mathsf{Func}(\mathcal{M}^{\leq k}, \{0,1\})$, respectively, unless otherwise specified.

By the definition of $\widetilde{V}^*$ and the perfect completeness of the protocol Π, the value in **B** of $|\eta_0\rangle$ is always 1, and $\widetilde{V}^*(x; |\widetilde{\psi}_\epsilon^{(0)}\rangle)$ does not act on **H** except for the final application of $U_{\boldsymbol{m}}$. Therefore, by Lemma 3.8, we have

$$\begin{aligned}
|\eta_0\rangle &= U_{\boldsymbol{m}} |0\rangle_{\mathbf{Cont}} |x\rangle_{\mathbf{X}} |0\rangle_{\mathbf{Count}} \otimes |1\rangle_{\mathbf{B}} \otimes \sum_{\boldsymbol{m},r,H} \left(\sqrt{\frac{D(H)}{|\mathcal{R}|}} |r,H\rangle_{\mathbf{R},\mathbf{H}} \otimes \sqrt{p_{r,\boldsymbol{m}}} |\boldsymbol{m}\rangle_{\mathbf{M}_1,\ldots,\mathbf{M}_k}\right) \\
&= |0\rangle_{\mathbf{Cont}} |x\rangle_{\mathbf{X}} |0\rangle_{\mathbf{Count}} \otimes |1\rangle_{\mathbf{B}} \otimes \sum_{\boldsymbol{m},r,H\in S_{\boldsymbol{m}}} \left(\sqrt{\frac{D(H)}{\epsilon^k \cdot |\mathcal{R}|}} |r,H\rangle_{\mathbf{R},\mathbf{H}} \otimes \sqrt{p_{r,\boldsymbol{m}}} |\boldsymbol{m}\rangle_{\mathbf{M}_1,\ldots,\mathbf{M}_k}\right)
\end{aligned}$$

where $p_{r,\boldsymbol{m}}$ is the probability that the transcript is $\boldsymbol{m}$ in the honest execution conditioned on the fixed verifier's randomness r.

On the other hand, by the definition of $\widetilde{V}^*$, the value in **B** of $|\eta_1\rangle$ is 1 if and only if $H \in S_{\boldsymbol{m}}$ for the transcript $\boldsymbol{m}$. Therefore we have

$$|\eta_1\rangle = |1\rangle_{\mathbf{Cont}} |x\rangle_{\mathbf{X}} |0\rangle_{\mathbf{Count}} \otimes \begin{pmatrix} & |1\rangle_{\mathbf{B}} \otimes \sum_{\boldsymbol{m},r,H\in S_{\boldsymbol{m}}} \left(\sqrt{\frac{D(H)}{|\mathcal{R}|}} |r,H\rangle_{\mathbf{R},\mathbf{H}} \otimes \sqrt{p_{r,\boldsymbol{m}}} |\boldsymbol{m}\rangle_{\mathbf{M}_1,\ldots,\mathbf{M}_k}\right) \\ + & |0\rangle_{\mathbf{B}} \otimes |\mathsf{garbage}\rangle_{\mathbf{R},\mathbf{H},\mathbf{M}_1,\ldots,\mathbf{M}_k} \end{pmatrix}$$

for some (sub-normalized) state $|\mathsf{garbage}\rangle_{\mathbf{R},\mathbf{H},\mathbf{M}_1,\ldots,\mathbf{M}_k}$. Therefore, we have

$$\begin{aligned}
(|1\rangle\langle 1|)_{\mathbf{B}} |\eta\rangle = \frac{1}{\sqrt{2}} \left(\sqrt{\frac{1}{\epsilon^k}} |0\rangle_{\mathbf{Cont}} + |1\rangle_{\mathbf{Cont}}\right) \otimes |x\rangle_{\mathbf{X}} |0\rangle_{\mathbf{Count}} \otimes |1\rangle_{\mathbf{B}} \\
\otimes \sum_{\boldsymbol{m},r,H\in S_{\boldsymbol{m}}} \left(\sqrt{\frac{D(H)}{|\mathcal{R}|}} |r,H\rangle_{\mathbf{R},\mathbf{H}}\right).
\end{aligned}$$

Here we omit the identity operator on registers other than **B** and $(|1\rangle\langle 1|)_{\mathbf{B}}$ simply means the projection onto states whose values in **B** is 0.

Then Lemma 3.9 immediately follows by normalization.

□

Suppose that there is a quantum black-box simulator $\mathsf{Sim}_{\mathsf{exp}}$ (exp stands for 'expected') for the protocol Π whose expected number of queries is at most $q/2 = \mathsf{poly}(\lambda)$ that works for all possibly inefficient verifiers.[15] Especially, we assume that for any ϵ, we have

$$\{\mathsf{OUT}_{\widetilde{V}^*}\langle P(w), \widetilde{V}^*(x; |\widetilde{\psi}_\epsilon\rangle)\rangle(x)\}_{\lambda,x,w} \overset{comp}{\approx} \{\mathsf{OUT}_{\widetilde{V}^*}(\mathsf{Sim}_{\mathsf{exp}}^{\widetilde{V}^*(x;|\widetilde{\psi}_\epsilon\rangle)}(x))\}_{\lambda,x,w} \tag{4}$$

[15] We write exp in the subscript to differentiate this from strict-polynomial query simulators that appear in previous subsection. We take the expected number of queries to be $q/2$ instead of q just for convenience of the proof. Since q can be arbitrary polynomial, this does not lose generality.

where $\lambda \in \mathbb{N}$, $x \in L \cap \{0,1\}^\lambda$, $w \in R_L(\lambda)$. By Lemma 3.9, we can show that the final state in **Cont** after the execution $\mathsf{Sim}_{\mathsf{exp}}^{\widetilde{V}^*(x;|\widetilde{\psi}_\epsilon\rangle)}(x)$ conditioned on that $\widetilde{V}^*$ accepts (i.e., the value in **B** is 1) should be close to $|\phi_\epsilon\rangle$. (Remark that the probability that $\widetilde{V}^*$ accepts is larger than 1/2 since it always accepts if the value in **Cont** is 0.) In the following, we show stronger claims. Specifically, we show that

1. the probability that $\widetilde{V}^*$ accepts *and* the number of $\mathsf{Sim}_{\mathsf{exp}}$'s queries is at most q is at least $1/4 - \mathsf{negl}(\lambda)$ (Lemma 3.10), and
2. the final state in **Cont** after the execution $\mathsf{Sim}_{\mathsf{exp}}^{\widetilde{V}^*(x;|\widetilde{\psi}_\epsilon\rangle)}(x)$ conditioned on the above event is close to $|\phi_\epsilon\rangle$ (Lemma 3.11).

By combining Lemmas 3.10 and 3.11, we can show that the "truncated version" of $\mathsf{Sim}_{\mathsf{exp}}$ that makes at most q queries can let $V^*(x;|\psi_\epsilon\rangle)$ accept with probability at least $\frac{\epsilon^k}{4} - \mathsf{negl}(\lambda)$ (Lemma 3.12). By Lemma 3.2, this implies $L \in \mathbf{BQP}$, which completes the proof of Theorem 3.7. The details follow.

Similarly to Observation 1, because registers **Cont** is only used as a control qubit throughout the execution of $\mathsf{Sim}^{V^*(x;|\tilde{\psi}_\epsilon\rangle)}(x)$, we can trace out registers **Cont** while preserving the behavior of this simulator. We have the following observation:

Observation 4. *Let $M_{\mathbf{Cont}}$ be the measurement on the register* **Cont**. *For any (inefficient) black-box simulator* Sim *it has zero advantage of distinguishing if it has black-box access to $V^*(x;|\tilde{\psi}\rangle)$ or $V^*(x;M_{\mathbf{Cont}}|\tilde{\psi}\rangle)$.*

For any $x \in L \cap \{0,1\}^\lambda$, we consider an experiment $\mathsf{Exp}(x,\epsilon)$ where we run $\mathsf{Sim}_{\mathsf{exp}}^{\widetilde{V}^*(x;|\widetilde{\psi}_\epsilon\rangle)}(x)$ and then measure **B**. Let $\mathsf{Q}_{\leq q}$ be the event that the number of queries made by $\mathsf{Sim}_{\mathsf{exp}}$ is at most q and $\mathsf{E}_{\mathbf{B}=1}$ be the event that the measurement outcome of **B** is 1. Then we prove the following lemmas.

Lemma 3.10. *For any $x \in L \cap \{0,1\}^\lambda$ and $\epsilon \in [0,1]$, we have*

$$\Pr_{\mathsf{Exp}(x,\epsilon)}[\mathsf{Q}_{\leq q} \wedge \mathsf{E}_{\mathbf{B}=1}] \geq 1/4 - \mathsf{negl}(\lambda).$$

Proof. First, we recall that no black-box simulator can distinguish oracle access to $\widetilde{V}^*(x;|\widetilde{\psi}_\epsilon\rangle)$ and $\widetilde{V}^*(x;M_{\mathbf{Cont}}|\widetilde{\psi}_\epsilon\rangle)$ (see Observation 4). Let $\mathsf{Exp}_0(x,\epsilon)$ be the same as $\mathsf{Exp}(x,\epsilon)$ except that the auxiliary input of $\widetilde{V}^*$ is replaced with $|0\rangle_{\mathbf{Cont}}|\psi_\epsilon\rangle_{\mathbf{R},\mathbf{H}}$. Remark that $|0\rangle_{\mathbf{Cont}}|\psi_\epsilon\rangle_{\mathbf{R},\mathbf{H}}$ is the post-measurement state after measuring **Cont** of $|\widetilde{\psi}_\epsilon\rangle$ conditioned on that the measurement outcome is 0, which happens with probability 1/2. By the above observation, we have

$$\Pr_{\mathsf{Exp}(x,\epsilon)}[\mathsf{Q}_{\leq q} \wedge \mathsf{E}_{\mathbf{B}=1}] \geq \frac{1}{2}\Pr_{\mathsf{Exp}_0(x,\epsilon)}[\mathsf{Q}_{\leq q} \wedge \mathsf{E}_{\mathbf{B}=1}]. \tag{5}$$

Since the expected number of queries made by Sim is at most $q/2$ for any malicious verifier given as an oracle, by Markov's inequality, we have

$$\Pr_{\mathsf{Exp}_0(x,\epsilon)}[\mathsf{Q}_{\leq q}] \geq \frac{1}{2}. \tag{6}$$

When we run $\langle P(w), \widetilde{V}^*(|0\rangle_{\mathbf{Cont}}|\psi_\epsilon\rangle_{\mathbf{R},\mathbf{H}})\rangle(x)$ for some $w \in R_L(x)$ and then measure **B**, the measurement outcome of **B** is always 1 noting that $\widetilde{V}^*$ just runs the honest verifier followed by an

additional "adjusting unitary" $U_{\boldsymbol{m}}$, which does not affect the value in $\mathbf{B}$, when its auxliary input is $|0\rangle_{\mathbf{Cont}} |\psi_\epsilon\rangle_{\mathbf{R},\mathbf{H}}$. Therefore, by our assumption that $\mathsf{Sim}_{\mathsf{exp}}$ is a simulator for the protocol Π, we must have

$$\Pr_{\mathsf{Exp}_0(x,\epsilon)}[\mathsf{E}_{\mathbf{B}=1}] = 1 - \mathsf{negl}(\lambda). \tag{7}$$

By combining Eq. 5, 6, and 7, we obtain Lemma 3.10:

$$\begin{aligned}\Pr_{\mathsf{Exp}(x,\epsilon)}[\mathsf{Q}_{\leq q} \wedge \mathsf{E}_{\mathbf{B}=1}] \geq& \frac{1}{2} \Pr_{\mathsf{Exp}_0(x,\epsilon)}[\mathsf{Q}_{\leq q} \wedge \mathsf{E}_{\mathbf{B}=1}] \\ \geq& \frac{1}{2}\left(\Pr_{\mathsf{Exp}_0(x,\epsilon)}[\mathsf{Q}_{\leq q}] - \Pr_{\mathsf{Exp}_0(x,\epsilon)}[\neg\mathsf{E}_{\mathbf{B}=1}]\right) \\ \geq& \frac{1}{4} - \mathsf{negl}(\lambda).\end{aligned}$$

□

Lemma 3.11. *For any $x \in L \cap \{0,1\}^\lambda$ and ϵ, let $\sigma_{x,\epsilon}$ be the state in* $\mathbf{Cont}$ *(tracing out other registers) after executing $\mathsf{Exp}(x,\epsilon)$ conditioned on $\mathsf{Q}_{\leq q} \wedge \mathsf{E}_{\mathbf{B}=1}$. Then we have*

$$\mathsf{TD}(\sigma_{x,\epsilon}, |\phi_\epsilon\rangle\langle\phi_\epsilon|) = \mathsf{negl}(\lambda)$$

where $|\phi_\epsilon\rangle$ is as defined in Lemma 3.9.

Proof. Let $w \in R_L(x)$ be an arbitrary witness for x. Let $\rho^{\mathsf{real}}_{x,w,\epsilon} := \mathsf{OUT}_{\widetilde{V}^*}\langle P(w), \widetilde{V}^*(|\widetilde{\psi}_\epsilon\rangle)\rangle(x)$ and $\rho^{\mathsf{sim}}_{x,\epsilon} := \mathsf{OUT}_{\widetilde{V}^*}\mathsf{Sim}_{\mathsf{exp}}^{\widetilde{V}^*(x;|\widetilde{\psi}_\epsilon\rangle)}(x)$. We note that they are states over $\mathbf{Out} = (\mathbf{Cont}, \mathbf{B})$. We consider the following distinguisher $\mathcal{D}$ that tries to distinguish $\rho^{\mathsf{real}}_{x,w,\epsilon}$ and $\rho^{\mathsf{sim}}_{x,\epsilon}$.

$\mathcal{D}(\rho)$**:** It measures the register $\mathbf{B}$. If the measurement outcome is 0, then it outputs 0. Otherwise, it generates a state $|\phi_\epsilon\rangle_{\mathbf{Cont}'} := \sqrt{\frac{1}{1+\epsilon^k}}|0\rangle_{\mathbf{Cont}'} + \sqrt{\frac{\epsilon^k}{1+\epsilon^k}}|1\rangle_{\mathbf{Cont}'}$ in a new register $\mathbf{Cont}'$, runs the SWAP test (Lemma 2.6) between registers $\mathbf{Cont}$ and $\mathbf{Cont}'$, and outputs 0 if the SWAP test accepts and 1 otherwise.

$\mathcal{D}$ will output 1 if and only if $\mathbf{B}$ is measured as 1, and the SWAP test rejects. By Lemmas 2.6 and 3.9, we have

$$\Pr[\mathcal{D}(\rho^{\mathsf{real}}_{x,w,\epsilon}) = 1] = 0.$$

Since $\rho^{\mathsf{real}}_{x,w,\epsilon}$ and $\rho^{\mathsf{sim}}_{x,\epsilon}$ are computationally indistinguishable by Eq. 4, the above equation implies

$$\Pr[\mathcal{D}(\rho^{\mathsf{sim}}_{x,\epsilon}) = 1] = \mathsf{negl}(\lambda). \tag{8}$$

On the other hand, by Lemma 2.6, we have

$$\Pr[\mathcal{D}(\rho^{\mathsf{sim}}_{x,\epsilon}) = 1] \geq \Pr_{\mathsf{Exp}(x,\epsilon)}[\mathsf{Q}_{\leq q} \wedge \mathsf{E}_{\mathbf{B}=1}] \frac{1 - \langle\phi_\epsilon| \sigma_{x,\epsilon} |\phi_\epsilon\rangle}{2}.$$

since the r.h.s. is the probability that $\mathcal{D}(\rho^{\mathsf{sim}}_{x,\epsilon})$ returns 1 *and* $\mathsf{Sim}_{\mathsf{exp}}$ made at most q queries when generating $\rho^{\mathsf{sim}}_{x,\epsilon}$. By Lemma 3.10, $\Pr_{\mathsf{Exp}(x,\epsilon)}[\mathsf{Q}_{\leq q} \wedge \mathsf{E}_{\mathbf{B}=1}] \geq 1/4 - \mathsf{negl}(\lambda)$. Therefore, for satisfying Eq. 8, we must have $\langle\phi_\epsilon| \sigma_{x,\epsilon} |\phi_\epsilon\rangle = 1 - \mathsf{negl}(\lambda)$, which implies $\mathsf{TD}(\sigma_{x,\epsilon}, |\phi_\epsilon\rangle\langle\phi_\epsilon|) = \mathsf{negl}(\lambda)$. □

Lemma 3.12. *Let* Sim *be the "truncated version" of* $\mathsf{Sim}_{\mathsf{exp}}$ *that works similarly to* $\mathsf{Sim}_{\mathsf{exp}}$ *except that it immediately halts when* $\mathsf{Sim}_{\mathsf{exp}}$ *tries to make* $(q+1)$*-th query. Then for any* $x \in L \cap \{0,1\}^\lambda$ *and* ϵ*, we have*

$$\Pr\left[M_{\mathbf{B}} \circ \mathsf{OUT}_{V^*}\left(\mathsf{Sim}^{V^*(x;|\psi_\epsilon\rangle)}(x)\right) = 1\right] \geq \frac{\epsilon^k}{4} - \mathsf{negl}(\lambda).$$

Remark that in the above lemma, Sim is given the oracle V^*, which is the random-aborting verifier defined in Section 3.1, rather than $\widetilde{V}^*$

Proof of Lemma 3.12. Let $\sigma_{x,\epsilon}$ be as in Lemma 3.11. By $\mathsf{TD}(\sigma_{x,\epsilon}, |\phi_\epsilon\rangle\langle\phi_\epsilon|) = \mathsf{negl}(\lambda)$ as shown in Lemma 3.11, if we measure **Cont** of $\sigma_{x,\epsilon}$, then the outcome is 1 with probability $\frac{\epsilon^k}{1+\epsilon^k} \pm \mathsf{negl}(\lambda)$. Combining this with Lemma 3.10, we have

$$\Pr_{\mathsf{Exp}'(x,\epsilon)}[\mathsf{Q}_{\leq q} \wedge \mathsf{E}_{\mathbf{B}=1} \wedge \mathsf{E}_{\mathbf{Cont}=1}] \geq \frac{\epsilon^k}{4(1+\epsilon^k)} - \mathsf{negl}(\lambda) \geq \frac{\epsilon^k}{8} - \mathsf{negl}(\lambda) \tag{9}$$

where $\mathsf{Exp}'(x,\epsilon)$ is the same as $\mathsf{Exp}(x,\epsilon)$ except that **Cont** is also measured at the end, and $\mathsf{E}_{\mathbf{Cont}=1}$ is the event that the measurement outcome of **Cont** is 1. Since Sim works similarly to $\mathsf{Sim}_{\mathsf{exp}}$ when $\mathsf{Q}_{\leq q}$ occurs, we have

$$\Pr\left[M_{\mathbf{B}} \circ \mathsf{OUT}_{V^*}\left(\mathsf{Sim}^{V^*(x;|\psi_\epsilon\rangle)}(x)\right) = 1\right] \geq \Pr[M_{\mathbf{B}} \circ \mathsf{Sim}_{\mathsf{exp}}^{V^*(x;|\psi_\epsilon\rangle)}(x) = 1 \wedge \mathsf{Q}_{\leq q}].$$

By Observation 4, there is no difference if we measure **Cont** at the beginning of the experiment instead of at the end, and when the measurement outcome of **Cont** is 1, $\widetilde{V}^*$ with auxiliary input $|\widetilde{\psi}_\epsilon\rangle$ works similarly to V^* with auxiliary input $|\psi_\epsilon\rangle$. Therefore, we have

$$\begin{aligned}
\Pr[M_{\mathbf{B}} \circ \mathsf{Sim}_{\mathsf{exp}}^{V^*(x;|\psi_\epsilon\rangle)}(x) = 1 \wedge \mathsf{Q}_{\leq q}] &= \Pr_{\mathsf{Exp}'(x,\epsilon)}[\mathsf{Q}_{\leq q} \wedge \mathsf{E}_{\mathbf{B}=1} | \mathsf{E}_{\mathbf{Cont}=1}] \\
&= \frac{\Pr_{\mathsf{Exp}'(x,\epsilon)}[\mathsf{Q}_{\leq q} \wedge \mathsf{E}_{\mathbf{B}=1} \wedge \mathsf{E}_{\mathbf{Cont}=1}]}{\Pr_{\mathsf{Exp}'(x,\epsilon)}[\mathsf{E}_{\mathbf{Cont}=1}]} \\
&\geq \frac{\epsilon^k}{4} - \mathsf{negl}(\lambda)
\end{aligned}$$

where the last inequality follows from Eq. 9 and $\Pr_{\mathsf{Exp}'(x,\epsilon)}[\mathsf{E}_{\mathbf{Cont}=1}] = 1/2$, which is easy to see. □

Finally, we prove Theorem 3.7 based on Lemma 3.12.

Proof of Theorem 3.7. Let ϵ_q^* be as in Lemma 3.2 and $\epsilon := \epsilon_q^*$. Since Sim makes at most q queries, Lemma 3.2 and 3.12 immediately imply $L \in \mathbf{BQP}$. □

3.3 Expected-Polynomial-Time Simulation for Efficient Verifier

In this section, we extend the proof of Theorem 3.7 in Section 3.2 to prove Theorem 3.1. That is, we prove that black-box simulation is impossible even for *QPT* malicious verifiers.

The reason why the malicious verifier $\widetilde{V}^*$ in Section 3.2 is inefficient is that it has to apply the unitary $U_{\boldsymbol{m}}$ given in Lemma 3.8, which works over an exponential-qubit register $\mathbf{H}$. Therefore, if we have an analogue of Lemma 3.8 for a family of efficiently computable functions that is indistinguishable from a random function taken from $\mathcal{H}_\epsilon$ by at most Q quantum queries. We prove such a lemma in the following.

Lemma 3.13. *Let $\epsilon \in [0,1]$ be a rational number expressed as $\epsilon = \frac{B}{A}$ for some $A, B \in \mathbb{N}$ such that $\log A = \mathsf{poly}(\lambda)$ and $\log B = \mathsf{poly}(\lambda)$.*[16] *For any $Q = \mathsf{poly}(\lambda)$, there exists a family $\widetilde{\mathcal{H}}_\epsilon = \{\widetilde{H}_\kappa : \mathcal{M}^{\le k} \to \{0,1\}\}_{\kappa \in \mathcal{K}}$ of classical polynomial-time computable functions that satisfies the following properties.*

1. *For any algorithm $\mathcal{A}$ that makes at most Q quantum queries and any quantum input ρ, we have*
$$\Pr_{H \xleftarrow{\$} \mathcal{H}_\epsilon}\left[\mathcal{A}^H(\rho) = 1\right] = \Pr_{\kappa \xleftarrow{\$} \mathcal{K}}\left[\mathcal{A}^{\widetilde{H}_\kappa}(\rho) = 1\right].$$
2. *For any $\boldsymbol{m} = (m_1, ..., m_k) \in \mathcal{M}^k$, let $S_{\boldsymbol{m}} \subseteq \mathcal{K}$ be the subset consisting of all κ such that $\widetilde{H}_\kappa(m_1, ..., m_i) = 1$ for all $i \in [k]$. There exists a unitary $U^{(Q)}_{\boldsymbol{m}}$ such that*
$$U^{(Q)}_{\boldsymbol{m}} \sqrt{\frac{1}{|\mathcal{K}|}} \sum_{\kappa \in \mathcal{K}} |\kappa\rangle = \sqrt{\frac{1}{|S_{\boldsymbol{m}}|}} \sum_{\kappa \in S_{\boldsymbol{m}}} |\kappa\rangle.$$
$U_{\boldsymbol{m}}$ can be implemented by a quantum circuit of size $\mathsf{poly}(\lambda)$.

Proof. Let $\mathcal{H}'_{2Q,A} = \{H'_{\kappa'} : \mathcal{M}^{\le k} \to [A]\}_{\kappa' \in \mathcal{K}'}$ be a $2Q$-wise independent hash family from $\mathcal{M}^{\le k}$ to $[A]$. By Lemma 2.4, for any $\mathcal{A}$ that makes at most Q quantum queries and any quantum input ρ, we have
$$\Pr_{H' \xleftarrow{\$} \mathsf{Func}(\mathcal{M}^{\le k}, [A])}\left[\mathcal{A}^{H'}(\rho) = 1\right] = \Pr_{\kappa' \xleftarrow{\$} \mathcal{K}'}\left[\mathcal{A}^{H'_{\kappa'}}(\rho) = 1\right]. \tag{10}$$
We define $\widetilde{\mathcal{H}}_\epsilon = \{\widetilde{H}_\kappa : \mathcal{M}^{\le k} \to \{0,1\}\}_{\kappa \in \mathcal{K}}$ as follows.

- $\mathcal{K} := \mathcal{K}' \times [A]^k$. In other words, a key κ' for $\mathcal{H}'_{2Q,A}$ is sampled, and k additional random additive terms $\{a_i\}$ for each input length is sampled, as explained below.
- For $\kappa = (\kappa', \{a_i\}_{i \in [K]})$ and $(m_1, ..., m_i) \in \mathcal{M}^{\le k}$, we define
$$\widetilde{H}_\kappa(m_1, ..., m_i) := \begin{cases} 1 & \text{if } (H'_{\kappa'}(m_1, ..., m_i) + a_i \mod A) \le B \\ 0 & \text{otherwise} \end{cases}.$$

In the above definition, if we use a uniformly random function $H' \xleftarrow{\$} \mathsf{Func}(\mathcal{M}^{\le k}, [A])$ instead of $H'_{\kappa'}$, $\widetilde{H}_\kappa$ is distributed according to $\mathcal{H}_\epsilon$. Therefore, Eq. 10 implies the first item of Lemma 3.13.

For proving the second item, we consider unitaries $U_{\mathsf{add},\boldsymbol{m},i}$, $U_{\le A}$, and $U_{\le B}$ that satisfy the following.

- For any $\boldsymbol{m} = (m_1, ..., m_k) \in \mathcal{M}^k$, $i \in [k]$, and $(\kappa', a_1, ..., a_k) \in \mathcal{K}$, we have
$$U_{\mathsf{add},\boldsymbol{m},i} |\kappa', a_1, ..., a_k\rangle = |\kappa', a_1, ..., a_{i-1}, (H'_{\kappa'}(m_1, ..., m_i) + a_i \mod A), a_{i+1}, ..., a_k\rangle.$$
- For any $\kappa' \in \mathcal{K}'$, we have
$$U_{\le A} |\kappa', 0, ..., 0\rangle = \sqrt{\frac{1}{A^k}} \sum_{(a_1, ..., a_k) \in [A]^k} |\kappa', a_1, ..., a_k\rangle.$$

[16] Note that ϵ is also a function of λ, but we omit to explicitly write the dependence on ϵ for simplicity.

- For any $\kappa' \in \mathcal{K}'$, we have

$$U_{\le B}\left|\kappa', 0, ..., 0\right\rangle = \sqrt{\frac{1}{B^k}} \sum_{(a_1,...,a_k)\in[B]^k} \left|\kappa', a_1, ..., a_k\right\rangle.$$

We can see that such unitaries exist and are implementable by polynomial-size quantum circuits.

For any $\boldsymbol{m} = (m_1, ..., m_k) \in \mathcal{M}^{\le k}$, we define $U_{\boldsymbol{m}}$ as

$$U_{\boldsymbol{m}}^{(Q)} := \left(U_{\le A} U_{\le B}^{\dagger} \prod_{i=1}^{k} U_{\mathsf{add},\boldsymbol{m},i}\right)^{\dagger}.$$

Then $U_{\boldsymbol{m}}^{(Q)}$ is implementable by a polynomial-size quantum circuit. For any $\kappa' \in \mathcal{K}'$ and $\boldsymbol{m} = (m_1, ..., m_k)$, we let $T_{\kappa',\boldsymbol{m}} \subseteq [A]^k$ be the subset consisting of all $(a_1, ..., a_k)$ such that we have $(H'_{\kappa'}(m_1, ..., m_i) + a_i \mod A) \le B$ for all $i \in [k]$. Then we have

$$\begin{aligned}
U_{\boldsymbol{m}}^{(Q)\dagger} \sqrt{\frac{1}{|S_{\boldsymbol{m}}|}} \sum_{\kappa\in S_{\boldsymbol{m}}} |\kappa\rangle &= U_{\boldsymbol{m}}^{(Q)\dagger} \sqrt{\frac{1}{|\mathcal{K}'|\cdot B^k}} \sum_{\kappa'\in\mathcal{K}'} |\kappa'\rangle \sum_{(a_1,...,a_i)\in T_{\kappa',\boldsymbol{m}}} |a_1, ..., a_i\rangle \\
&= U_{\le A} U_{\le B}^{\dagger} \sqrt{\frac{1}{|\mathcal{K}'|\cdot B^k}} \sum_{\kappa'\in\mathcal{K}'} |\kappa'\rangle \sum_{(a_1,...,a_i)\in[B]^k} |a_1, ..., a_i\rangle \\
&= U_{\le A} \sqrt{\frac{1}{|\mathcal{K}'|}} \sum_{\kappa'\in\mathcal{K}'} |\kappa'\rangle |0, ..., 0\rangle \\
&= \sqrt{\frac{1}{|\mathcal{K}'|\cdot A^k}} \sum_{\kappa'\in\mathcal{K}'} |\kappa'\rangle \sum_{(a_1,...,a_i)\in[A]^k} |a_1, ..., a_i\rangle \\
&= \sqrt{\frac{1}{|\mathcal{K}|}} \sum_{\kappa\in\mathcal{K}} |\kappa\rangle.
\end{aligned}$$

By applying $U_{\boldsymbol{m}}^{(Q)}$ on both sides, the second item of Lemma 3.13 follows. □

With Lemma 3.13 in hand, we can prove Theorem 3.1 similarly to the proof of Theorem 3.7 except that we consider the efficient version of $\widetilde{V^*}$ using Lemma 3.13. Here, we remark that we only need to take sufficiently small yet noticeable ϵ (depending on q) in the proof of Theorem 3.7, and thus we can assume that ϵ satisfies the assumption of Lemma 3.13 without loss of generality. Specifically, we modify the auxiliary input to

$$|\widetilde{\psi_\epsilon^{(q)}}\rangle_{\mathbf{Cont},\mathbf{R},\mathbf{H}} := \frac{1}{\sqrt{2}}(|0\rangle_{\mathbf{Cont}} + |1\rangle_{\mathbf{Cont}}) \otimes |\psi_\epsilon^{(q)}\rangle_{\mathbf{R},\mathbf{H}}$$

where

$$|\psi_\epsilon^{(q)}\rangle_{\mathbf{R},\mathbf{H}} := \sum_{r\in\mathcal{R},\kappa\in\mathcal{K}} \sqrt{\frac{1}{|\mathcal{R}|\cdot|\mathcal{K}|}} |r, \kappa\rangle_{\mathbf{R},\mathbf{H}}$$

and modify $\widetilde{V^*}$ to apply H_κ whenever it applies H and apply $U_{\boldsymbol{m}}^{(Q)}$ instead of $U_{\boldsymbol{m}}$ for the case of $\mathbf{Cont} = 0$ and $\mathbf{Count} = k - 1$ (i.e., in the final round) where $Q := 2kq$ In this setting, we can

prove an analogue of Lemma 3.9 by using the second item of Lemma 3.13 instead of Lemma 3.8. We can prove analogues of Lemma 3.10, 3.11, and 3.12 in the exactly the same way to the original ones since these proofs do not use anything on the state in $\mathbf{H}$. Finally, when we can prove Theorem 3.1 by Lemma 3.2 and the analogue of Lemma 3.12 noting that Sim that makes at most q queries to the verifier can be seen as an algorithm that makes at most $Q = 2kq$ quantum queries to H by Observations 2 and 3, and thus it cannot distinguish H_κ for $\kappa \xleftarrow{\$} \mathcal{K}$ and $H \xleftarrow{\$} \mathcal{H}_\epsilon$ by the first item of Lemma 3.13.

Malicious verifier with fixed polynomial-time. In the proof of Theorem 3.1, we consider a malicious verifier whose running time depends on q, which is twice of the simulator's expected number of queries. Though this is sufficient for proving the impossibility of quantum black-box simulation (since Definition 2.2 requires a simulator to work for all QPT verifiers whose running time may be an arbitrarily large polynomial), one may think that it is "unfair" that the running time of the malicious verifier is larger than that of the simulator. We can resolve this issue if we use a quantumly-accessible PRF, which exists under the existence of post-quantum one-way functions [Zha12a].[17] Specifically, if we use a quantumly-accessible PRF instead of $4q$-wise independent function in Lemma 3.13, we can prove a similar lemma with a unitary whose size does not depend on q instead of $U_{\boldsymbol{m}}^{(Q)}$, which naturally yields a malicious verifier whose running time does not depend on q.

4 Impossibility of BB ϵ-ZK for Constant-Round Public-Coin Arguments

In this section, we prove the following theorem.

Theorem 4.1. *If there exists a constant-round public-coin post-quantum black-box ϵ-zero-knowledge argument for a language L, then $L \in \mathbf{BQP}$.*

First, we define notations that are used throughout this section. Let $\Pi = (P, V)$ be a classical constant-round public-coin interactive argument for a language L. Without loss of generality, we assume that P sends the first message, and let (P, V) be $(2k-1)$-round protocol where P sends $k = O(1)$ messages in the protocol and V sends $k-1$ messages (which are public-coin). We use m_i to mean P's i-th message and c_i to mean V's i-th message. We assume that all messages sent from P are elements of a classical set $\mathcal{M}$ and V's messages are uniformly chosen from a classical set $\mathcal{C}$. Without loss of generality, we assume that $\mathcal{C}$ is a subset of $\mathcal{M}$. (If it is not the case, we can simply augment $\mathcal{M}$ to include $\mathcal{C}$.) For any $\boldsymbol{c} = (c_1, ..., c_{k-1}) \in \mathcal{C}^{k-1}$, we denote by $\mathsf{Acc}[x, \boldsymbol{c}] \subseteq \mathcal{M}^k$ the subset consisting of all $(m_1, ..., m_k) \in \mathcal{M}^k$ such that $(m_1, c_1, ..., m_{k-1}, c_{k-1}, m_k)$ is an accepting transcript. (Note that this is well-defined as we assume that Π is public-coin). For any q, let $\mathcal{H}_{4(k-1)q}$ be a family of $4(k-1)q$-wise independent hash functions from $\mathcal{M}^{\leq(k-1)}$ to $\mathcal{C}$.

A high level structure of the rest of this section is similar to Section 3.1. To prove Theorem 4.1, we consider a malicious verifier that derives its messages by applying a random function on the transcript so far. More precisely, for any $q = \mathsf{poly}(\lambda)$, we consider a malicious verifier V^* with an auxiliary input $|\psi_q\rangle_{\mathbf{H}}$ as follows.

$|\psi_q\rangle_{\mathbf{H}}$ is the uniform superposition over $H \in \mathcal{H}_{4(k-1)q}$.

[17] A similar idea is used to resolve a similar issue in the context of the impossibility of strict-polynomial-time simulation in the classical setting [BL02].

V^* works over its internal register $(\mathbf{X}, \mathbf{Aux} = \mathbf{H}, \mathbf{W} = (\mathbf{Count}, \mathbf{M}_1, ..., \mathbf{M}_k, \mathbf{B}))$ and an additional message register $\mathbf{M}$. We define the output register as $\mathbf{Out} := \mathbf{B}$. It works as follows where $\mathbf{Count}$ stores a non-negative integer smaller than k (i.e. $\{0, 1, \cdots, k-1\}$), each register of $\mathbf{M}_1, ..., \mathbf{M}_k$ and $\mathbf{M}$ stores an element of $\mathcal{M}$ and $\mathbf{B}$ stores a single bit. $\mathbf{M}$ is the register to store messages from/to external prover, and $\mathbf{M}_i$ is a register to record the i-th message from the prover.

We next explain the unitary U^* for V^*.

1. V^* takes inputs a statement x and a quantum auxiliary input $|\psi\rangle$: $\mathbf{X}$ is initialized to be $|x\rangle_{\mathbf{X}}$ where x is the statement to be proven, $\mathbf{Aux}$ is initialized to be $|\psi\rangle_{\mathbf{H}}$, and all other registers are initialized to be 0.
2. *Verifier V^* on round $< k$*: Upon receiving the i-th message from P for $i < k$ in $\mathbf{M}$, swap $\mathbf{M}$ and $\mathbf{M}_i$ and increment the value in $\mathbf{Count}$. We note that V^* can know i since it keeps track of which round it is playing by the value in $\mathbf{Count}$. Let $(m_1, ..., m_i)$ be the messages sent from P so far. Then it returns $H(m_1, ..., m_i) \in \mathcal{C}$ to P as its next message.

   Unitary U^* on $\mathbf{Count} < k-1$: It acts on registers $\mathbf{Count}$ (whose value is less than $k-1$), $\mathbf{X}, (\mathbf{M}_1, ..., \mathbf{M}_k)$, $\mathbf{H}$ and $\mathbf{M}$:

   - It reads the value j in $\mathbf{Count}$ and increments it to $i := j + 1$. It swaps $\mathbf{M}$ and $\mathbf{M}_i$ (in superposition).
   - It applies the following unitary.[18]

$$|x, m_1, \cdots, m_i, H, m\rangle \to |x, m_1, \cdots, m_i, H, m + H(m_1, \cdots, m_i)\rangle .$$

3. *Verifier V^* on round k*: Upon receiving the k-th message from P in $\mathbf{M}$, swap $\mathbf{M}$ and $\mathbf{M}_k$ and increment the value in $\mathbf{Count}$. Then flip the bit in $\mathbf{B}$ if $(m_1, ..., m_k) \in \mathsf{Acc}[x, (c_1, ..., c_{k-1})]$ where $c_i := H(m_1, ..., m_i)$ for all $i \in [k-1]$ and $(m_1, ..., m_k)$ and H are values in registers $(\mathbf{M}_1, ..., \mathbf{M}_k)$ and $\mathbf{H}$.

   Unitary U^* on $\mathbf{Count} = k-1$: It acts on registers $\mathbf{Count}$ (whose value is exactly equal to $k-1$), $\mathbf{X}, (\mathbf{M}_1, ..., \mathbf{M}_k)$, $\mathbf{H}$ and $\mathbf{B}$:

   - It reads the value $i = k-1$ in $\mathbf{Count}$ and sets it to 0. It swaps $\mathbf{M}$ and $\mathbf{M}_k$ (in superposition).
   - Let $x, (m_1, \cdots, m_k), H$ and b be the values in registers $\mathbf{X}, (\mathbf{M}_1, \cdots, \mathbf{M}_k), \mathbf{H}$ and $\mathbf{B}$. Let $F^*[x, H]$ be the following function: on input $(m_1, \cdots, m_k) \in \mathcal{M}^k$,

$$F^*[x, H](m_1, ..., m_k) := \begin{cases} 1 & \text{if } (m_1, ..., m_k) \in \mathsf{Acc}[x, (c_1, ..., c_{k-1})] \\ 0 & \text{otherwise} \end{cases}$$

     where $c_i := H(m_1, ..., m_i)$ for $i \in [k-1]$.
     It applies the function in superposition.

$$|x, m_1, \cdots, m_k, r, H, b\rangle \to |x, m_1, \cdots, m_k, r, H, b + F^*[x, H](m_1, \cdots, m_k)\rangle .$$

With the description of V^* above, we have the following observation.

[18] Here, $H(m_1, \cdots, m_i)$ is seen as an element of $\mathcal{M}$ by using our assumption $\mathcal{C} \subseteq \mathcal{M}$.

Observation 5. *Let $M_{\mathbf{H}}$ be the measurement on the register $\mathbf{H}$. For any (inefficient) black-box simulator Sim it has zero advantage of distinguishing if it has black-box access to $V^*(x;|\psi\rangle_{\mathbf{H}})$ or $V^*(x;M_{\mathbf{H}}\circ|\psi\rangle_{\mathbf{H}})$.*

Observation 5 says that even for an unbounded simulator with black-box access to $V^*(x;|\psi_q\rangle_{\mathbf{H}})$, it has no way to tell if the auxiliary input $|\psi_q\rangle_{\mathbf{H}}$ gets measured at the beginning or never gets measured. This is because the register $\mathbf{H}$ is only used as control qubits throughout the execution of $\mathsf{Sim}^{V^*(x;|\psi_q\rangle)}(x)$, we can trace out the register $\mathbf{H}$ while preserving the behavior of this simulator.

Observation 6. *An oracle that applies U^* can be simulated by $2(k-1)$ quantum oracle access to H.*

This can be easily seen from the description of U^* above. Note that we require $2(k-1)$ queries instead of $k-1$ queries since we need to compute $k-1$ values of H to compute $F^*[x,r,H]$ (when the input is in $\mathcal{M}^k$) and then need to uncompute them.

We then prove the following lemma.

Lemma 4.2. *If there exists a quantum black-box simulator Sim that makes at most $q=\mathsf{poly}(\lambda)$ queries such that*

$$\Pr\left[M_{\mathbf{B}}\circ\mathsf{OUT}_{V^*}\left(\mathsf{Sim}^{V^*(x;|\psi_q\rangle)}(x)\right)=1\right]\geq\frac{1}{\mathsf{poly}(\lambda)}$$

for all $x\in L\cap\{0,1\}^\lambda$ where $M_{\mathbf{B}}$ means measuring and outputting the register $M_{\mathbf{B}}$, then we have $L\in\mathbf{BQP}$.

The above lemma immediately implies Theorem 4.1.

Proof of Theorem 4.1. Let $\Pi=(P,V)$ be a constant-round public-coin post-quantum black-box ϵ-zero-knowledge argument for a language L where P sends $k=O(1)$ messages and V^* and $|\psi_q\rangle$ are defined above. By definition, for any noticeable ϵ, there exists a quantum black-box simulator Sim for Π that makes at most $q=\mathsf{poly}(\lambda)$ queries such that

$$\{\mathsf{OUT}_{V^*}\langle P(w),V^*(|\psi_q\rangle)\rangle(x)\}_{\lambda,x,w}\overset{comp}{\approx}_\epsilon\{\mathsf{OUT}_{V^*}(\mathsf{Sim}^{V^*(x;|\psi_q\rangle)}(x))\}_{\lambda,x,w}$$

where $\lambda\in\mathbb{N}$, $x\in L\cap\{0,1\}^\lambda$, and $w\in R_L(\lambda)$. (Note that we can assume that the number of queries by the simulator is strict-polynomial as explained in Remark 1.) Especially, we take $\epsilon:=1/2$. By perfect completeness of Π and the definitions of V^* and $|\psi_q\rangle$, for any $x\in L\cap\{0,1\}^\lambda$ and its witness $w\in R_L(x)$, we have

$$\Pr[M_{\mathbf{B}}\circ\mathsf{OUT}_{V^*}(\langle P(w),V^*(|\psi_q\rangle)\rangle(x))=1]=1.$$

By combining the above, we have

$$\Pr\left[M_{\mathbf{B}}\circ\mathsf{OUT}_{V^*}\left(\mathsf{Sim}^{V^*(x;|\psi_q\rangle)}(x)\right)=1\right]=1-\epsilon=\frac{1}{2}>\frac{1}{\mathsf{poly}(\lambda)}.$$

By Lemma 4.2, this implies $L\in\mathbf{BQP}$. □

Remark 5. *As one can see from the above proof, we can actually prove a stronger statement than Theorem 4.1. That is, even a black-box simulation with approximation error as large as $1-\frac{1}{\mathsf{poly}(\lambda)}$ is still impossible for a language outside $\mathbf{BQP}$.*

Then we prove Lemma 4.2.

Proof of Lemma 4.2. By Observation 5, we can assume $|\psi_q\rangle$ is measured at the beginning. In other words, the auxiliary state is sampled as $H \xleftarrow{\$} \mathcal{H}_{4(k-1)q}$. Once H is fixed, the unitary U^* (corresponding to V^*) and its inverse can be simulated by $2(k-1)$ quantum access to a classical function H as observed in Observation 6. Moreover, since the simulator makes at most q queries, by Lemma 2.4, the simulator's behavior does not change even if H is uniformly sampled from $\mathsf{Func}(\mathcal{M}^{\leq(k-1)}, \mathcal{C})$.

We let $\mathsf{Acc}^*[x,H] \subseteq \mathcal{M}^k$ be the set of $\boldsymbol{m} = (m_1, ..., m_k)$ such that $(m_1, ..., m_k) \in \mathsf{Acc}[x, (c_1, ..., c_{k-1})]$ where $c_i := H(m_1, ..., m_i)$ for $i \in [k-1]$. After the execution of $\mathsf{Sim}^{V^*(x;|H\rangle)}(x)$, $\mathbf{B}$ contains 1 if and only if $(\mathbf{M}_1, ..., \mathbf{M}_k)$ contains an element in $\mathsf{Acc}^*[x,H]$. Therefore, for proving Lemma 4.2, it suffices to prove the following lemma.

Lemma 4.3. *If there exists an oracle-aided quantum algorithm $\mathcal{S}$ that makes at most $\mathsf{poly}(\lambda)$ quantum queries such that*

$$\Pr_{H \xleftarrow{\$} \mathsf{Func}(\mathcal{M}^{\leq(k-1)}, \mathcal{C})} \left[\mathcal{S}^H(x) \in \mathsf{Acc}^*[x,H]\right] \geq \frac{1}{\mathsf{poly}(\lambda)}$$

for all $x \in L \cap \{0,1\}^\lambda$, then we have $L \in \mathbf{BQP}$.

We prove the above lemma below. Assuming Lemma 4.3, we show Lemma 4.2 holds. Let Sim be a quantum black-box simulator that makes at most q quantum queries, such that

$$\Pr\left[M_{\mathbf{B}} \circ \mathsf{OUT}_{V^*}\left(\mathsf{Sim}^{V^*(x;|\psi_q\rangle)}(x)\right) = 1\right] \geq \frac{1}{\mathsf{poly}(\lambda)}.$$

By Observation 5 and Lemma 2.4, we have

$$\begin{aligned}&\Pr_{H \xleftarrow{\$} \mathsf{Func}(\mathcal{M}^{\leq(k-1)}, \mathcal{C})} \left[M_{\mathbf{B}} \circ \mathsf{OUT}_{V^*}\left(\mathsf{Sim}^{V^*(x;|H\rangle)}(x)\right) = 1\right] \\ =&\Pr\left[M_{\mathbf{B}} \circ \mathsf{OUT}_{V^*}\left(\mathsf{Sim}^{V^*(x;|\psi_q\rangle)}(x)\right) = 1\right] \geq \frac{1}{\mathsf{poly}(\lambda)}.\end{aligned}$$

Finally, we note that $M_{\mathbf{B}} \circ \mathsf{OUT}_{V^*}\left(\mathsf{Sim}^{V^*(x;|r,H\rangle)}(x)\right)$ can be computed by only having black-box access to H (by Observation 6). It outputs 1 (the register $\mathbf{B}$ is 1) if and only if the values $(m_1, \cdots, m_k)$ in $\mathbf{M}_1, \cdots, \mathbf{M}_k$ are in $\mathsf{Acc}^*[x,H]$. Thus, there is an algorithm $\mathcal{S}$ that computes $M_{\mathbf{B}} \circ \mathsf{Sim}^{V^*(x;|H\rangle)}(x)$ and measures registers $\mathbf{M}_1, \cdots, \mathbf{M}_k$. Such an algorithm $\mathcal{S}$ satisfies the requirement in Lemma 4.3. Therefore L is in $\mathbf{BQP}$. □

The remaining part is to prove Lemma 4.3.

Proof of Lemma 4.3. As shown in [DFM20], Fiat-Shamir transform for constant-round public-coin protocol preserves the soundness up to a polynomial security loss in the quantum random oracle model where an adversary may quantumly query the random oracle. Let Π_{ni} be the non-interactive protocol that is obtained by applying Fiat-Shamir transform to Π. More concretely, Π_{ni} is a non-interactive argument in the random oracle model where a proof is of the form $(m_1, ..., m_k)$ and it is accepted for a statement x if and only if $(m_1, ..., m_k) \in \mathsf{Acc}^*[x,H]$. Since we assume that Π has a negligible soundness error, so does Π_{ni} by the result of [DFM20].[19] This means that we have

$$\Pr_{H \xleftarrow{\$} \mathsf{Func}(\mathcal{M}^{\leq(k-1)}, \mathcal{C})} \left[\mathcal{S}^H(x) \in \mathsf{Acc}^*[x,H]\right] \leq \mathsf{negl}(\lambda)$$

[19] Though the way of applying the Fiat-Shamir is slightly different from that in [DFM20] (where they derive a challenge by nested applications of the random oracle), their proof still works for the above way as noted in [DFM20, Remark 12]

for all $x \in \{0,1\}^\lambda \setminus L$ since otherwise we can use $\mathcal{S}$ to break the soundness of Π_{ni}. Therefore, we can use $\mathcal{S}$ to decide L by simulating a random function H by itself, which implies $L \in \mathsf{BQP}$. (Note that one can efficiently simulate a random function for quantum-query adversaries that makes at most $2(k-1)q$ queries by using $4(k-1)q$-wise independent hash function by Lemma 2.4.) □

5 Impossibility of BB ϵ-ZK for Three-Round Arguments

In this section, we prove the following theorem.

Theorem 5.1. *If there exists a three-round post-quantum black-box ϵ-zero-knowledge argument for a language L, then $L \in \mathbf{BQP}$.*

First, we define notations that are used throughout this section. Let $\Pi = (P, V)$ be a classical three-round interactive argument for a language L. We assume that all messages sent between P and V are elements of a classical set $\mathcal{M}$ (e.g., we can take $\mathcal{M} := \{0,1\}^\ell$ for sufficiently large ℓ). Let $\mathcal{R}$ be V's randomness space. For any fixed statement x and randomness $r \in \mathcal{R}$, V's message in the second round can be seen as a deterministic function of the prover's first message m_1. We denote this function by $F[x,r] : \mathcal{M} \to \mathcal{M}$. We denote by $\mathsf{Acc}[x,r] \subseteq \mathcal{M}^2$ the subset consisting of all (m_1, m_2) such that the verifier accepts when its randomness is r and prover's messages in the first and third rounds are m_1 and m_2, respectively. For any q, let $\mathcal{H}_{4q}$ be a family of $4q$-wise independent hash functions from $\mathcal{M}$ to $\mathcal{R}$.

A high level structure of the rest of this section is similar to Section 3.1. To prove Theorem 5.1, we consider a malicious verifier that derives its randomness by applying a random function on the prover's first message. More precisely, for any $q = \mathsf{poly}(\lambda)$, we consider a malicious verifier V^* with an auxiliary input $|\psi_q\rangle_{\mathbf{H}}$ as follows.

$|\psi_q\rangle_{\mathbf{H}}$ is the uniform superposition over $H \in \mathcal{H}_{4q}$.

V^* works over its internal register $(\mathbf{X}, \mathbf{Aux} = \mathbf{H}, \mathbf{W} = (\mathbf{Count}, \mathbf{M}_1, \mathbf{M}_2, \mathbf{B}))$ and an additional message register $\mathbf{M}$. We define the output register as $\mathbf{Out} := \mathbf{B}$. It works as follows where $\mathbf{Count}$ stores an integer 0 or 1,[20] each register of $\mathbf{M}_1, \mathbf{M}_2$, and $\mathbf{M}$ stores an element of $\mathcal{M}$ and $\mathbf{B}$ stores a single bit. $\mathbf{M}$ is the register to store messages from/to a external prover, and $\mathbf{M}_1$ and $\mathbf{M}_2$ are registers to record the prover's first and second messages, respectively.[21]

We next explain the unitary U^* for V^*. The interaction between V^* and the honest prover P has been formally defined in Section 2.1. We recall it here.

1. V^* takes inputs a statement x and a quantum auxiliary input $|\psi_q\rangle$: $\mathbf{X}$ is initialized to be $|x\rangle_{\mathbf{X}}$ where x is the statement to be proven, $\mathbf{Aux}$ is initialized to be $|\psi_q\rangle_{\mathbf{H}}$, and all other registers are initialized to be 0.
2. *Verifier V^* on the first message*: Upon receiving the first message from P in $\mathbf{M}$, swap $\mathbf{M}$ and $\mathbf{M}_1$ and increment the value in $\mathbf{Count}$[22]. We note that V^* can know i since it keeps track of which round it is playing by the value in $\mathbf{Count}$. Let m_1 be the first message sent from P. Then return $F[x, H(m_1)](m_1)$ to P.

   Unitary U^* on $\mathbf{Count} = 0$: It acts on registers $\mathbf{Count}$ (whose value is exactly equal to 0), $\mathbf{X}$, $(\mathbf{M}_1, \mathbf{M}_2)$, $\mathbf{H}$ and $\mathbf{M}$:

[20] We view them as an integer rather than a bit for the consistency to the description of V^* in Section 3.1.
[21] Remark that the prover's second message is a message sent in the third round.
[22] This is a NOT gate that maps $|i\rangle$ to $|(i+1) \bmod 2\rangle$.

- It reads the value $i = 0$ in **Count** and increments it to 1. It swaps $\mathbf{M}$ and $\mathbf{M}_1$ (in superposition).
- Let x, m_1, r, H and m be the values in registers $\mathbf{X}, \mathbf{M}_1, \mathbf{H}$ and $\mathbf{M}$. Let $F^*[x, H]$ be the following function: on input $m_1 \in \mathcal{M}$,

$$F^*[x, H](m_1) := F[x, H(m_1)](m_1).$$

It then applies the function in superposition:

$$|x, m_1, H, m\rangle \to |x, m_1, H, m + F^*[x, H](m_1)\rangle .$$

3. *Verifier V^* on the second message*: Upon receiving the second message from P in $\mathbf{M}$, swap $\mathbf{M}$ and $\mathbf{M}_2$ and increment the value in **Count**. Then flip the bit in $\mathbf{B}$ if $(m_1, m_2) \in \mathsf{Acc}[x, H(m_1)]$ where (m_1, m_2) and H are values in registers $(\mathbf{M}_1, \mathbf{M}_2)$, and $\mathbf{H}$.

Unitary U^* on **Count** $= 1$: It acts on registers **Count** (whose value is exactly equal to 1), $\mathbf{X}, (\mathbf{M}_1, \mathbf{M}_2)$, $\mathbf{H}$ and $\mathbf{B}$:

- It reads the value $i = 1$ in **Count** and sets it to 0. It swaps $\mathbf{M}$ and $\mathbf{M}_2$ (in superposition).
- Let $x, (m_1, m_2), H$ and b be the values in registers $\mathbf{X}, (\mathbf{M}_1, \mathbf{M}_2), \mathbf{H}$ and $\mathbf{B}$. Let $F_{\mathsf{Acc}^*[x,H]}$ be the following function: on input $(m_1, m_2) \in \mathcal{M}^2$,

$$F_{\mathsf{Acc}^*[x,H]}(m_1, m_2) := \begin{cases} 1 & \text{if } (m_1, m_2) \in \mathsf{Acc}^*[x, H] \\ 0 & \text{otherwise} \end{cases}$$

where $\mathsf{Acc}^*[x, H] \subseteq \mathcal{M}^2$ is the set of all (m_1, m_2) such that $(m_1, m_2) \in \mathsf{Acc}[x, H(m_1)]$. It applies the function in superposition.

$$|x, m_1, m_2, H, b\rangle \to |x, m_1, m_2, H, b + F_{\mathsf{Acc}^*[x,H]}(m_1, m_2)\rangle .$$

With the description of V^* above, we have the following observation.

Observation 7. *Let $M_{\mathbf{H}}$ be the measurement on register $\mathbf{H}$. For any (inefficient) black-box simulator Sim it has zero advantage of distinguishing if it has black-box access to $V^*(x; |\psi\rangle_{\mathbf{H}})$ or $V^*(x; M_{\mathbf{H}} \circ |\psi\rangle_{\mathbf{H}})$.*

Observation 7 says that even for an unbounded simulator with black-box access to $V^*(x; |\psi\rangle_{\mathbf{H}})$, it has no way to tell if the auxiliary input $|\psi\rangle_{\mathbf{H}}$ gets measured at the beginning or never gets measured. This is because the register $\mathbf{H}$ is only used as control qubits throughout the execution of $\mathsf{Sim}^{V^*(x;|\psi\rangle)}(x)$, we can trace out the register $\mathbf{H}$ while preserving the behavior of this simulator.

Observation 8. *V^* can be simulated giving oracle access to $F^*[x, H]$ and $F_{\mathsf{Acc}^*[x,H]}$.*

This can be easily seen from the description of U^* above.

Observation 9. *Given x and r, a quantum oracle that computes $F^*[x, H]$ or $F_{\mathsf{Acc}^*[x,H]}$ can be simulated by 2 quantum oracle access to H.*

This can be easily seen from the definitions of $F^*[x, H]$ and $F_{\mathsf{Acc}^*[x,H]}$. Note that we require 2 queries instead of 1 query since we need to compute the value of H to compute $F^*[x, H]$ or $F_{\mathsf{Acc}^*[x,H]}$ and then need to uncompute it.

We then prove the following lemma.

Lemma 5.2. *If there exists a quantum black-box simulator* Sim *that makes at most* $q = \mathsf{poly}(\lambda)$ *queries such that*

$$\Pr\left[M_{\mathbf{B}} \circ \mathsf{OUT}_{V^*}\left(\mathsf{Sim}^{V^*(x;|\psi_q\rangle)}(x)\right) = 1\right] \geq \frac{1}{\mathsf{poly}(\lambda)}$$

for all $x \in L \cap \{0,1\}^\lambda$ *where* $M_{\mathbf{B}}$ *means measuring and outputting the register* $M_{\mathbf{B}}$*, then we have* $L \in \mathbf{BQP}$.

The above lemma immediately implies Theorem 5.1.

Proof of Theorem 5.1. This is exactly the same as the proof of Theorem 4.1 based on Lemma 4.2 Let $\Pi = (P, V)$ be a three-round post-quantum black-box ϵ-zero-knowledge argument for a language L and V^* and $|\psi_q\rangle$ are defined above. By definition, for any noticeable ϵ, there exists a quantum black-box simulator Sim for Π that makes at most $q = \mathsf{poly}(\lambda)$ queries such that

$$\{\mathsf{OUT}_{V^*}\langle P(w), V^*(|\psi_q\rangle)\rangle(x)\}_{\lambda,x,w} \overset{comp}{\approx}_\epsilon \{\mathsf{OUT}_{V^*}(\mathsf{Sim}^{V^*(x;|\psi_q\rangle)}(x))\}_{\lambda,x,w}$$

where $\lambda \in \mathbb{N}$, $x \in L \cap \{0,1\}^\lambda$, and $w \in R_L(\lambda)$. (Note that we can assume that the number of queries by the simulator is strict-polynomial as explained in Remark 1.) Especially, we take $\epsilon := 1/2$. By perfect completeness of Π and the definitions of V^* and $|\psi_q\rangle$, for any $x \in L \cap \{0,1\}^\lambda$ and its witness $w \in R_L(x)$, we have

$$\Pr[M_{\mathbf{B}} \circ \mathsf{OUT}_{V^*}(\langle P(w), V^*(|\psi_q\rangle)\rangle(x)) = 1] = 1.$$

By combining the above, we have

$$\Pr\left[M_{\mathbf{B}} \circ \mathsf{OUT}_{V^*}\left(\mathsf{Sim}^{V^*(x;|\psi_q\rangle)}(x)\right) = 1\right] = 1 - \epsilon = \frac{1}{2} > \frac{1}{\mathsf{poly}(\lambda)}.$$

By Lemma 5.2, this implies $L \in \mathbf{BQP}$. □

Remark 6. *As one can see from the above proof, we can actually prove a stronger statement than Theorem 5.1. That is, even a black-box simulation with approximation error as large as* $1 - \frac{1}{\mathsf{poly}(\lambda)}$ *is still impossible for a language outside* $\mathbf{BQP}$.

Then we prove Lemma 5.2.

Proof of Lemma 5.2. By Observation 7, we can assume $|\psi_q\rangle$ is measured at the beginning. In other words, the auxiliary state is sampled as $|H\rangle_{\mathbf{H}}$ for $H \xleftarrow{\$} \mathcal{H}_{4q}$. Once H is fixed, the unitary U^* (corresponding to V^*) and its inverse can be simulated by a single quantum access to a classical function $F^*[x, H]$ or $F_{\mathsf{Acc}^*[x,H]}$ defined in the description of V^* (Observation 8). Moreover, since the simulator makes at most q queries to the verifier and a single query can be simulated by two queries to H as observed in Observation 9, the simulator can be seen as an oracle-aided algorithm that makes at most $2q$ quantum queries to H. Therefore, by Lemma 2.4, the simulator's behavior does not change even if H is uniformly sampled from $\mathsf{Func}(\mathcal{M}, \mathcal{R})$. After the execution of $\mathsf{Sim}^{V^*(x;|H\rangle)}(x)$, $\mathbf{B}$ contains 1 if and only if $(\mathbf{M}_1, \mathbf{M}_2)$ contains an element in $\mathsf{Acc}^*[x, H]$.

Therefore, for proving Lemma 5.2, it suffices to prove the following lemma.

Lemma 5.3. *If there exists an oracle-aided quantum algorithm* $\mathcal{S}$ *that makes at most* $\mathsf{poly}(\lambda)$ *quantum queries such that*

$$\Pr_{H \xleftarrow{\$} \mathsf{Func}(\mathcal{M},\mathcal{R})}\left[\mathcal{S}^{F^*[x,H], F_{\mathsf{Acc}^*[x,H]}}(x) \in \mathsf{Acc}^*[x, H]\right] \geq \frac{1}{\mathsf{poly}(\lambda)}$$

for all $x \in L \cap \{0,1\}^\lambda$*, then we have* $L \in \mathbf{BQP}$.

We prove the above lemma below. Assuming Lemma 5.3, we show Lemma 5.2 holds. We note that this proof is similar to the proof of Lemma 4.2 based on Lemma 4.3.

Let Sim be a quantum black-box simulator that makes at most q quantum queries, such that

$$\Pr\left[M_{\mathbf{B}} \circ \mathsf{OUT}_{V^*}\left(\mathsf{Sim}^{V^*(x;|\psi_q\rangle)}(x)\right) = 1\right] \geq \frac{1}{\mathsf{poly}(\lambda)}.$$

By Observation 7 and Lemma 2.4, we have

$$\begin{aligned}&\Pr_{H \xleftarrow{\$} \mathsf{Func}(\mathcal{M},\mathcal{R})}\left[M_{\mathbf{B}} \circ \mathsf{OUT}_{V^*}\left(\mathsf{Sim}^{V^*(x;|H\rangle)}(x)\right) = 1\right]\\ =&\Pr\left[M_{\mathbf{B}} \circ \mathsf{OUT}_{V^*}\left(\mathsf{Sim}^{V^*(x;|\psi_q\rangle)}(x)\right) = 1\right] \geq \frac{1}{\mathsf{poly}(\lambda)}.\end{aligned}$$

Finally, we note that $M_{\mathbf{B}} \circ \mathsf{OUT}_{V^*}\left(\mathsf{Sim}^{V^*(x;|r,H\rangle)}(x)\right)$ can be computed by only having black-box access to $F^*[x,H]$ and $F_{\mathsf{Acc}^*[x,H]}$ (by Observation 8). It outputs 1 (the register $\mathbf{B}$ is 1) if and only if the values (m_1, m_2) in $\mathbf{M}_1, \mathbf{M}_2$ are in $\mathsf{Acc}^*[x,H]$. Thus, there is an algorithm $\mathcal{S}$ that computes $M_{\mathbf{B}} \circ \mathsf{Sim}^{V^*(x;|H\rangle)}(x)$ and measures registers $\mathbf{M}_1, \mathbf{M}_2$. Such an algorithm $\mathcal{S}$ satisfies the requirement in Lemma 5.3. Therefore L is in $\mathbf{BQP}$. □

For proving Lemma 5.3 we reduce it to a simplified lemma (Lemma 5.4 below) where $\mathcal{S}$ is not given the oracle $F_{\mathsf{Acc}^*[x,H]}$.

Lemma 5.4. *If there exists an oracle-aided quantum algorithm $\mathcal{S}$ that makes at most $\mathsf{poly}(\lambda)$ quantum queries such that we have*

$$\Pr_{H \xleftarrow{\$} \mathsf{Func}(\mathcal{M},\mathcal{R})}\left[\mathcal{S}^{F^*[x,H]}(x) \in \mathsf{Acc}^*[x,H]\right] \geq \frac{1}{\mathsf{poly}(\lambda)}$$

for all $x \in L \cap \{0,1\}^\lambda$, then we have $L \in \mathbf{BQP}$.

We first prove Lemma 5.3 assuming Lemma 5.4 by using Corollary 2.8.

Proof of Lemma 5.3. Let $\mathcal{S}$ be an algorithm that satisfies the assumption of Lemma 5.3. We apply Corollary 2.8 by considering $\mathcal{S}$ as $\mathcal{A}$ in Corollary 2.8. Then we can see that the algorithm corresponding to $\mathcal{C}$ in Corollary 2.8 satisfies the assumption of Lemma 5.4, which implies $L \in \mathsf{BQP}$. (Note that though $\mathcal{S}$ has an additional oracle $F^*[x,H]$, Corollary 2.8 is still applicable by considering an augmented algorithm $\mathcal{S}'$ that takes H as part of its input and simulates $F^*[x,H]$ by itself.) □

The remaining part is to prove Lemma 5.4.

Proof of Lemma 5.4. In the following, we simply write r and H in subscripts of probabilities to mean $r \xleftarrow{\$} \mathcal{R}$ and $H \xleftarrow{\$} \mathsf{Func}(\mathcal{M},\mathcal{R})$ for notational simplicity.

We apply Lemma 2.10 to $\mathcal{S}^{F^*[x,H]}$ where $k := 1$, $\mathcal{X} := \mathcal{M}$, $\mathcal{Y} := \mathcal{M}$, $\mathcal{Z} := \mathcal{M}$, and we define a relation $R \subseteq \mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$ by $(m_1, m_V, m_2) \in R$ if and only if $(m_1, m_V, m_2) \in \mathsf{Acc}'[x]$ where $\mathsf{Acc}'[x]$ is the set of all accepting transcripts (i.e., $(m_1, m_V, m_2) \in \mathsf{Acc}'[x]$ if and only if there exists $r \in \mathcal{R}$ such that $F[x,r](m_1) = m_V$ and $(m_1, m_2) \in \mathsf{Acc}[x,r]$). Note that $\mathsf{Acc}'[x]$ is not necessarily efficiently recognizable. By Lemma 2.10, for any x, H, r, and m_1^* (where we set $m_V := F[x,r](m_1^*)$), we have

$$\begin{aligned}&\Pr\left[m_1' = m_1^* \wedge (m_1', F[x,r](m_1^*), m_2) \in \mathsf{Acc}'[x] : (m_1', m_2) \xleftarrow{\$} \widetilde{\mathcal{S}}[F^*[x,H], F[x,r](m_1^*)](x)\right]\\ \geq&\frac{1}{(2q+1)^2}\Pr\left[m_1 = m_1^* \wedge (m_1, F[x,r](m_1^*), m_2) \in \mathsf{Acc}'[x] : (m_1, m_2) \xleftarrow{\$} \mathcal{S}^{F^*[x,H]_{m_1^*, F[x,r](m_1^*)}}(x)\right]\end{aligned} \tag{11}$$

where $\widetilde{\mathcal{S}}[F^*[x,H],F[x,r](m_1^*)]$ and $F^*[x,H]_{m_1^*,F[x,r](m_1^*)}$ are as defined in Lemma 2.10. Note that $F^*[x,H]_{m_1^*,F[x,r](m_1^*)} \equiv F^*[x,H_{m_1^*,r}]$ and $H_{m_1^*,r}$ is uniformly distributed over $\mathsf{Func}(\mathcal{M},\mathcal{R})$ if H and r are randomly chosen for any fixed m_1^*. Therefore, by taking the average over all H and r for Eq. 11, for any fixed m_1^* we have

$$\begin{aligned}&\Pr_{H,r}\left[m_1'=m_1^*\wedge(m_1',F[x,r](m_1^*),m_2)\in\mathsf{Acc}'[x]:(m_1',m_2)\xleftarrow{\$}\widetilde{\mathcal{S}}[F^*[x,H],F[x,r](m_1^*)](x)\right]\\&\geq\frac{1}{(2q+1)^2}\Pr_{H}\left[m_1=m_1^*\wedge(m_1,F[x,H(m_1^*)](m_1^*),m_2)\in\mathsf{Acc}'[x]:(m_1,m_2)\xleftarrow{\$}\mathcal{S}^{F^*[x,H]}(x)\right]\end{aligned}\tag{12}$$

When $m_1'=m_1^*$ and $m_1=m_1^*$, $(m_1',F[x,r](m_1^*),m_2)\in\mathsf{Acc}'[x]$ and $(m_1,F[x,H(m_1^*)](m_1^*),m_2)\in\mathsf{Acc}'[x]$ are equivalent to $(m_1',m_2)\in\mathsf{Acc}[x,r]$ and $(m_1,m_2)\in\mathsf{Acc}^*[x,H]$, respectively. Therefore, by taking a summation over all m_1^* for Eq. 12, we have

$$\begin{aligned}&\sum_{m_1^*\in\mathcal{M}}\Pr_{H,r}\left[m_1'=m_1^*\wedge(m_1',m_2)\in\mathsf{Acc}[x,r]:(m_1',m_2)\xleftarrow{\$}\widetilde{\mathcal{S}}[F^*[x,H],F[x,r](m_1^*)](x)\right]\\&\geq\frac{1}{(2q+1)^2}\Pr_{H}\left[\mathcal{S}^{F^*[x,H]}(x)\in\mathsf{Acc}^*[x,H]\right]\geq\frac{1}{\mathsf{poly}(\lambda)}\end{aligned}\tag{13}$$

for all $x\in L\cap\{0,1\}^\lambda$ where the last inequality follows from the assumption of Lemma 5.4.

For any H and r, we consider an algorithm $\mathcal{B}[H,r](x)$ that "imitates" the LHS of Eq. 13. Specifically, $\mathcal{B}[H,r](x)$ works as follows:

$\mathcal{B}[H,r](x)$**:** It works as follows.

1. Pick $(j^*,b^*)\xleftarrow{\$}([q]\times\{0,1\})\cup\{(\bot,\bot)\}$.
2. Run $\mathcal{S}$ where its oracle is simulated by $\mathcal{O}$ that is initialized to be $F^*[x,H]$. When $\mathcal{S}$ makes its j-th query to $\mathcal{O}$,
   (a) If $j=j^*$, measure $\mathcal{S}$'s query register to obtain m_1'.
      i. If $b^*=0$, reprogram $\mathcal{O}\leftarrow\mathsf{Reprogram}(\mathcal{O},m_1',F[x,r](m_1'))$ and answer $\mathcal{S}$'s j-th query by using the reprogrammed oracle.
      ii. If $b^*=1$, answer $\mathcal{S}$'s j-th query by using the oracle before the reprogramming and then reprogram $\mathcal{O}\leftarrow\mathsf{Reprogram}(\mathcal{O},m_1',F[x,r](m_1'))$.
   (b) Otherwise, answer $\mathcal{S}$'s j-th query by just using the oracle $\mathcal{O}$.
3. Let (m_1,m_2) be $\mathcal{S}$'s output. If $j^*=\bot$ (in which case m_1' has not been defined), set $m_1':=m_1$. Output (m_1',m_2).

Then we prove the following claims.

Claim 5.5. *For any $x\in L\cap\{0,1\}^\lambda$, we have*

$$\Pr_{H,r}\left[\mathcal{B}[H,r](x)\in\mathsf{Acc}[x,r]\right]\geq\frac{1}{\mathsf{poly}(\lambda)}.$$

Proof of Claim 5.5. By definition, we can see that $\mathcal{B}[H,r]$ works similarly to $\widetilde{\mathcal{S}}[F^*[x,H],F[x,r](m_1^*)](x)$ conditioned on that the measured query m_1' is equal to m^*. Therefore we have

$$\begin{aligned}&\Pr_{H,r}\left[m_1'=m_1^*\wedge(m_1',m_2)\in\mathsf{Acc}[x,r]:(m_1',m_2)\xleftarrow{\$}\mathcal{B}[H,r](x)\right]\\&=\Pr_{H,r}\left[m_1'=m_1^*\wedge(m_1',m_2)\in\mathsf{Acc}[x,r]:(m_1',m_2)\xleftarrow{\$}\widetilde{\mathcal{S}}[F^*[x,H],F[x,r](m_1^*)](x)\right]\end{aligned}$$

By substituting this for the LHS of Eq. 13, Claim 5.5 follows. □

Claim 5.6. *For any $x \in \{0,1\}^\lambda \setminus L$, we have*

$$\Pr_{H,r}\left[\mathcal{B}[H,r](x) \in \mathsf{Acc}[x,r]\right] \leq \mathsf{negl}(\lambda).$$

Proof of Claim 5.6. We construct a cheating prover P^* against the protocol Π that wins with probability $\Pr_{H,r}\left[\mathcal{B}[H,r](x) \in \mathsf{Acc}[x,r]\right]$, which immediately implies Claim 5.6 by the soundness of Π. Intuitively, $P^*(x)$ just runs $\mathcal{B}[H,r](x)$ where H is chosen by itself and r is chosen by the external verifier. Though P^* does not know r, it can simulate $\mathcal{B}[H,r](x)$ because it needs r only when responding to the measured query, and P^* can then send such a (classical) query to the external verifier to get the response.

Formally, P^* is described as follows. We will mark the difference between P^* and $\mathcal{B}[H,r]$ (for $H \xleftarrow{\$} \mathcal{H}_{4q}$ and $r \xleftarrow{\$} \mathcal{R}$) with underline.

$P^*(x)$**:** The cheating prover P^* interacts with the external verifier as follows:

1. Choose a function $H \xleftarrow{\$} \mathcal{H}_{4q}$ where $\mathcal{H}_{4q}$ is a family of $4q$-wise independent hash function, and initialize an oracle $\mathcal{O}$ to be a (quantumly-accessible) oracle that computes $F^*[x,H]$.
2. Pick $(j^*, b^*) \xleftarrow{\$} ([q] \times \{0,1\}) \cup \{(\bot,\bot)\}$.
3. Run $\mathcal{S}$ where its oracle is simulated by $\mathcal{O}$. When $\mathcal{S}$ makes its j-th query to $\mathcal{O}$,
   (a) If $j = j^*$, measure $\mathcal{S}$'s query register to obtain m_1'. Send m_1' to the external verifier as the first message, and receives the response m_V.
      i. If $b^* = 0$, reprogram $\mathcal{O} \leftarrow \mathsf{Reprogram}(\mathcal{O}, m_1', m_V)$ and answer $\mathcal{S}$'s j-th query by using the reprogrammed oracle.
      ii. If $b^* = 1$, answer $\mathcal{S}$'s j-th query by using the oracle before the reprogramming and then reprogram $\mathcal{O} \leftarrow \mathsf{Reprogram}(\mathcal{O}, m_1', m_V)$.
   (b) Otherwise, answer $\mathcal{S}$'s j-th query by just using the oracle $\mathcal{O}$.
4. Let (m_1, m_2) be $\mathcal{S}$'s output. If $j^* = \bot$ (in which case P^* has not sent the first message to the external verifier yet), complete the protocol by sending m_1 and m_2 as first and second messages to the external verifier (regardless of the verifier's response in the second round). Otherwise, P^* should have already run the protocol until the second round, so it completes the protocol by sending m_2 to the external verifier as the prover's second message.

By definitions, we can see that P^* perfectly simulates $\mathcal{B}[H,r]$ for $H \xleftarrow{\$} \mathcal{H}_{4q}$ and $r \xleftarrow{\$} \mathcal{R}$ where r is chosen by the external verifier. Moreover, P^* wins (i.e., the verifier accepts) if and only if the output of $\mathcal{B}[H,r]$ is in $\mathsf{Acc}[x,r]$. Moreover, by Lemma 2.4 and that we can simulate $\mathcal{B}[H,r](x)$ by at most $2q$ oracle access to H, the probability that $\mathcal{B}[H,r](x) \in \mathsf{Acc}[x,r]$ does not change if we choose a completely random function H instead of one from $\mathcal{H}_{4q}$. Therefore, the soundness of the protocol ensures $\Pr_{H,r}[\mathcal{B}[H,r](x) \in \mathsf{Acc}[x,r]] \leq \mathsf{negl}(\lambda)$. This completes the proof of Claim 5.6. □

Finally, we conclude the proof of Lemma 5.4 by using Claim 5.5 and 5.6. Since $\mathcal{B}[H,r](x)$ can be seen as an oracle-aided algorithm that makes at most $2q$ queries to H, we have

$$\Pr_{H \xleftarrow{\$} \mathcal{H}_{4q}, r}\left[\mathcal{B}[H,r](x) \in \mathsf{Acc}[x,r]\right] = \Pr_{H,r}\left[\mathcal{B}[H,r](x) \in \mathsf{Acc}[x,r]\right]$$

by Lemma 2.4. Then we can decide if a given element x is in L by running $\mathcal{B}[H,r](x)$ for $H \xleftarrow{\$} \mathcal{H}_{4q}$ and $r \xleftarrow{\$} \mathcal{R}$ and seeing if the output is in $\mathsf{Acc}[x,r]$. This means $L \in \mathbf{BQP}$. This completes the proof of Lemma 5.4. □

References

[AHU19] Andris Ambainis, Mike Hamburg, and Dominique Unruh. Quantum security proofs using semi-classical oracles. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 269–295. Springer, Heidelberg, August 2019.

[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, Oct 1997.

[BCY91] Gilles Brassard, Claude Crépeau, and Moti Yung. Constant-round perfect zero-knowledge computationally convincing protocols. *Theoretical Computer Science*, 84(1):23–52, 1991.

[BJY97] Mihir Bellare, Markus Jakobsson, and Moti Yung. Round-optimal zero-knowledge arguments based on any one-way function. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 280–305. Springer, Heidelberg, May 1997.

[BL02] Boaz Barak and Yehuda Lindell. Strict polynomial-time in simulation and extraction. In *34th ACM STOC*, pages 484–493. ACM Press, May 2002.

[Blu86] Manuel Blum. How to prove a theorem so no one else can claim it. In *Proceedings of the International Congress of Mathematicians*, page 1444–1451, 1986.

[BS20] Nir Bitansky and Omri Shmueli. Post-quantum zero knowledge in constant rounds. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *52nd ACM STOC*, pages 269–279. ACM Press, June 2020.

[CCY20a] Nai-Hui Chia, Kai-Min Chung, and Takashi Yamakawa. A black-box approach to post-quantum zero-knowledge in constant rounds. Cryptology ePrint Archive, Report 2020/1384, 2020. https://eprint.iacr.org/2020/1384.

[CCY20b] Nai-Hui Chia, Kai-Min Chung, and Takashi Yamakawa. Classical verification of quantum computations with efficient verifier. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part III*, volume 12552 of *LNCS*, pages 181–206. Springer, Heidelberg, November 2020.

[DFM20] Jelle Don, Serge Fehr, and Christian Majenz. The measure-and-reprogram technique 2.0: Multi-round fiat-shamir and more. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 602–631. Springer, Heidelberg, August 2020.

[DFMS19] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Security of the Fiat-Shamir transformation in the quantum random-oracle model. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 356–383. Springer, Heidelberg, August 2019.

[FGJ18] Nils Fleischhacker, Vipul Goyal, and Abhishek Jain. On the existence of three round zero-knowledge proofs. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 3–33. Springer, Heidelberg, April / May 2018.

[FS90] Uriel Feige and Adi Shamir. Zero knowledge proofs of knowledge in two rounds. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 526–544. Springer, Heidelberg, August 1990.

[GK96a] Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for NP. *Journal of Cryptology*, 9(3):167–190, June 1996.

[GK96b] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM Journal on Computing*, 25(1):169–192, 1996.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity for all languages in NP have zero-knowledge proof systems. *J. ACM*, 38(3):691–729, 1991.

[HRS15] Andreas Hülsing, Joost Rijneveld, and Fang Song. Mitigating multi-target attacks in hash-based signatures. Cryptology ePrint Archive, Report 2015/1256, 2015. https://eprint.iacr.org/2015/1256.

[JKMR09] Rahul Jain, Alexandra Kolla, Gatis Midrijanis, and Ben W. Reichardt. On parallel composition of zero-knowledge proofs with black-box quantum simulators. *Quantum Inf. Comput.*, 9(5&6):513–532, 2009.

[Kat08] Jonathan Katz. Which languages have 4-round zero-knowledge proofs? In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 73–88. Springer, Heidelberg, March 2008.

[KRR17] Yael Tauman Kalai, Guy N. Rothblum, and Ron D. Rothblum. From obfuscation to the security of Fiat-Shamir for proofs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 224–251. Springer, Heidelberg, August 2017.

[PW09] Rafael Pass and Hoeteck Wee. Black-box constructions of two-party protocols from one-way functions. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 403–418. Springer, Heidelberg, March 2009.

[SV03] Amit Sahai and Salil P. Vadhan. A complete problem for statistical zero knowledge. *J. ACM*, 50(2):196–249, 2003.

[Unr12] Dominique Unruh. Quantum proofs of knowledge. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 135–152. Springer, Heidelberg, April 2012.

[Unr15] Dominique Unruh. Revocable quantum timed-release encryption. *J. ACM*, 62(6):49:1–49:76, 2015.

[Unr16] Dominique Unruh. Computationally binding quantum commitments. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 497–527. Springer, Heidelberg, May 2016.

[Wat09] John Watrous. Zero-knowledge against quantum attacks. *SIAM J. Comput.*, 39(1):25–58, 2009.

[YZ20] Takashi Yamakawa and Mark Zhandry. Classical vs quantum random oracles. Cryptology ePrint Archive, Report 2020/1270, 2020. https://eprint.iacr.org/2020/1270.

[Zha12a] Mark Zhandry. How to construct quantum random functions. In *53rd FOCS*, pages 679–687. IEEE Computer Society Press, October 2012.

[Zha12b] Mark Zhandry. Secure identity-based encryption in the quantum random oracle model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 758–775. Springer, Heidelberg, August 2012.

Contents